HAW HAMBURG

Bachelorarbeit

Jan Poth

Wie KI und Big Data die Immobilienbewertung verändern könnten

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

Jan Poth

Wie KI und Big Data die Immobilienbewertung verändern könnten

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Angewandte Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuende Prüferin: Prof. Dr. Marina Tropmann-Frick
Zweitgutachter: Prof. Dr. Olaf Zukunft

Eingereicht am: 13. Oktober 2022

Jan Poth

Thema der Arbeit

Wie KI und Big Data die Immobilienbewertung verändern könnten

Stichworte

KI, Künstliche Intelligenz, Big Data, Maschinelles Lernen, Künstliche neuronale Netze, Immobilienbewertung, Immobilienpreisschätzung, Datenanalyse

Kurzzusammenfassung

Die Immobilienbewertung ist komplex und beruht auf einer Vielzahl von Faktoren, die sich auf den Immobilienpreis auswirken. Künstliche Intelligenz (KI) besitzt die Fähigkeit solche komplexen Zusammenhänge automatisiert zu erkennen. Diese Arbeit widmet sich der Erforschung und Evaluation von künstlicher Intelligenz für die automatisierte Schätzung von Immobilienpreisen in Deutschland. Die besten Schätzungen konnten durch einen eXtreme Gradient Boosting Regression Tree erzielt werden, der eine mittlere Abweichung von 81 773 € zum tatsächlichen Immobilienpreis besitzt. Der Median der absoluten Abweichungen beträgt 45 527 €. In den Ergebnissen hat sich herausgestellt, dass die Leistung von maschinellem Lernen der multiplen linearen Regression überlegen ist. Praktische Anwendung könnte eine automatisierte Immobilienbewertung in Form eines Unterstützungssystems für Immobilienmakler und für die Ersteinschätzung durch Immobilienbesitzer finden.

Gender Erklärung

Aus Gründen der besseren Lesbarkeit wird in dieser Arbeit auf die Verwendung von genderspezifischer Sprache verzichtet. Alle ausschließlich männlichen Formen sollen jedoch gleichermaßen alle Geschlechter ansprechen.

Jan Poth

Title of Thesis

How AI and Big Data could change real estate valuation

Keywords

AI, artificial intelligence, big data, machine learning, artificial neural networks, real estate valuation, property valuation, property price prediction, data analysis

Abstract

Real estate valuation is complex and based on a large number of factors that affect the price of real estate. Artificial intelligence (AI) has the ability to recognize such complex relationships in an automated way. This thesis is dedicated to the research and evaluation of artificial intelligence for the automated estimation of real estate prices in Germany. The best estimates could be obtained by an eXtreme Gradient Boosting Regression Tree, which has a average deviation of 81 773 € to the actual real estate price. The median absolute deviation is 45 527 €. The use of machine learning is superior to multiple linear regression in the performance of the results. In practice, an automated real estate valuation application can be used in the form of a support system for real estate agents and for initial appraisal by property owners.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

API Application Programming Interface.

CDN Content Delivery Network.

CSS Cascading Style Sheets.

HTML Hypertext Markup Language.

HTTP Hypertext Transfer Protocol.

JSON JavaScript Object Notation.

REST Representational State Transfer.

SQL Structured Query Language.

Symbolverzeichnis

NaN Not a Number (deutsch: keine Zahl). Ist ein spezieller Wert für Gleitkommazahlen, der angibt, dass der Wert keine Zahl darstellt.

1 Einleitung

Künstliche Intelligenz (KI) ist aus vielen Teilen des Lebens und der Industrie nicht mehr wegzudenken. So wird diese beispielsweise in der Medizin, der Robotik und Informatik erfolgreich eingesetzt. Die Bewertung von Immobilien ist komplex und beruht auf einer Vielzahl von Faktoren, die sich auf den Immobilienpreis auswirken. Solche komplexen Zusammenhänge könnten durch künstliche Intelligenz automatisiert erkannt werden. Einige Bereiche würden eine fundamentale Veränderung erfahren oder deutlich erleichtert werden, wenn eine automatisierte Bewertung von Immobilien möglich wäre. Dies würde zu einer Reduzierung des menschlichen Zeitaufwands und der damit verbundenen Kosten führen. Solch ein automatisiertes System hätte die Möglichkeit sowohl für Immobilienfirmen aller Größen als auch für die Ersteinschätzung durch Immobilienbesitzer von Nutzen zu sein. Deshalb widmet sich diese Arbeit der Erforschung und Evaluation von künstlicher Intelligenz für die automatisierte Schätzung von Immobilienpreisen in Deutschland.

1.1 Problemstellung

Die Bewertung von Immobilien ist ein manueller und aufwendiger Prozess. Dabei wird eine Immobilie von einem Makler besichtigt, die Eckdaten werden erfasst und diese Immobilie in verschiedenen Kategorien bewertet. Anhand dieser Faktoren wird anschließend ein Wert für die Immobilie ermittelt. Solch ein Wert beruht mitunter auf Erfahrungswerten des Maklers und ist eine subjektive Einschätzung, wodurch teilweise starke Abweichungen bei den Schätzungen entstehen. Zudem ist so eine Bewertung meist kostenintensiv und zeitaufwendig. Unerfahreneren Maklern kann es, in für sie unbekannten Gebieten, zunächst schwerfallen Immobilienpreise einzuschätzen. Zu der beschriebenen Problemstellung gibt es in Deutschland bisher keine öffentliche Forschungsarbeit.

1.2 Zielsetzung

Ziel dieser Arbeit ist die Umsetzung und Evaluation von maschinellem Lernen für die automatisierte Bewertung von Immobilien in Deutschland. Hierfür sollen zwei verschiedene Varianten des maschinellen Lernens umgesetzt werden. Die Varianten sollen jeweils durch ein eigenständiges Modell abgebildet werden. Die beiden Modelle des maschinellen Lernens sind ein eXtreme Gradient Boosting Regression Tree und ein Multi-Layer Perceptron Regressor. Zudem soll eine multiple lineare Regression durchgeführt werden, um diese mit den Ergebnissen des maschinellen Lernens zu vergleichen. Insgesamt werden demnach drei Modelle erstellt. Dazu soll zusätzlich eine prototypische Umsetzung in Form einer Benutzeroberfläche und eines Backend entwickelt werden. Die Benutzeroberfläche erfragt die essenziellen Daten zu einer Immobilie, welche daraufhin von den Modellen für eine Preisschätzung verwendet werden. Ziel ist es mit einer geringen Anzahl an Nutzereingaben eine möglichst genaue Preiseinschätzung zu erhalten. Die verwendeten Nutzereingaben sollen zudem nur aus Informationen bestehen, die leicht zu erfassen sind und keine besonderen Fachkenntnisse benötigen.

1.3 Verwandte Arbeiten

Die Forschung von künstlicher Intelligenz für die Schätzung von Werten nahm in den letzten Jahren stark zu. Für exakte Schätzungen müssen genügend Daten zum Trainieren zur Verfügung stehen. Es existiert allerdings weder eine Forschungsarbeit zur Anwendung von maschinellem Lernen für die Immobilienpreisschätzung noch ein öffentlicher großer Datensatz von Immobilienverkäufen in Deutschland.

Jedoch gibt es ein paar wissenschaftliche Arbeiten, die sich mit der Schätzung von Immobilienpreisen in anderen Ländern befassen. Bei der Vorhersage von Immobilienpreisen gibt es kein Modell, das in der Lage ist, für jede Beobachtung das richtige Ergebnis vorherzusagen.

Die Arbeit von Martin Foldvik Buodd und Erlend Jørgensen Derås aus 2020 beschäftigt sich mit der Schätzung von Immobilienpreisen für die Besteuerung von Immobilien in Norwegen. Dafür werden verschiedene Methoden des maschinellen Lernens mit der multiplen linearen Regression verglichen. Dabei konnten Gradient Boosting Verfahren und künstliche neuronale Netze die besten Ergebnisse erzielen. Diese waren ca. 20 % besser

als die aktuell verwendete MLR Methode in Norwegen. Beschrieben wurden die gesellschaftlichen Einflüsse von einer automatisierten Steuerberechnung, welche bei Fehlschätzungen falsche Steuerforderungen von den Immobilienbesitzern fordern könnte. Grund hierfür ist, dass eine automatisierte Schätzung nicht alle Merkmale erfassen kann, die in die Immobilienpreisschätzung einfließen. [3]

Die Anwendung von künstlichen neuronalen Netzen für die Vorhersage von Immobilienpreisen wurde 2004 bereits von Visit Limsombunchai, Christopher Gan und Minsoo Lee erforscht. Diese bezog sich nur auf den Häusermarkt in Christchurch, Neuseeland. Verglichen wurden die Ergebnisse erneut mit der multiplen linearen Regression. Dabei wurde herausgearbeitet, dass ein künstliches neuronales Netz eine erhöhte Flexibilität besitzt, weniger Kenntnis über die Daten benötigt sowie eine bessere Leistung für die Immobilienpreisschätzung erzielt. [21]

Diese Arbeit wird der bestehenden Literatur hinzufügen, wie ähnliche Ansätze des maschinellen Lernens bei der Vorhersage von Immobilienpreisen in Deutschland Anwendung finden könnten. Dazu wird die multiple lineare Regression, die in anderen Ländern bereits Anwendung findet, mit den in dieser Arbeit entwickelten Modellen des maschinellen Lernens verglichen. Hinzu kommt die praktische Anwendung, die beschreibt, wie solche Modelle in Form einer Benutzeroberfläche für Nutzer zur Verfügung gestellt werden kann.

1.4 Aufbau der Arbeit

Diese Arbeit ist in insgesamt fünf Kapitel unterteilt. In Kapitel 2 wird der Leser in die Immobilienbewertung, grundlegende Prinzipien der künstlichen Intelligenz und Big Data eingeführt. Zusätzlich wird der Datensatz, der in dieser Arbeit Anwendung findet, beschrieben und eine Datenexploration durchgeführt. Kapitel 3 widmet sich der Datenaufbereitung, dem Training der Modelle und der Erstellung einer Benutzeroberfläche sowie eines Backend. Die Auswertung und Diskussion der Ergebnisse findet in Kapitel 4 statt. Hier werden sowohl die Ergebnisse des maschinellen Lernens, der multiplen linearen Regression als auch die praktische Anwendung der Benutzeroberfläche und des Backend ausgewertet. Zudem wird eine Einschätzung für mögliche praktische Anwendungsgebiete vorgestellt. Zuletzt wird in Kapitel 5 rückblickend mit Bezug auf die Zielsetzung ein Fazit gezogen und ein Ausblick auf zukünftige Arbeiten gegeben.

2 Grundlagen

In diesem Kapitel wird sowohl in die Bewertung von Immobilien als auch in die Konzepte von künstlicher Intelligenz und Big Data eingeführt. Zusätzlich wird der verwendete Datensatz beschrieben und eine Datenexploration durchgeführt.

2.1 Immobilienbewertung

Im Regelfall besichtigt ein Makler eine Immobilie. Um diese zu bewerten, erfasst der Makler einige Eckdaten zu dem Objekt. Es wird anhand von Werten aus der Vergangenheit und dem allgemeinen Marktumfeld ein Wert geschätzt, der von vielen Faktoren abhängig ist.

Dabei ist zwischen Preis und Wert einer Immobilie zu unterscheiden. Der Wert gibt an, was allgemein für eine Immobilie bezahlt werden würde. Der Preis ist eine subjektive Einschätzung und bezeichnet, was im Einzelfall für eine Immobilie bezahlt wird. Dabei sind Wert und Preis einer Immobilie nicht immer gleich. [11, vgl. S. 5] In dieser Arbeit wird versucht den Preis von Immobilien zu bestimmen, da die Verkaufspreise von vergangenen Immobilien die Ausgangsdaten dafür sind. Statistisch gesehen liegt der Preis und der Wert der Immobilien bei sehr vielen Datensätzen nah aneinander. Somit ist zwar in Einzelfällen der Preis über oder unter dem Wert, in der Menge ist dies aber zu vernachlässigen. Deshalb werden in dieser Arbeit die Wörter Preis und Wert einer Immobilie oftmals als Synonym verwendet.

Essenziell für die Bewertung einer Immobilie ist die Lage der Immobilie. Die Lage lässt sich dabei in Makro- und Mikrolage unterteilen. [5, vgl. S. 30]

Die Makrolage beschreibt, ob sich die Immobilie in einer wirtschafts- und zukunftsstarken Region befindet. So sind beispielsweise Regionen wie Hamburg, Berlin und München

deutlich teurer als wirtschaftlich schwächere Regionen, wie zum Beispiel Teile von Ostdeutschland. [11] [5] [30]

Die Mikrolage hingegen wird von vielen kleineren Faktoren beeinflusst. Dazu können beispielsweise folgende gehören:

- Wie weit ist der nächste Supermarkt entfernt?
- Liegt die Immobilie in der Nähe einer lauten Straße, Bahnstrecke oder Flugschneise?
- Wie schnell sind öffentliche Verkehrsmittel zu erreichen und wie sind diese ausgebaut?
- Was für Freizeitangebote befinden sich in der Nähe?
- Wie schnell sind Naherholungsgebiete zu erreichen?

[5, vgl. S. 30-35]

Der Wert ist aber nicht nur von der Immobilie selbst und der Lage abhängig, sondern auch von verschiedenen Faktoren im allgemeinen Marktumfeld. Beeinflussen können den Preis zum Beispiel auch:

- Wie hoch sind die Zinsen für eine Finanzierung?
- Wie hoch ist die Inflation?
- Welche politischen Vorgaben können den Wert verringern oder erhöhen?

[5, vgl. S. 30-35]

Die Bewertung dieser großen Menge an Faktoren ist zeitaufwendig und bedarf einer Vielzahl an Daten, die für die Auswertung benötigt werden.

Wichtig zu erwähnen ist, dass verschiedene Makler eine Immobilie unterschiedlich bewerten würden. Einen genauen einzig wahren Preis zu bestimmen ist somit nahezu unmöglich. Es geht bei der Immobilienbewertung aber viel mehr darum eine Annäherung an den Wert der Immobilie zu erreichen. [5] [11]

2.1.1 Expertenwissen

Nach Rücksprache mit Mitarbeitern von Engel & Völkers wurden einige Informationen herausgearbeitet, die für die spätere Auswertung der Modelle notwendig sind. So wurden durch Makler einige Hauptmerkmale von Immobilien bestimmt, die für deren Preisbewertung besonders entscheidend sind. Die Wohnfläche verfügt dabei über den größten Einfluss, gefolgt von der Lage und dem Zustand der Immobilie. Der Immobilientyp ist ebenfalls wichtig für die Schätzung eines Verkaufspreises.

Des Weiteren haben interne Analysen ergeben, dass Bewertungen grundsätzlich für Immobilien mit einem Verkaufspreis von unter 1 500 000 € genauer sind. Deshalb beschränkt sich diese Arbeit auf Immobilien mit einem Verkaufspreis, der unter 1 500 000 € liegt.

Als Letztes ist zu erwähnen, dass automatisierte Bewertungen nur für Häuser und Wohnungen durchgeführt werden. In den Daten, die zu anderen Immobilientypen vorhanden sind, fehlen viele relevante Informationen. Folglich werden nur Preisbewertungen von Häusern und Wohnungen in dieser Arbeit durchgeführt.

2.2 Künstliche Intelligenz

Künstliche Intelligenz ist ein Teilgebiet der Informatik. [18, vgl. S. 1] Dieses Teilgebiet versucht Systeme, welche intelligente Verhaltensweisen nachahmen, nicht nur zu verstehen, sondern auch zu erstellen. Es besteht aus einer Vielzahl von weiteren Teilgebieten. [37, vgl. S. 6] [32, vgl. S. 1]

2.2.1 Maschinelles Lernen

Ein Teilgebiet der künstlichen Intelligenz, welches in dieser Arbeit Anwendung findet, ist das maschinelle Lernen. Es beschreibt die Fähigkeit von Computersystemen, sich an neue Umstände anzupassen, Muster zu erkennen und diese zu extrapolieren. [32, vgl. S. 2]

Maschinelles Lernen lässt sich dabei in drei Arten unterteilen:

Supervised Learning

Beim Supervised Learning (überwachtem Lernen) werden für das Training Paare aus Eingabewerten und Ausgabewerten zur Verfügung gestellt. Anhand dieser Paare wird versucht eine mathematische Funktion aufzustellen, welche die Eingabewerte zu den Ausgabewerten übersetzen kann. Diese mathematische Funktion wird dann auf unbekannte Eingabewerte angewendet. Damit kann berechnet werden, wie exakt die Vorhersage für unbekannte Eingaben ist, da diese mit dem jeweiligen tatsächlichen Wert verglichen werden kann. Für numerische Ausgabewerte wird diese Methodik *Regression* genannt. Bei endlichen kategorischen Ausgabewerten wird hingegen von *Klassifikation* gesprochen.

Unsupervised Learning

Beim Unsupervised Learning (unüberwachtem Lernen) sind zu den Eingaben keine Ausgabewerte bekannt. Das Ziel ist es die Struktur und Muster anhand des Datensatzes abzuleiten. Die Methodiken dabei sind meist das *Clustering* oder die *Dimensionsreduktion*.

Reinforcement Learning

Reinforcement Learning (verstärkendes Lernen) beschreibt die Strategie, dass ein sogenannter Agent basierend auf Erfolg und Bestrafung lernt. Dem Agenten wird nur die Rückmeldung gegeben, ob eine oder eine Vielzahl von Aktionen gut oder schlecht war. Für das Modell ist es das Ziel, die optimale (oder annähernd optimale) Strategie für die Umgebung zu erlernen.

[23] [32]

Diese Arbeit beschränkt sich auf das Supervised Learning (überwachtes Lernen), da für die Immobilien die jeweiligen Verkaufspreise bekannt sind. Da es sich bei den Immobilienpreisen um numerische Ausgabewerte handelt, wird von einer *Regression* gesprochen.

2.2.2 Künstliche neuronale Netze

Künstliche neuronale Netze sind auf die ursprünglichen Erkenntnisse von Frank Rosenblatt aus dem Jahre 1957 zurückzuführen. Diese Netze werden dem menschlichen biologischen Gehirn nachempfunden. In seiner Arbeit verwendet er den Begriff des Perzeptron. Ein Perzeptron ist die Idee eines einfachen Modells, welches zwei Eingaben und eine Ausgabe besitzt. [31] [19] [34]

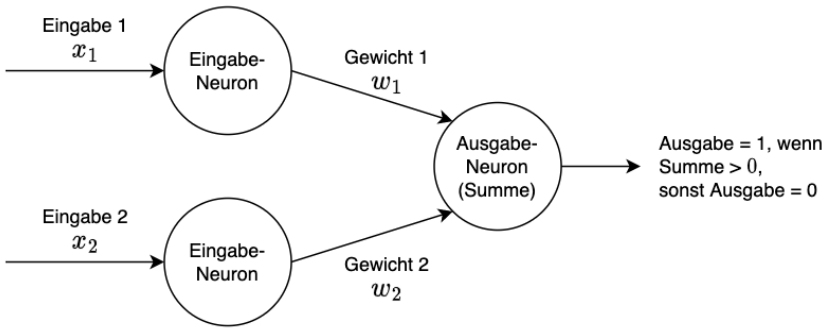


Abbildung 2.1: Beispiel: Perzeptron

In Abbildung 2.1 wird ein Perzeptron beispielhaft visualisiert. Dieses enthält die zwei Eingaben x_1 und x_2, welche jeweils in eines der beiden Eingabeneuronen führt. Die Eingabeneuronen sind dann mit dem Ausgabeneuron verbunden. Für die Verbindungen sind jeweils die Gewichte w_1 und w_2 angegeben. Das Ausgabeneuron hat demnach zwei Eingaben, welche in diesem Beispiel mit den Gewichten multipliziert und anschließend summiert werden. Die Formel zur Berechnung der Summe lautet folglich $x_1 \cdot w_1 + x_2 \cdot w_2$. Wenn die Summe größer als 1 ist, dann wird der Wert 1 ausgegeben. Andernfalls beträgt der Wert der Ausgabe 0. Eine solche Bestimmung des Ausgabewertes anhand der Eingabewerte wird als Aktivierungsfunktion bezeichnet. [34] [19]

Erweitert wurde das Perzeptron durch die Entwicklung von mehrschichtigen Netzen. Dabei gibt es zwischen den Eingabeneuronen und den Ausgabeneuronen noch mindestens eine weitere Schicht. Diese Schichten werden als „Hidden Layer“ bezeichnet. [17] [34]

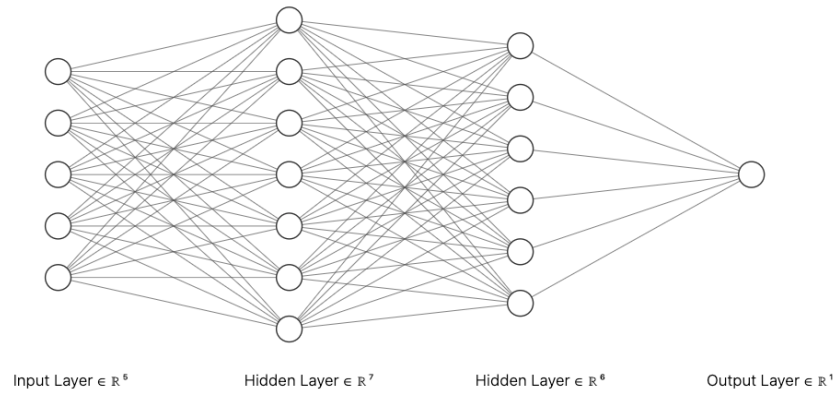


Abbildung 2.2: Beispiel: Visualisierung eines künstlichen neuronalen Netzes

In Abbildung 2.2 wird ein mehrschichtiges künstliches neuronales Netz dargestellt. Die Eingabeneuronen werden als „Input Layer“ und die Ausgabeneuronen als „Output Layer“ bezeichnet. Ein künstliches neuronales Netz gibt in dem „Output Layer“ eine oder mehrere Zahlen aus. In dem Beispiel gibt es exakt eine Zahl aus, da es nur ein Ausgabeneuron gibt. Um zu bestimmen, wie stark das Ergebnis des künstlichen neuronalen Netzes von dem Erwartungswert abweicht, wird ein Fehler mit einer Fehlerfunktion berechnet. [34]

Ziel eines künstlichen neuronalen Netzes ist es den Fehler zu minimieren. Ein weit verbreitetes Verfahren ist die sogenannte „Backpropagation“. Dabei werden die Gewichte des Netzes basierend auf dem Fehler angepasst. Dazu wird ein Gradientenabstieg verwendet. Die Anpassung der Gewichte wird durch einen Optimierer durchgeführt. [19] [34]

2.3 Big Data

Big Data ist nicht eindeutig definiert und wird meist als Sammelbegriff verwendet. Grundsätzlich wird meist von Big Data gesprochen, wenn die Datenhaltung, Verarbeitung oder Analyse konventionelle Hardware übersteigt. Dabei wird meist von den fünf „Vs“ gesprochen:

Volume (Umfang, Datenvolumen) Der Datenbestand ist besonders umfangreich. Dies kann die Dateigröße oder die Anzahl an Datensätzen beschreiben.

Variety (Vielfalt) Der Datenbestand enthält strukturierte, semi-strukturierte und unstrukturierte Daten.

Velocity (Geschwindigkeit) Die benötigte Geschwindigkeit, in der Daten verarbeitet werden müssen. Erfordert oftmals die Auswertung von Datenströmen in Echtzeit.

Value (Unternehmenswert) Die Nutzung von Big Data soll den Unternehmenswert potenziell steigern. Investitionen in Big Data und die Infrastruktur sollen einen erheblichen Mehrwert liefern.

Veracity (Wahrhaftigkeit) Viele Daten sind vage oder ungenau. Große Datenbestände bedeuten demnach nicht automatisch eine bessere Auswertungsqualität.

[8, vgl. S. 3-6]

Für aktuelle Arbeiten sind oftmals die Daten deutlich wichtiger als der Algorithmus, der ausgewählt wird. [32, S. 27-28]

Eine Zuordnung der fünf „Vs“ auf diese Arbeit findet nach der Datenexploration in Abschnitt 2.4.3 statt.

Für die Verarbeitung wird BigQuery von der Google Cloud verwendet. Damit lassen sich die Daten mittels Structured Query Language (SQL) innerhalb weniger Sekunden filtern, selektieren und anschließend exportieren. [1]

2.4 Datensatz

2.4.1 Beschreibung

Der Datensatz, der in dieser Arbeit verwendet wird, stammt von Engel & Völkers. Er besteht aus einer Tabelle mit insgesamt 2 110 441 Zeilen. Die Größe beträgt 3,4 Gigabyte.

Herkunft der Daten sind verschiedene interne Tools, in denen Immobiliendaten seit 1994 gespeichert werden. Diese werden zu einem Datensatz zusammengefügt, der in dieser Arbeit verwendet wird.

Die Daten enthalten verschiedene Informationen über die jeweilige Immobilie. Dazu zählen beispielsweise der Verkaufspreis, der Immobilientyp, die Wohnfläche, die Postleitzahl und andere grundlegende Merkmale. Eine gesamte Auflistung aller 29 Attribute ist in Tabelle A.1 zu finden. Attribute werden auch als Merkmale oder Features bezeichnet. In dieser Arbeit wird das Wort Attribut für die Nennung von Merkmalen oder Features verwendet.

Diese Arbeit beschränkt sich auf den deutschen Markt. Von den 2 110 441 Daten sind 1 109 642 aus Deutschland. Zusätzlich ist der bekannte Netto Verkaufspreis eine Voraussetzung, da dieser vorhergesagt bzw. geschätzt werden soll. 178 555 Immobilien sind mit einem Netto Verkaufspreis versehen. Die Schnittmenge aus beiden Mengen, also Daten aus Deutschland und mit einem Verkaufspreis, enthält insgesamt 99 119 Daten.

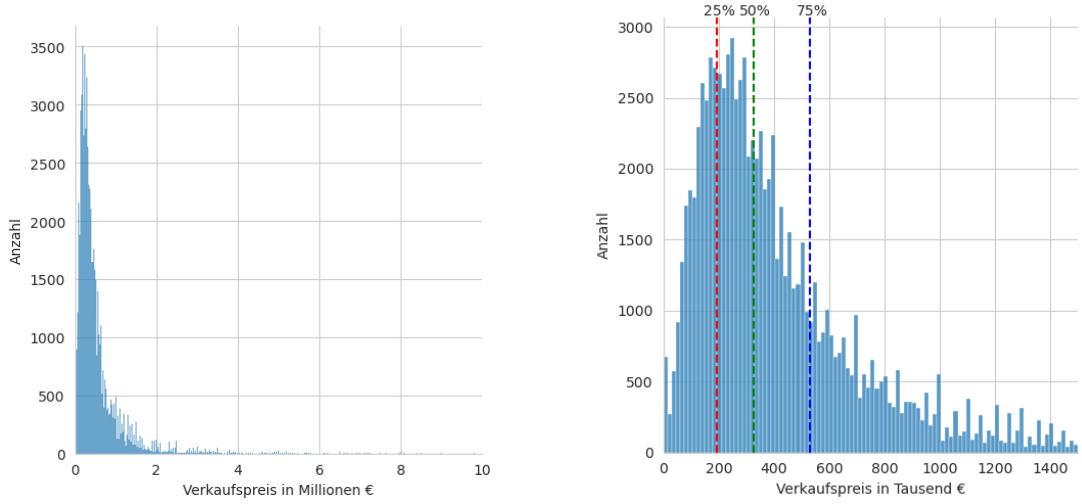
Wie in dem Expertenwissen aus Abschnitt 2.1.1 beschrieben, beschränkt sich diese Arbeit nur auf Immobilien mit einem Verkaufspreis von unter 1 500 000 €. Dazu wird nur der Immobilientyp „Haus“ und „Wohnung“ verwendet.

Die Schnittmenge aus Immobilien, die alle diese Kriterien erfüllen, enthält 93 767 Elemente.

2.4.2 Datenexploration

Visualisierungen helfen dabei einen Überblick über den verwendeten Datensatz zu erhalten und diesen auf Plausibilität zu analysieren.

Die Verteilung der Verkaufspreise wurde in den Abbildungen 2.3 dargestellt.



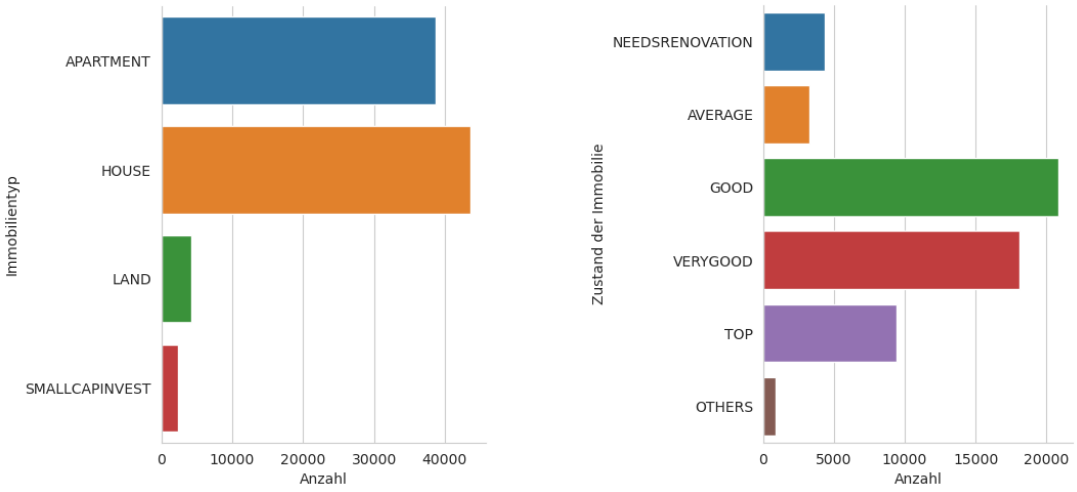
(a) Verkaufspreis bis 10 000 000 €

(b) Verkaufspreis bis 1 500 000 €

Abbildung 2.3: Verteilung der Verkaufspreise aller Immobilien im Datensatz

In Abbildung 2.3a ist zu erkennen, dass nur wenig Daten ab einem Verkaufspreis von ca. 1 500 000 € vorliegen. Deshalb wird in 2.3b die x-Achse auf 1 500 000 € limitiert. Somit lässt sich genauer betrachten, dass nur 25 % der Verkäufe einen Verkaufspreis unter 195 000 € haben. Weitere 25 % haben einen Verkaufspreis von über 530 000 €. Der durchschnittliche Verkaufspreis liegt bei 325 000 €.

In den Abbildungen 2.4 wird die Verteilung des Immobilientyps und des Immobilienzustands im Datensatz visualisiert. Ungefähr 95 % der Verkäufe ist auf Häuser und Wohnungen zurückzuführen, wie in 2.4a dargelegt. Wie bereits im Expertenwissen in Abschnitt 2.1.1 beschrieben, werden demnach nur Häuser und Wohnungen in dieser Arbeit verwendet. In Abbildung 2.4b ist erkennbar, dass sich die Verteilung des Immobilienzustands hauptsächlich auf *GOOD*, *VERYGOOD* und *TOP* beschränkt, wobei andere Zustände ebenfalls vertreten sind. Die Bewertung des Zustands einer Immobilie ist nicht einheitlich und unterliegt dem Makler.

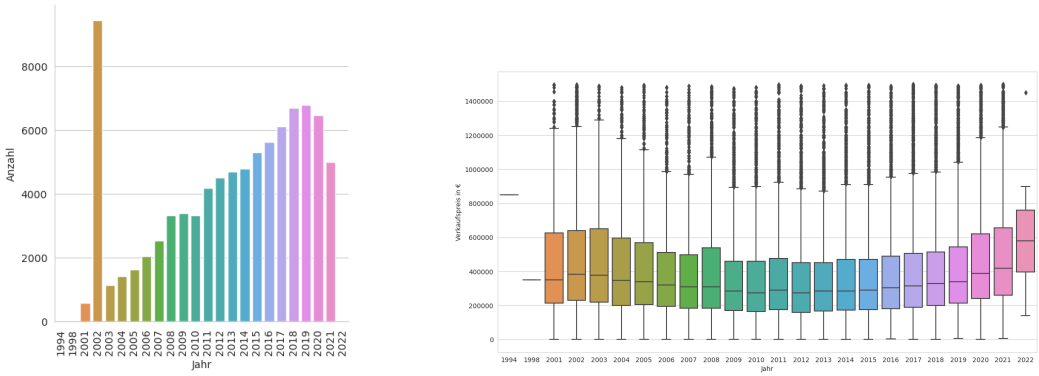


(a) Verteilung des Immobilientyps im Datensatz

(b) Verteilung des Zustands der Immobilien im Datensatz

Abbildung 2.4: Verteilung von Typ und Zustand der Immobilien im Datensatz

In den Graphen der Abbildungen 2.5 wurde der zeitliche Verlauf der Immobiliendaten von 1994 bis 2022 dargestellt.



(a) Anzahl der verkauften Immobilien von 1994 bis 2022

(b) Entwicklung der durchschnittlichen Verkaufspreise von 1994 bis 2022

Abbildung 2.5: Verteilung der Immobilienverkäufe vom Jahre 1994 bis 2022

Es ist zu erkennen, dass bis einschließlich dem Jahre 2000 kaum Daten vorliegen. Hingegen sind sehr viele Verkäufe auf das Jahr 2002 datiert. Dies ist darauf zurückzuführen, dass vergangene nicht zentral erfasste Daten nachträglich auf den Zeitpunkt 2002 datiert wurden. Für das Jahr 2022 sind weniger Daten verfügbar, da nur abgeschlossene Verkäufe mit einem Verkaufspreis enthalten sind und der Datensatz nur Daten bis Anfang 2022

enthält. Von 2003 bis 2019 ist ein annähernd linearer Anstieg der Verkäufe zu erkennen. 2020 stagnierte das Wachstum und 2021 sind die Verkäufe leicht zurückgegangen. Internen Analysen zufolge ist dies durch die Corona-Pandemie bedingt.

Für die Verwendung in dieser Arbeit wird die Jahreszahl des Inserats verwendet.

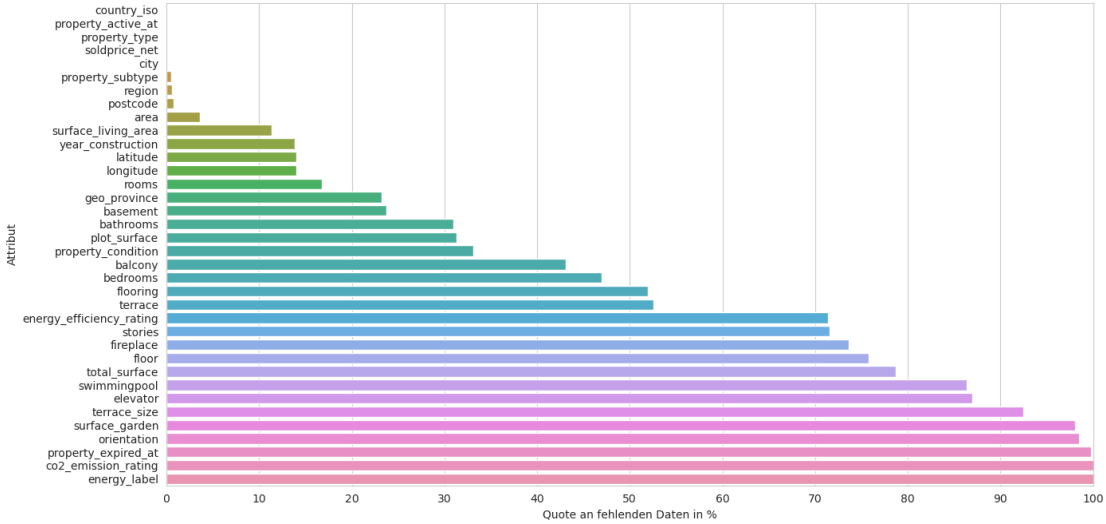


Abbildung 2.6: Quote an fehlenden Daten für alle Attribute im Datensatz

Es sind nicht zu jeder Immobilie alle Attribute vorhanden. In Abbildung 2.6 ist dargestellt, wie viele Daten zu den jeweiligen Attributen fehlen. Dabei ist zu erkennen, dass sich einige Attribute nicht für die weitere Verwendung eignen, da diese nur in niedrigem Prozentsatz vorhanden sind. Für Attribute wie *property_type*, *city* oder *year_active* sind immer Daten verfügbar. Die Varianz lässt sich dadurch erklären, dass einige Daten nicht immer erhoben werden oder erhoben wurden. Dies ist damit zu begründen, das viele Eingaben optional sind und nicht von allen Maklern gleichmäßig gepflegt werden.

In Abbildung 2.7 wird die Korrelation zwischen allen Attributen dargestellt. Eine besonders hohe Korrelation ist zwischen Zimmer (*rooms*), Schlafzimmer (*bedrooms*) und Badezimmer (*bathrooms*) erkennbar. Ebenfalls lässt sich zwischen dem Baujahr und der Anzahl der Stockwerke des Gebäudes eine Korrelation feststellen. Die Korrelation zwischen der Wohnfläche und der Anzahl an Stockwerken eines Gebäudes lässt sich erklären. So haben viele Häuser mit mehreren Stockwerken eine höhere Wohnfläche. Jedoch ist diese Korrelation auf Wohnungen nur bedingt übertragbar.

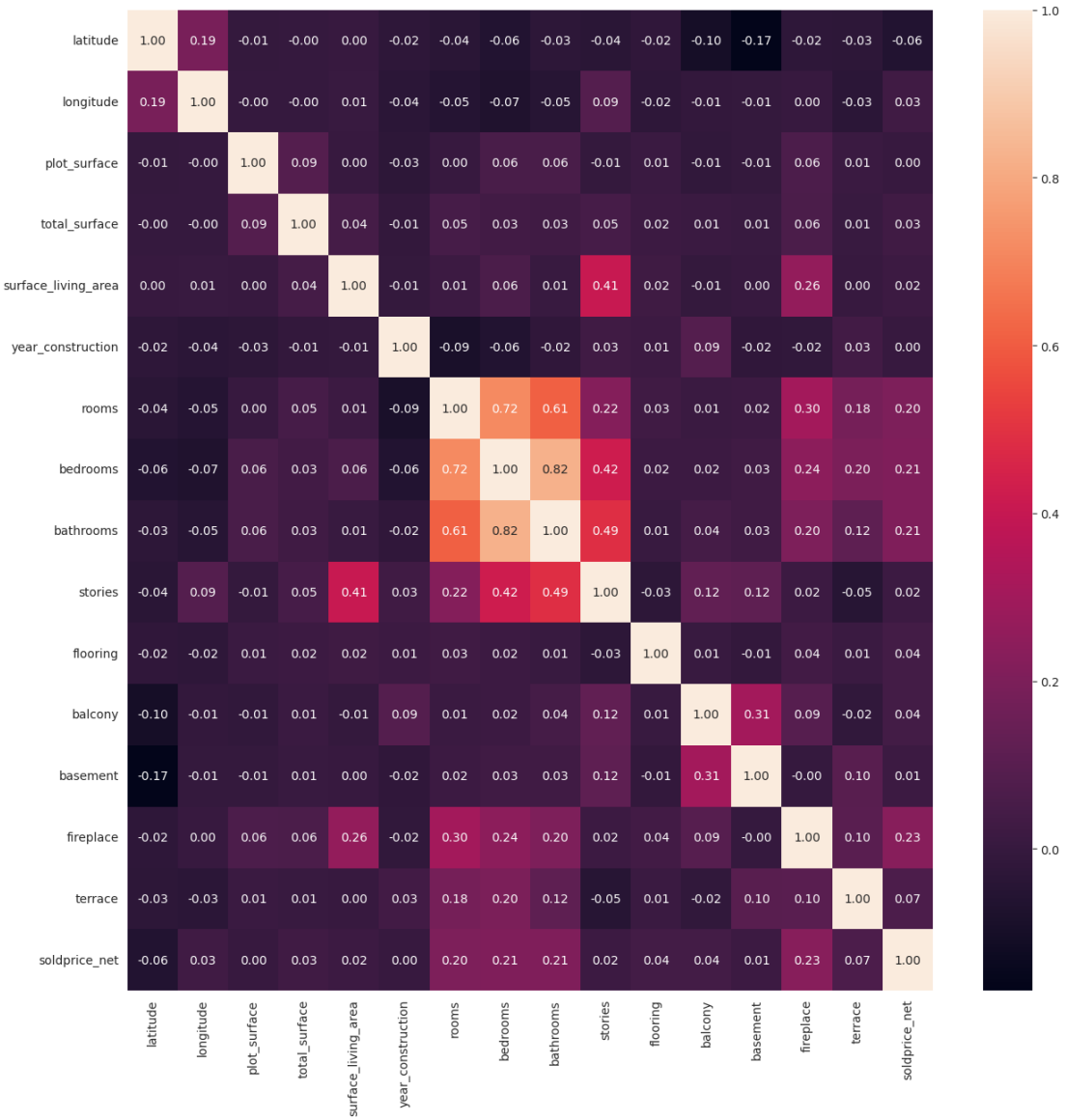


Abbildung 2.7: Korrelationsmatrix aller Attribute im Datensatz

2.4.3 Anwendung von Big Data

Nach der Definition von Big Data in Abschnitt 2.3 finden in dieser Arbeit mindestens zwei der fünf „Vs“ Anwendung. Zum einen die Vielfalt (Variety), da ein Teil der Daten strukturiert und ein anderer Teil semi-strukturiert vorliegt. Zudem wird der Unternehmenswert (Value) sowohl durch die Speicherung aller Immobilienverkäufe als auch die Nutzung dieser potenziell stark gesteigert.

Der Umfang (Volume) des Datensatzes beträgt eine Größe von 3,4 Gigabyte in 2 110 441 Einträgen. Dennoch trifft dieses „V“ nicht zwangsweise zu. Bei Big Data können hunderte Millionen Datensätze Anwendung finden. Die Datengröße kann dabei im Bereich von Tera-, Peta-, Exabyte oder größeren Dimensionen liegen.

Die Velocity (Geschwindigkeit) ist für diese Arbeit irrelevant. Das Training in dieser Arbeit findet einmalig statt. Es ist kein Datenstrom in Echtzeit zu verarbeiten.

Bei der Veracity (Wahrhaftigkeit) ist eine Einordnung nicht eindeutig möglich. Je mehr Daten für das Training zur Verfügung stehen, desto zutreffender sind die Ergebnisse. Dies ist jedoch nur die Folge, wenn die Datenqualität sichergestellt ist. Da ein Großteil der Daten in ausreichender Qualität vorliegt, trifft dieses „V“ nicht unbedingt zu. Dennoch müssen mit einer Datenaufbereitung einige der Daten entfernt werden, die nur vage oder ungenau sind. Im nächsten Kapitel wird die Datenaufbereitung in Abschnitt 3.1 durchgeführt.

3 Umsetzung

In diesem Kapitel wird die Umsetzung der Preisschätzung von Immobilien beschrieben. Dazu wird zunächst die Datenaufbereitung durchgeführt. Anschließend werden das Training und die ausgewählten Algorithmen erläutert. Die Algorithmen werden daraufhin in den Modellen verwendet. Die Modelle des maschinellen Lernens sind zum einen ein eXtreme Gradient Boosting Regression Tree und zum anderen ein Multi-Layer Perceptron Regressor. Zusätzlich wird ein Modell der multiplen linearen Regression als Vergleich erstellt. Folgend wird beschrieben, wie die Modelle exportiert werden, um diese in der praktischen Anwendung in Abschnitt 3.6 nutzen zu können. Zuletzt werden die Modelle prototypisch umgesetzt, in dem ein Backend und eine Benutzeroberfläche entwickelt werden.

3.1 Datenaufbereitung

Die Datenaufbereitung ist ein essenzieller Bestandteil des maschinellen Lernens. Dafür werden in dieser Arbeit verschiedene Techniken in Kombination genutzt, um den Datensatz für das Training vorzubereiten.

3.1.1 Numerische Werte

Attribute des Datensatzes, die Zahlenwerte darstellen (siehe Tabelle A.1), werden durch Subtraktion des Mittelwertes standardisiert und basierend auf dem 25. und 75. Quantil skaliert. Dies geschieht unabhängig für jedes Attribut einzeln. Dazu werden zuerst für jedes Attribut die relevanten Statistiken mit dem Trainingsdatensatz berechnet. Der Median und Interquartilsabstand (IQR) werden dafür gespeichert. Dadurch werden alle Daten mit den gleichen Faktoren transformiert. Im Interquartilsabstand liegen genau die Hälfte der beobachteten Werte. Der IQR des Datensatzes ist in Abbildung 2.3b mit senkrechten roten und blauen gestrichelten Linien dargestellt. Für eine Transformation der

Daten X werden diese zuerst zentriert. Dazu wird $X - m$ berechnet, wobei m der Median ist. Anschließend wird die Skalierung $\frac{X}{s}$ mit der IQR s durchgeführt. Die gesamte Formel lautet dann $\frac{X-m}{s}$. [23]

Für die Auswertung lässt sich somit der ursprüngliche Zahlenwert zurückrechnen. Dazu wird die Formel umgestellt zu $X = (X \cdot s) + m$.

3.1.2 Kategorische Werte

Die kategorischen Werte sind, wie in Tabelle A.1 aufgelistet, als Strings gespeichert. Strings können nicht von den Modellen als Eingabe erfasst werden. Deshalb müssen diese umgewandelt werden und in dem Datensatz ersetzt werden. Dafür werden vor dem Training die gesamten Trainingsdaten durch einen *Encoder* verarbeitet. Mithilfe des *Encoders* ist es nach dem Training möglich die ersetzen Werte wieder in die ursprünglichen Strings zurückzuwandeln. [37, vgl. S. 200-201]

Die zwei klassischen Varianten für das *Encoding* von Strings sind das *Ordinal Encoding* und das *One Hot Encoding*.

Beim *Ordinal Encoding* werden die Strings in dem Datensatz durch eindeutige Zahlen ersetzt. Dazu wird jedem eindeutigem String eine aufsteigende Zahl zugeordnet. Für ein Attribut mit N einzigartigen Strings werden dementsprechend die natürlichen Zahlen 0 *bis* $N - 1$ vergeben. [37, vgl. S. 200-201]

Ein Beispiel eines Attributs *Stadt* mit 4 Einträgen der Werte *[Hamburg, Berlin, Hamburg, München]* würde die Werte folgendermaßen ersetzen:

Eintrag	**Stadt**
Eintrag 1	0
Eintrag 2	1
Eintrag 3	0
Eintrag 4	2

Tabelle 3.1: Beispiel: Ordinal Encoding

Beim *One Hot Encoding* hingegen werden die Strings in dem Datensatz durch Spalten mit binären Werten ersetzt. Dabei wird die jeweilige Spalte von kategorischen Attributen

entfernt. Es werden dann für jedes der kategorischen Attribute mit N einzigartigen Werten N zusätzliche Spalten dem Datensatz hinzugefügt. Anschließend werden für jeden Wert alle Spalten binär entweder mit zutreffend (1 / wahr) oder unzutreffend (0 / falsch) markiert. Dies wird für jedes kategorische Attribut durchgeführt. [37, vgl. S. 203-205]

Das oben genannte Beispiel eines Attributs *Stadt* mit 4 Einträgen der Werte *[Hamburg, Berlin, Hamburg, München]* würde die Werte folgendermaßen ersetzen:

Eintrag	**Stadt_Hamburg**	**Stadt_Berlin**	**Stadt_München**
Eintrag 1	1	0	0
Eintrag 2	0	1	0
Eintrag 3	1	0	0
Eintrag 4	0	0	1

Tabelle 3.2: Beispiel: One Hot Encoding

Ein *Ordinal Encoder* besitzt den Nachteil, dass das Modell fälschlicherweise einen Zusammenhang zwischen der Nähe der Zahlen und der Bedeutung interpretieren könnte. Dennoch wird ein *Ordinal Encoder* für diese Arbeit verwendet.

Grund dafür ist, dass *One Hot Encoding* die Dimensionalität des Datensatzes enorm erhöhen würde. Für das Attribut *city* gibt es 10 513 einzigartige Werte. Zudem gibt es für andere Attribute, wie *postcode*, 6 194 einzigartige Werte. Zusammen würde die Dimensionalität mit allen kategorischen Attributen auf über 20 000 steigen. Eine Erhöhung der Dimensionalität auf diese Größenordnung würde den Arbeitsspeicherverbrauch des Systems und die allgemeine Trainingszeit deutlich erhöhen und die Verwendung unpraktikabel machen.

3.1.3 Auffüllen von fehlenden Werten

Da der Datensatz, wie in Abbildung 2.6 dargestellt, viele fehlende Werte enthält, müssen diese für die Verwendung künstlich aufgefüllt werden. Grund hierfür ist, dass die Modelle nicht mit fehlenden Daten oder NaN funktionieren. In dieser Arbeit werden die fehlenden Zahlen durch -1 ersetzt, da diese nicht im Datensatz vorkommen. Fehlende Strings werden durch *missing_value* aufgefüllt. Eine Auffüllung mit künstlichen Werten, wie dem Median oder der am häufigsten vorkommenden Werte, würde nicht wiedergeben, dass der Wert ursprünglich nicht vorhanden war.

3.1.4 Reduktion der Attribute

Um ein effizienteres und schnelleres Training zu ermöglichen, werden einige von den 29 Attributen aus Tabelle A.1 entfernt. Zunächst werden die Attribute entfernt, die aufgrund von zu hoher Quote an fehlenden Daten unbrauchbar sind. Zusätzlich werden im Laufe der Arbeit einige Attribute entfernt, die unter 2 % Feature Importance beim besten Modell haben (siehe Abbildung 4.2).

Darüber hinaus wurden die Attribute aus der Kategorie Energie (siehe Tabelle A.1) ebenfalls aus dem Datensatz herausgenommen. Dies ist damit zu begründen, dass diese Informationen nur mittels eines Energieausweises zu erhalten sind. Energieausweise müssen von Fachpersonal erstellt werden und erfüllen demnach nicht die Zielsetzung in Abschnitt 1.2.

Einige weitere Attribute wurden aufgrund von hoher Korrelation mit anderen Attributen, die eine deutlich höhere Feature Importance (FI) haben, entfernt.

Die Attribute über die Anzahl der Schlafzimmer, Badezimmer und gesamten Zimmer weisen eine hohe Korrelation auf. So wurden beispielsweise Schlafzimmer (*bedrooms*: 0,007 FI) und Badezimmer (*bathrooms*: 0,019 FI) entfernt. Nur das Attribut Zimmer (*rooms*: 0,041 FI) blieb erhalten, da dies die höchste Feature Importance besitzt.

Übrig bleiben demnach 14 Attribute: *surface_living_area*, *year_construction*, *plot_surface*, *Latitude*, *longitude*, *rooms*, *year_active*, *city*, *region*, *postcode*, *area*, *property_condition*, *property_type* und *property_subtype*.

Die Reduktion der Attribute wurde im Laufe der Arbeit durchgeführt, damit keine Attribute entfernt werden, die potenziell positive Auswirkungen auf das Training der Modelle haben könnten.

Für das Attribut Immobilientyp wurden die Auswahlmöglichkeiten, wie bereits im Expertenwissen in Abschnitt 2.1.1 erwähnt, reduziert. Übrig bleiben dort die Immobilientypen *Haus* und *Wohnung*.

3.1.5 Aufteilung des Datensatzes

Für ein besseres Training und eine anschließende Auswertung wird der Datensatz in Trainingsdatensatz und Testdatensatz aufgeteilt. Mit dem Trainingsdatensatz werden

die Modelle trainiert. Der Testdatensatz wird nach dem Training dazu verwendet, um auszuwerten, wie das Modell aus den Trainingsdaten abstrahiert und wie genau die Vorhersagen sind. Dazu wird der Testdatensatz dem Modell nicht während des Trainings zur Verfügung gestellt, damit die Auswertungen auf unbekannten Daten basieren. Die Aufteilung sieht wie folgt aus:

- 80 % Trainingsdatensatz
- 20 % Testdatensatz

Die Aufteilung wurde so gewählt, dass ausreichend Daten für das Training mit dem Trainingsdatensatz zur Verfügung stehen. Des Weiteren sind genügend Daten im Testdatensatz vorhanden, um die Leistung der Modelle repräsentativ bewerten zu können.

Für das Training der Modelle des maschinellen Lernens wurde jeweils ein Validierungsdatensatz aus dem Trainingsdatensatz erzeugt. Dies wird in den jeweiligen Abschnitten der Modelle genauer erläutert.

3.2 Training

Das Training wird mittels eines selbst programmierten Python-Skripts durchgeführt. Dieses beinhaltet sowohl alle beschriebenen Schritte aus der Datenaufbereitung in Abschnitt 3.1, die Pipelines 3.1, 3.3 und 3.5 als auch die Berechnung der Metriken. Für jedes Modell werden anschließend in dem Skript die Graphen für die Auswertung und Diskussion in Kapitel 4 generiert. Eine Pipeline beschreibt die notwendigen Schritte und Transformationen, die durchgeführt werden, um die Daten für die Modelle vorzubereiten und das Training durchzuführen. Nach dem Training können für die Auswertung und die Vorhersage von neuen Werten die Transformationen vor dem Training umgekehrt werden. Folglich können Eingaben in dem Format des ursprünglichen Datensatzes in die Pipeline eingegeben werden und es werden Verkaufspreise von Immobilien zurückgegeben. Dabei werden alle drei Modelle in separaten Pipelines voneinander trainiert.

Um die optimalen Parameter für beide Algorithmen des maschinellen Lernens zu finden wurde eine automatisierte Suche durchgeführt. Dazu wurde eine Parametermatrix angelegt, die alle möglichen Kombinationen enthält. Zu diesen Parametern zählen beispielsweise die Anzahl an Iterationen, die Aktivierungsfunktion, der Optimierer, die Lernrate und die Größe der Schichten im künstlichen neuronalen Netz. Für jeden Eintrag in der

Parametermatrix, also jeder Kombination, wird die Pipeline des jeweiligen Modells trainiert und ausgewertet. Die Parameter, die zu den besten Ergebnissen geführt haben, werden daraufhin für das finale Training verwendet.

Bei der technischen Umsetzung des Python-Skripts wird die Open-Source Bibliothek *scikit-learn* verwendet. Unter dem Slogan „Machine Learning in Python“ werden Hilfswerkzeuge im Bereich des maschinellen Lernens zur Verfügung gestellt. [33]

3.3 Maschinelles Lernen

Zentral für diese Arbeit ist die Umsetzung der Modelle des maschinellen Lernens. Dazu werden beide verwendeten Techniken für die Modelle des maschinellen Lernens eingeführt und die jeweilige Umsetzung dargelegt.

3.3.1 eXtreme Gradient Boosting Regression Tree

2001 entwickelte Jerome H. Friedmann ein neues Konzept basierend auf Entscheidungsbäumen. Die Idee ist es, Bäume basierend auf den Fehlern des Modells zu stapeln. [10] Das Konzept ist ein sogenanntes *Ensemble Modell*. Dabei werden mehrere „schwache Modelle“ zu einem „starken Modell“ zusammengesetzt. Die „schwachen Modelle“ werden dabei basierend auf dem aktuellen Fehler betrachtet und nicht auf einer gewichteten Version des initialen Trainingsdatensatzes. [37, vgl. S. 189]

2016 wurde die Idee von Chen und Guestrin zu einem neuem Gradient Boosting Algorithmus, dem eXtreme Gradient Boosting (kurz XGBoost) weiterentwickelt. [4] Die Verbesserungen liegen dabei in den mathematischen Details und in der Implementierungsweise. Für diese Arbeit liegen die Verbesserungen primär darin, dass XGBoost schneller konvergiert, meistens zutreffendere Ergebnisse liefert und mit mehr Parametern optimiert werden kann. [37, vgl. S. 189]

Die Popularität von XGBoost entstand durch dessen erfolgreiche Verwendung in Wettbewerben für maschinelles Lernen. [40]

In dieser Arbeit wird die XGBoost Python Bibliothek verwendet, die mit *scikit-learn* zusammen genutzt werden kann. [39]

Als Ziel wird dem XGBoost Algorithmus die Minimierung des *Tweedie* Fehlers vorgegeben. Die *Tweedie* Abweichung für Regression Modelle wurde 2016 von Wagner H. Bonat und Célestin C. Kokonendji beschrieben. [2] Für die Berechnung des Fehlers wird in jeder Iteration folgende Formel verwendet:

$$tweedie\_loss(\hat{y}, y) = 2 \cdot \left(\frac{(\max(y, 0))^2}{(1 - \text{power})(2 - \text{power})} - \frac{y(\hat{y})^{1-\text{power}}}{1 - \text{power}} + \frac{(\hat{y})^{2-\text{power}}}{2 - \text{power}} \right)$$

Wobei $\hat{y}$ ein Tensor der tatsächlichen Werte und y ein Tensor der vorhergesagten Werte ist. In Python werden diese als Array aus Gleitkommazahlen dargestellt. Die Gleitkommazahlen sind dabei jeweils die Preise der Immobilien. Zusätzlich lässt sich mit p die Abweichung weiter optimieren. Die besten Ergebnisse wurden mit $p = 1,5$ erzielt. Dabei müssen allerdings als Bedingung alle Zielwerte ≥ 0 und alle Vorhersagen > 0 sein. Dies ist diesem Anwendungsfall gegeben. [14, vgl. S. 134-140] [2]

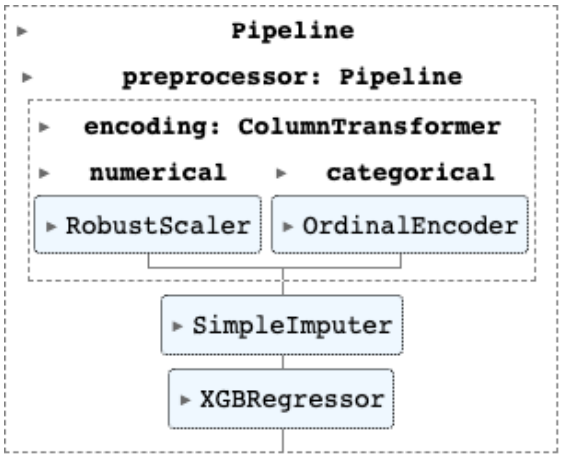


Abbildung 3.1: Pipeline des eXtreme Gradient Boosting Regression Tree

In der Abbildung 3.1 ist die Pipeline des eXtreme Gradient Boosting Regression Tree (XGBRegressor) visualisiert. Diese stellt die technische Implementierung des eXtreme Gradient Boosting Regression Tree Verfahrens dar. Die Pipeline beginnt mit der Datenaufbereitung ab dem Bereich „preprocessor“. Zunächst werden dabei die Werte korrekt kodiert. Numerische Werte werden normalisiert („RobustScaler“), wie in Abschnitt 3.1.1 beschrieben. Kategorische Werte werden hingegen durch eindeutige Zahlen ersetzt („OrdinalEncoder“), wie in Abschnitt 3.1.2 näher ausgeführt. Anschließend werden fehlende Werte aufgefüllt („SimpleImputer“), wie in 3.1.3 erklärt. Nach der Datenaufbereitung werden die bereinigten Daten in das Modell, den *XGBRegressor*, geleitet.

Um Overfitting zu verhindern wird Early Stopping eingesetzt. Dabei wird dem Modell der Testdatensatz als Validierungsdatensatz zur Verfügung gestellt. Dieser Datensatz wird nicht für das Training genutzt, sondern nur um zu überprüfen, ob das Training frühzeitig abgebrochen werden sollte. Dazu wird der mittlere absolute Fehler (beschrieben in Abschnitt 4.1) für den Validierungsdatensatz berechnet. Verbessert dieser sich über 100 Iterationen nicht, wird das Training beendet.

3.3.2 Multi-Layer Perceptron Regressor

Der Multi-Layer Perceptron Regressor (kurz MLPRegressor) ist ein künstliches neuronales Netz. Diese wurden in Abschnitt 2.2.2 eingeführt. Multi-Layer gibt dabei an, dass das künstliche neuronale Netz aus mehreren Schichten besteht. Das Modell versucht den Fehler in Form der Wurzel der mittleren quadratischen Abweichung (kurz RMSE) zu minimieren. Dazu wird ein stochastischer Gradientenabstieg genutzt. [37, S. 17]

Als Aktivierungsfunktion wird die *ReLU* Aktivierungsfunktion verwendet. *ReLU* steht für *Rectified Linear Units*. Die mathematische Formel für diese lautet $f(x) = max(0, x)$. Wobei x der Wert ist, der in die Aktivierungsfunktion eingesetzt wird. Wie in der Formel angegeben, geben alle Werte < 0 den Wert 0 zurück. Für Werte ≥ 0 wird jeweils der Wert x zurückgegeben. Diese simple Formel ermöglicht ein schnelleres und stabileres Training. [25]

Adam wird als Optimierer verwendet, welcher auf dem Gradientenverfahren basiert. Dieser ist ein sogenannter stochastischer Optimierer. Der Name Adam steht für *Adaptive Moment Estimation*. Adam ist einer der bekanntesten Optimierer und wird eingesetzt, da dieser den Gradientenabstieg beschleunigt und die Schwankungen reduziert. [16] [37, S. 239]

Das verwendete künstliche neuronale Netz des Multi-Layer Perceptron Regressor wird in Abbildung 3.2 visualisiert. Der „Input Layer“ besteht aus 14 Neuronen. Diese stellen die 14 Attribute aus Abschnitt 3.1.4 dar. Der erste und einzige „Hidden Layer“ besitzt eine Größe von 12 Neuronen. Es gibt nur ein Neuron, welches den geschätzten Immobilienpreis ausgibt.

Bei einem Mini-Batch wird nur ein Teil der Trainingsdaten für das Training in einer Iteration verwendet. Für den Multi-Layer Perceptron Regressor beträgt die Größe jedes Mini-Batch 200. Die Trainingsdaten werden vor jedem Mini-Batch neu gemischt.

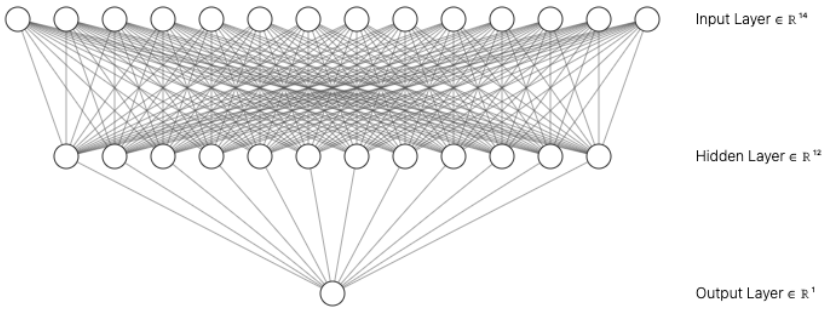


Abbildung 3.2: Visualisierung des künstlichen neuronalen Netzes vom Multi-Layer Perceptron Regressor

Die zeitliche Komplexität lässt sich als $O(n \cdot m \cdot h^k \cdot o \cdot i)$ darstellen. Dabei beschreibt n die Anzahl an Daten im Trainingsdatensatz, m die Anzahl an Features, k die Anzahl an Hidden Layers. h beschreibt die Anzahl der Neuronen in jedem Hidden Layer, o die Anzahl der Ausgabeneuronen und i die Anzahl an Iterationen. Für den, in dieser Arbeit verwendeten, Multi-Layer Perceptron Regressor beträgt die zeitliche Komplexität $O(93767 \cdot 14 \cdot 12^1 \cdot 1 \cdot 9133) = 143\,870\,833\,848$.

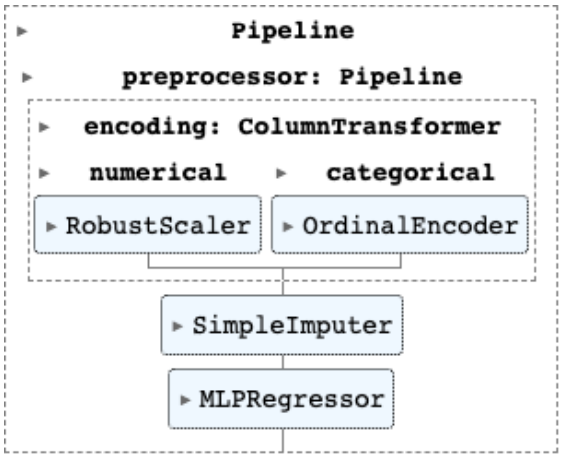


Abbildung 3.3: Pipeline des Multi-Layer Perceptron Regressor

In der Abbildung 3.3 ist die Pipeline des Multi-Layer Perceptron Regressor (MLPRegressor) zu sehen. Diese stellt die technische Implementierung des Multi-Layer Perceptron Regressor dar. Die gesamte Datenaufbereitung ist identisch zu der Pipeline des eXtreme Gradient Boosting Regression Tree aus Abschnitt 3.3.1. Nach der Datenaufbereitung werden die bereinigten Daten in den *MLPRegressor* geleitet.

Wie beim eXtreme Gradient Boosting Regression Tree aus Abschnitt 3.3.1 wird Early Stopping eingesetzt, um Overfitting zu verhindern. Dazu werden 10 % des Trainingsdatensatzes als Validierungsdatensatz verwendet. Der Validierungsdatensatz wird nicht für das Training des Modells genutzt. Es wird nach jeder Iteration überprüft, ob sich die Leistung des Modells im Vergleich zu 10 Iterationen zuvor verbessert. Sofern dies der Fall ist, wird das Training fortgesetzt. Andernfalls wird das Training beendet. Der Fehler für die Überprüfung wird mit der mittleren quadratischen Abweichung berechnet.

3.4 Multiple lineare Regression

Die lineare Regression beschreibt die Modellierung einer linearen Funktion, die den Einfluss einer oder mehrerer Variablen auf die Zielvariable abbilden soll. Da es meist keine lineare Funktion gibt, die die Zielvariablen genau abbildet, wird eine Annäherung durchgeführt. Das Ziel ist es eine Funktion zu finden, bei der die Summe aller Fehler zum Quadrat am niedrigsten ist. Als Fehler wird die Differenz zwischen der Funktion und einem Messpunkt bezeichnet. Dabei kann die lineare Regression in die *einfache* und *multiple* lineare Regression unterteilt werden. [12]

Die Entstehungsgeschichte der einfachen linearen Regression ist bis in das 19. Jahrhundert zurückzuführen. Bei der einfachen linearen Regression wird nur eine einzige Einflussgröße auf den hervorzusagenden Wert berücksichtigt. [35] [12]

Ein Beispiel für eine einfache lineare Regression wird in Abbildung 3.4 visualisiert. Dazu wird anhand der Wohnfläche eine Funktion berechnet, die den Verkaufspreis der Immobilien am besten darstellt. Diese wird in Rot angezeigt. Die hellblauen Punkte stellen die einzelnen Wertepaare aus Wohnfläche und Verkaufspreis dar. Dabei ist zu erkennen, dass eine lineare Regression anhand eines Parameters nicht die Komplexität einer Immobilienbewertung abbilden kann. Dies ist im Expertenwissen in Abschnitt 2.1.1 niedergelegt. Tendenziell nimmt zwar der Verkaufspreis beim Anstieg der Wohnfläche zu, aber die Punktwolke legt dar, dass dies eine zu starke Vereinfachung ist.

Die mathematische Formel für die einfache lineare Regression lautet $y_i = \beta_0 + \beta_1 x_i + u_i$. Für eine Berechnung sind die Messwerte $(x_1, y_1), \ldots, (x_i, y_i)$ gegeben. Die Zählung der Wertepaare wird durch $i = 1, \ldots, n$ beschrieben. Dabei ist y_i der Zielwert an der Position i. Die Unbekannten werden durch β_0 und β_1 dargestellt. Diese werden als Koeffizienten

Abbildung 3.4: Beispiel einer einfachen linearen Regression der Wohnfläche

der Funktion bezeichnet. Beschrieben wird β_0 meist als Achsenabschnitt und β_1 als Steigungsparameter. Die Störgröße u_i umfasst alle unsystematischen und zufälligen Fehler. [41] [12]

Da die einfache lineare Regression für komplexere Probleme nicht ausreichend ist, wurde die multiple lineare Regression entwickelt. Dabei werden mehrere Einflussgrößen mit in die Funktion einbezogen. [28] [7] Die multiple lineare Regression wird in dieser Arbeit mit den 14 Attributen aus Abschnitt 3.1.4 umgesetzt.

Bei der multiplen linearen Regression wird die mathematische Formel von der einfachen linearen Regression durch $+ \ldots + \beta_k x_k$ ergänzt:

$$y_i = \beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k + u_i$$

Wobei k die Anzahl der Attribute ist, die für das Modell verwendet werden. [41] [28] [7]

Es werden bei der linearen Regression die Koeffizienten berechnet, die den kleinsten Fehler zwischen der geschätzten linearen Funktion und den beobachteten Werten aufweisen.

Die Pipeline der multiplen linearen Regression wird in Abbildung 3.5 dargestellt. In dieser werden die Schritte aus der Datenaufbereitung aus Abschnitt 3.1 verwendet. Diese

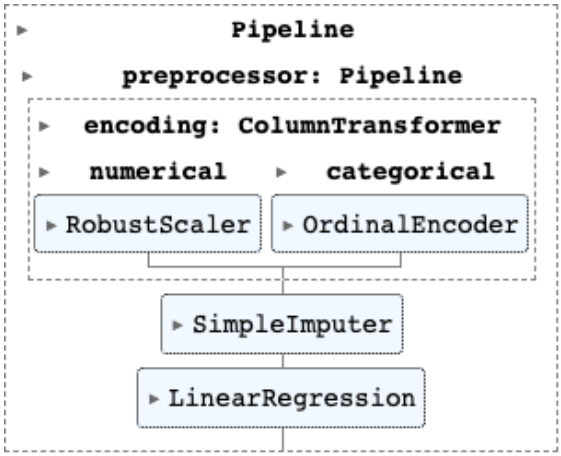


Abbildung 3.5: Pipeline der multiplen linearen Regression

werden identisch zu dem eXtreme Gradient Boosting Regression Tree und Multi-Layer Perceptron Regressor aus den Abschnitten 3.3.1 und 3.3.2 genutzt. Anschließend werden die Daten in die „LinearRegression“ geleitet. Diese stellt die multiple lineare Regression dar.

3.5 Export der Modelle

Damit die Modelle in anderen Komponenten, wie dem Backend aus Abschnitt 3.6.1, verwendet werden können, müssen diese exportiert werden. Dazu werden die Modelle als Python-Objekte mit der Bibliothek joblib serialisiert und in einer Datei gespeichert. [15] Diese Dateien enthalten alle Informationen, die nötig sind, um aus der Datei das Python-Objekt wiederherzustellen. Es lassen sich dadurch alle Funktionen verwenden, die für die Preisvorhersage benötigt werden.

Die exportierten Dateien sind verhältnismäßig zum verwendeten Datensatz klein. Für den eXtreme Gradient Boosting Regression Tree aus Abschnitt 3.3.1 beträgt die Dateigröße ca. 3,7 Megabyte, beim Multi-Layer Perceptron Regressor aus Abschnitt 3.3.2 ca. 712 Kilobyte und bei der multiplen linearen Regression aus Abschnitt 3.4 beträgt die Dateigröße nur ca. 302 Kilobyte.

3.6 Praktische Anwendung

In der praktischen Anwendung wird das Ziel verfolgt, dass ein Nutzer über eine Benutzeroberfläche die Preise von Immobilien in Deutschland schätzen kann. Dazu wird die praktische Anwendung in zwei Bereiche unterteilt. Zunächst wird ein Backend umgesetzt, welches die zuvor exportierten Modelle aus einer Datei lädt und über eine Schnittstelle die Schätzung von Immobilienpreisen anbietet. Dieses Backend wird dann von einer Benutzeroberfläche angefragt. Die Benutzeroberfläche fragt die Eingaben für die Modelle ab und zeigt die Ergebnisse der Preisschätzungen an. Zuletzt werden Optionen für das Hosting kurz dargestellt.

3.6.1 Backend

Ziel des Backend ist es die Preisschätzung der Modelle über eine Application Programming Interface (API) in Form einer Representational State Transfer (REST)-Schnittstelle zur Verfügung zu stellen. Dazu sollen Hypertext Transfer Protocol (HTTP) Anfragen mit den Attributen zu der Immobilie angenommen werden und die vorhergesagten Preise der Modelle als HTTP Antwort zurückgegeben werden.

Der Server, der das Backend dieser Arbeit darstellt, wird in Python programmiert. Dazu wird das Framework FastAPI verwendet. FastAPI erlaubt es, mit verhältnismäßig wenig Code, performante Webserver zu entwickeln. [9] Die Entscheidung für die Entwicklung in Python ist damit zu begründen, dass die Modelle in Python entwickelt wurden und so ausgeführt werden können.

Beim Start des Backend werden die Modelle aus Abschnitt 3.5 geladen. Damit sind die bereits trainierten scikit-learn [33] Pipelines in dem Backend verfügbar. Es können folglich die Preisschätzungen in dem Backend durch Aufrufe der entsprechenden Funktionen auf den Modellen durchgeführt werden.

Zentral wird der HTTP Endpunkt */predict* öffentlich unter der Methode *POST* zur Verfügung gestellt. An diesen müssen die Attribute aus dem Abschnitt 3.1.4 übergeben werden. Dazu werden die Attribute als JavaScript Object Notation (JSON) String kodiert. Die Attribute sind jeweils als optional gekennzeichnet, da nicht zu jeder Immobilie alle Informationen zur Verfügung stehen. Es werden die Attribute in ein für die Modelle nutzbares Format transformiert. Dann wird für jedes Modell mithilfe der *predict()*-Funktion von scikit-learn [33] ein Preis für die Immobilie geschätzt. Alle drei Schätzungen werden in

einem JSON String als HTTP Antwort zurückgegeben. Ergänzend wird die Dauer der Preisberechnung für jedes Modell angegeben.

Für die Bereinigung der Eingaben, die sogenannte *Input Sanitization*, wird eine Python Bibliothek verwendet. Diese stellt sicher, dass die richtigen Datentypen für die jeweiligen Attribute verwendet werden. Es wird dabei nicht geprüft, ob die Eingaben plausibel sind, da dies eine prototypische Umsetzung ist.

Zusätzlich wird ein HTTP Endpunkt */health* öffentlich unter der Methode *GET* zur Verfügung gestellt. Dieser gibt nur den statischen JSON String *{"success": true}* zurück. In der Benutzeroberfläche in Abschnitt 3.6.2 wird dieser Endpunkt verwendet, um zu prüfen, ob das Backend gestartet und bereit für Anfragen ist.

3.6.2 Benutzeroberfläche

Damit die Vorhersagen mittels des Backend für Nutzer einfach und grafisch dargestellt werden können, wird eine Benutzeroberfläche entwickelt. Diese soll im Webbrowser ausgeführt werden können, damit der Nutzer keine zusätzliche Software installieren muss.

Für die Benutzeroberfläche wird in dieser Arbeit React verwendet. React ist eine JavaScript Bibliothek zur Erstellung von Benutzeroberflächen. [29]

Dazu wird das React Framework NextJS mit TypeScript genutzt. NextJS bietet einige nützliche Funktionen, die React erweitern und die Entwicklererfahrung verbessern. In dieser Arbeit wird ein statischer Export durchgeführt, da keine dynamischen Inhalte der Website auf einem Server erstellt werden müssen. Die gesamten Dateien, bestehend aus Hypertext Markup Language (HTML)-, Cascading Style Sheets (CSS)- und JavaScript-Dateien, können somit einmalig bei Änderungen des Codes exportiert werden. Dazu wird der von NextJS mitgelieferte Befehl *next export* verwendet. Dabei wird zunächst eine Überprüfung der Typen in dem Code mit TypeScript durchgeführt. Anschließend werden die benötigten Dateien gesammelt, optimiert und gespeichert. [26]

Zur schnelleren Entwicklung wird zusätzlich das CSS Framework TailwindCSS für das Design der Website eingesetzt. TailwindCSS bietet viele CSS Klassen, die in dem React Code genutzt werden können. Während des Exports durch NextJS werden die genutzten CSS Klassen durch den TailwindCSS Compiler zu reinem CSS kompiliert. Die entstandene CSS Datei wird anschließend mit exportiert. Für den Endnutzer ist die Verwendung von TailwindCSS nicht erkennbar. [36]

Eine Bildschirmaufnahme der Benutzeroberfläche ist in Abbildung 3.6 zu sehen. Dort ist zu erkennen, dass das Formular in zwei logische horizontale Bereiche aufgeteilt ist. Somit ist die Trennung zwischen Eingaben und Ausgaben für den Nutzer besser ersichtlich.

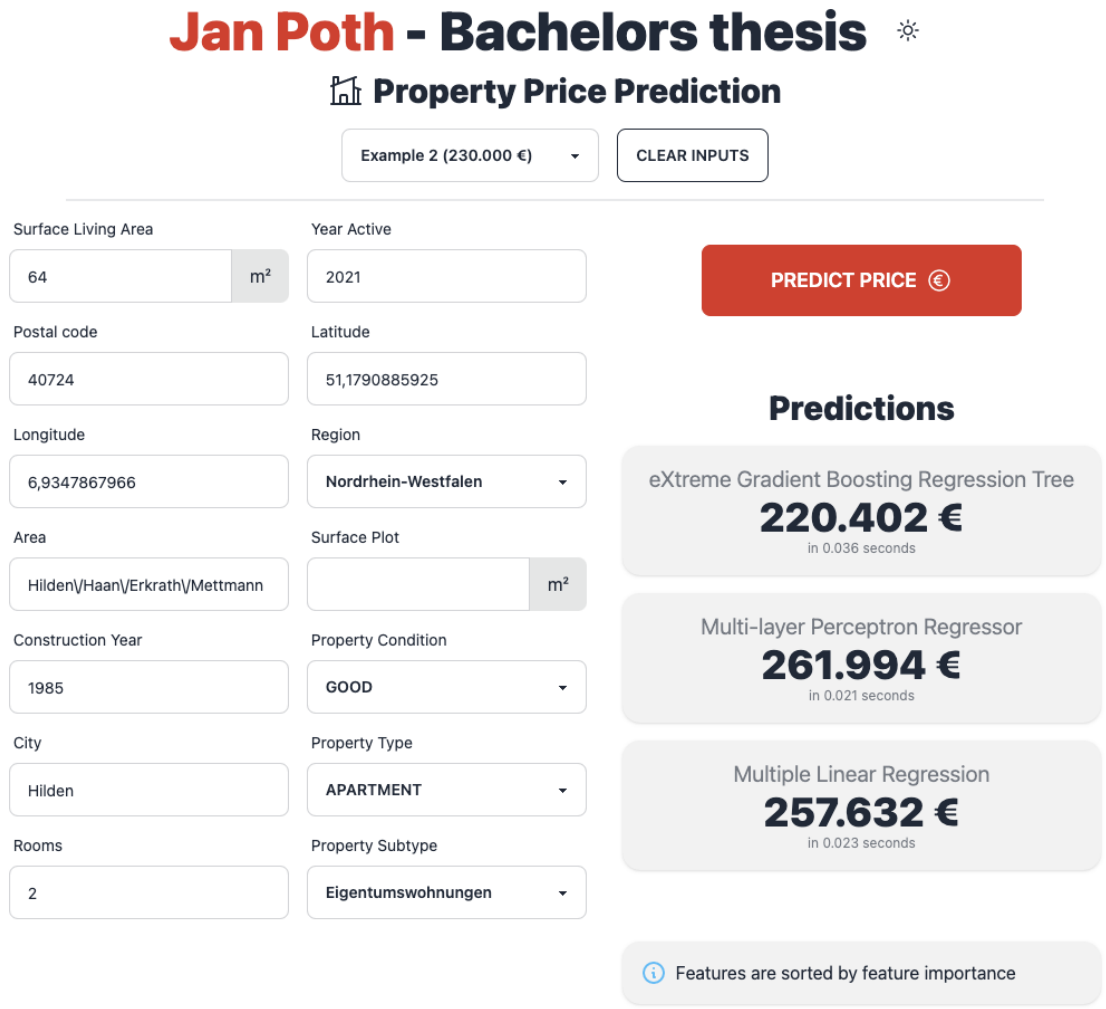


Abbildung 3.6: Bildschirmaufnahme der Benutzeroberfläche

Auf der linken Seite sind die Eingabefelder der Modelle zu sehen. Dabei ist zu erkennen, dass diese in unterschiedlichen Ausführungen vorhanden sind. So gibt es Eingaben für ganze Zahlen, wie das Baujahr oder die Anzahl an Zimmern. Zusätzlich gibt es Eingaben für die Wohnfläche und die Grundstücksfläche, welche in m^2 angegeben werden. Gleitkommazahlen, wie für den Längen- und Breitengrad, können in gesonderten Eingabefeldern eingetragen werden. Für Attribute, wie zum Beispiel den Immobilientyp,

Zustand der Immobilie oder die Region gibt es Auswahlfelder. Bei diesen lassen sich nur gültige Werte auswählen, die im Datensatz vorhanden sind.

Für die Bereinigung der Eingaben, die sogenannte *Input Sanitization*, wird in der Benutzeroberfläche die Funktionalität des Browsers verwendet. So werden die Eingabefelder mit dem jeweiligen HTML *type* versehen. Zu beachten ist, dass eine erneute Prüfung in dem Backend durchgeführt wird. Die Überprüfungen verbessern die Nutzererfahrung, da bereits in der Benutzeroberfläche falsche Eingaben reduziert werden.

Auf der rechten Seite ist oben ein roter Button sichtbar, der die Aufschrift „Predict Price“ (Preis vorhersagen) besitzt. Darunter werden dann die geschätzten Immobilienpreise für alle Modelle angezeigt. Zusätzlich wird die Zeit für die Berechnung der Preise angegeben.

Sobald der Nutzer auf diesen Button drückt, wird eine HTTP *POST* Anfrage an den Endpunkt */predict* des Backend aus Abschnitt 3.6.1 geschickt. Die Anfrage erhält alle Eingaben des Nutzers serialisiert in einen JSON-String. Als Antwort wird ein JSON-String mit den geschätzten Immobilienpreisen und den Zeiten für die Berechnung der Ergebnisse aller Modelle zurückgeschickt. Eine beispielhafte Antwort sieht wie folgt aus: *{ “xgb_gradient_boosting_regressor“: { “price“: 350151, “timing“: 0.035}, “mlp_regressor“: { “price“: 368582, “timing“: 0.029}, “linear_regression“: { “price“: 382168, “timing“: 0.024} }*

Oben unter dem Titel gibt es das Auswahlfeld, aus dem sich Beispieldaten aus dem Datensatz laden lassen. Diese Auswahl umfasst Daten des Testdatensatzes. Somit sind die Beispiele den Modellen nicht bekannt. Wird ein Beispiel geladen, so wird dies dort angezeigt mit dem Text „Example N (X)“. Hierbei gibt X den tatsächlichen Verkaufspreis an, der in dem Datensatz hinterlegt ist. N ist dabei die interne Nummerierung der Beispiele.

Über den Button „Clear Inputs“ lassen sich mit einem Klick alle Eingabefelder löschen. Dadurch müssen nicht alle Felder einzeln geleert werden und es ist einfacher mehrere Anfragen in Folge zu erstellen.

Die Auswahl und Reihenfolge der Eingabefelder sind dabei auf die Auswertung der Feature Importance in Abbildung 4.2 zurückzuführen.

Die Benutzeroberfläche stellt HTTP *GET* Anfragen an den Endpunkt */health* des Backend. Dadurch kann eine Warnung angezeigt werden, wenn das Backend nicht verfüg-

bar ist, bevor der Nutzer eine Anfrage abschickt. Diese Überprüfung findet beim Laden der Seite, bei Änderung der Netzwerkverbindung und beim Ändern des Browserfensters statt. Die Anfragen werden im Hintergrund durchgeführt. Sofern das Backend gestartet und verfügbar ist, wird dies durch eine Antwort mit *{"success": true}* bestätigt. Da die Überprüfungen im Hintergrund durchgeführt werden, wird dem Nutzer keine Meldung angezeigt.

Erweitert wurde die Benutzeroberfläche noch über eine wählbare Option des dunklen Modus. Dabei werden die Farben im Hintergrund dunkelgrau eingefärbt und die Texte hellgrau dargestellt. Die Akzentfarbe Rot bleibt dabei bestehen.

3.6.3 Hosting

Für das Hosting des Backend und der Benutzeroberfläche aus den Abschnitten 3.6.1 und 3.6.2 gibt es mehrere Möglichkeiten.

Lokal kann das Backend und die Benutzeroberfläche über Docker in Containern ausgeführt werden. Dazu wird für die zwei Systeme eine *Dockerfile* angelegt, die beschreibt, wie der Server gebaut werden soll, der das jeweilige System ausführt. Anschließend lassen sich die Systeme mit dem Befehl *docker compose up* gleichzeitig ausführen. [6]

Zusätzlich werden die entsprechenden Portfreigaben auf dem Rechner erstellt. Das Backend ist über den Port 8000 erreichbar. Der Port 3000 wird für die Benutzeroberfläche verwendet. Um das System lokal aufzurufen wird die Adresse `http://localhost:3000` in einem Webbrowser geöffnet.

Für den produktiven Betrieb der Benutzeroberfläche lässt sich mithilfe von NextJS eine statische Website exportieren. Somit werden nur HTML-, CSS- und JavaScript-Dateien erstellt. Diese können mit jedem gängigem Webserver ausgeliefert werden. Eine kostengünstige Variante ist das Hosting über ein Content Delivery Network (CDN). Dabei werden die statischen Dateien weltweit verteilt, sind so von Nutzern schneller abrufbar und haben im Durchschnitt verkürzte Ladezeiten im Vergleich zu einem zentralen Hosting. [27]

Das Backend könnte beispielsweise produktiv über eine „Serverless Function“ ausgeführt werden. Dabei ist kein Server dauerhaft im Betrieb, sondern es wird nur temporär ein Server gestartet, sobald das Backend angefragt wird. Dieser wird anschließend wieder beendet. Wird eine Funktion in kurzer Zeit regelmäßig aufgerufen, so kann diese zu

Optimierungszwecken zwischengespeichert bleiben, um die Ladezeit zu verkürzen. [13, vgl. S. 121-123] Ein Nachteil dabei könnten die langsameren Startzeiten sein, wenn die Funktion längere Zeit nicht aufgerufen wird. Dabei wären Startzeiten von ca. 2 Sekunden möglich. Das wäre für die Schätzung eines Immobilienpreises kein Problem, da folgende Anfragen dann wieder deutlich schneller wären. Die mögliche Ladezeit bezieht sich dabei nur auf das Backend und nicht auf die Benutzeroberfläche.

Eine weitere Option ist das Hosting des Backend mittels eines oder mehrerer Server, die je nach Last horizontal skaliert werden. Dafür müsste ein Load Balancer die Last zwischen den einzelnen Instanzen des Backend verteilen. Wird der Dienst momentan nur wenig verwendet, wird die Anzahl der Instanzen auf eins reduziert. Dadurch werden Kosten gespart, aber der Dienst bleibt zu jeder Zeit schnell aufrufbar. Grund hierfür ist, dass dauerhaft die Modelle auf einem Server geladen bleiben und dieser das Backend über die API zur Verfügung stellt. In diesem Szenario können die Docker Container aus der lokalen Umgebung genutzt werden. Dadurch ist die technische Umgebung bei der lokalen Entwicklung und bei der produktiven Ausführung identisch. Das kann in vielen Fällen die Entwicklung vereinfachen und Fehler vermeiden, die nur in der produktiven Anwendung auftreten. [38] [6]

4 Auswertung und Diskussion

In diesem Kapitel wird die Umsetzung aus Kapitel 3 ausgewertet und die Ergebnisse diskutiert. Dazu werden verschiedene Metriken eingeführt. Diese Metriken und weitere Auswertungen werden sowohl auf den Modellen des maschinellen Lernens als auch auf der multiplen linearen Regression angewendet. Zusätzlich wird ein Vergleich der drei Modelle durchgeführt. Anschließend wird die praktische Anwendung evaluiert. Dazu zählen die Benutzeroberfläche und das Backend. Es wird abschließend noch eine kurze Einführung in mögliche Anwendungsgebiete gegeben.

4.1 Metriken

Für eine objektive Auswertung werden in dieser Arbeit mehrere Metriken verwendet. Diese ermöglichen es die Leistungen der Modelle zu evaluieren.

Feature Importance Feature Importance gibt an, wie wichtig ein Attribut für die Vorhersagen eines Modells durchschnittlich ist. Sie wird meist zwischen $0,0$ und $1,0$ angegeben.

Mean absolute error Zentral ist die Nutzung des *mean absolute error* (mittleren absoluten Fehlers). Damit lässt sich bestimmen, wie stark die durchschnittliche Abweichung des tatsächlichen und des vorhergesagten Verkaufspreises ist. Diese wird in Euro angegeben. Der optimale Wert liegt bei $0,0$. Negative Werte sind nicht möglich.

Median absolute error Zusätzlich wird der *median absolute error* (Median der absoluten Fehler) verwendet. Dieser wird in Euro angegeben. Der Wert kann nicht negativ sein und besitzt den bestmöglichen Wert von $0,0$.

R^2 Score Der R^2 Score beschreibt den Determinationskoeffizienten (coefficient of determination). Dieser gilt als ein Messwert für die Leistung bei der Regression. Der bestmögliche Wert ist $1,0$. Der Wert kann negativ sein.

4.2 Maschinelles Lernen

4.2.1 eXtreme Gradient Boosting Regression Tree

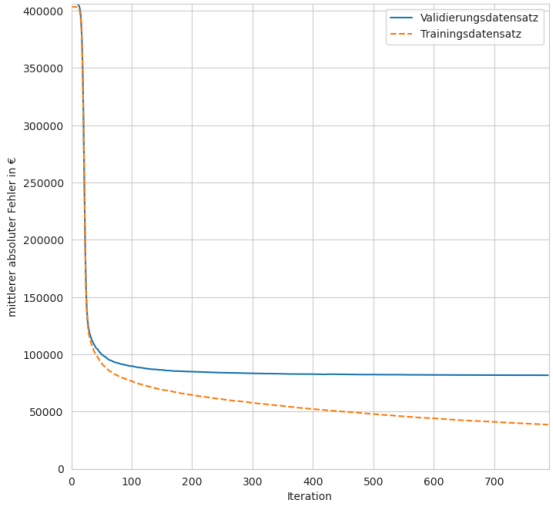


Abbildung 4.1: Mittlerer absoluter Fehler des eXtreme Gradient Boosting Regression Tree im Verlauf des Trainings

Wie beim Trainingsverlauf vom eXtreme Gradient Boosting Regression Tree in Abbildung 4.1 zu erkennen ist, verbessert sich das Training bereits innerhalb der ersten 50 Iterationen signifikant. Dabei wird der mittlere absolute Fehler (mean absolute error) beim Validierungsdatensatz von anfänglich >400 000 € auf knapp 125 000 € reduziert. Im weiteren Verlauf bis hin zu Iteration 789 ist zu erkennen, dass sich der Fehler des Validierungsdatensatz weiter auf 81 780 € vermindert. Der Fehler beim Trainingsdatensatz verringert sich final auf einen Wert von 38 692 €. Beim Training wird Early Stopping auf Basis des Validierungsdatensatzes verwendet, damit das Training stoppt, sobald sich die Leistung nicht mehr ausreichend verbessert.

Die finalen Metriken nach dem Training des eXtreme Gradient Boosting Regression Tree sind folgende:

- Mittlerer absoluter Fehler (mean absolute error): 81 773 €
- Median der absoluten Fehler (median absolute error): 45 527 €
- R^2 Score: 0, 781
- Trainingszeit: 62 Sekunden
- Iteration: 789

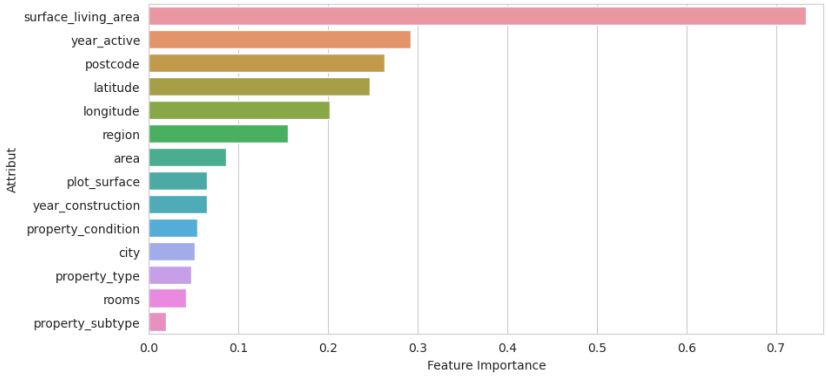


Abbildung 4.2: Feature Importances des eXtreme Gradient Boosting Regression Tree für jedes der 14 verwendeten Attribute

Bei der Auswertung der Feature Importance für jedes der 14 Attribute in Abbildung 4.2 ist ersichtlich, dass einige Attribute besonders relevant für die Bestimmung des Verkaufspreises sind. Mit großem Abstand das wichtigste Attribut ist dabei die Wohnfläche mit 0, 733. Als zweitwichtigstes Attribut ist das Jahr des Verkaufs mit 0, 292, als drittwichtigstes die Postleitzahl mit 0, 263 und als viertwichtigstes der Breitengrad mit ca. 0, 297 zu entnehmen. Andere Attribute, wie der Längengrad mit 0, 202 oder die Region mit 0, 156 sind für die Preisbestimmung ebenfalls relevant.

Im Vergleich der Thesen aus dem Expertenwissen aus Abschnitt 2.1.1 und den Ergebnissen der Feature Importance vom eXtreme Gradient Boosting Regression Tree lassen sich viele Übereinstimmungen feststellen.

Die Wohnfläche wird korrekt als der wichtigste Indikator für die Schätzung des Verkaufspreises festgestellt. Es geht außerdem hervor, dass die Lage sowohl für das Modell als auch für Makler einen entscheidenden Faktor darstellt.

Die Relevanz des Jahres, indem die Immobilie verkauft wurde, ist für das Modell von hoher Bedeutung. Das lässt sich damit begründen, dass sich die Immobilienpreise in den

letzten 25 Jahren stark verändert haben. Ob eine Immobilie in dem Jahr 1995 oder 2019 verkauft worden ist, ist für das Modell relevant, da sich sowohl der Durchschnittspreis (siehe Abbildung 2.5b) als auch weitere Faktoren über die Jahre verändert haben. Im Gegensatz zu dem Modell müssen Makler meist nur den Preis für das aktuelle Jahr berechnen.

Auch die weiteren Attribute mit der größten Feature Importance, die vom Modell festgestellt wurden, stimmen mit denen von Maklern als wichtig erachteten Attributen überein.

Das Modell besaß im Vorhinein jedoch keine Kenntnis über das Expertenwissen und erlernte diese wichtigen Faktoren eigenständig. Es lässt sich vermuten, dass das Modell selbstständig anhand der Daten sinnvolle Zusammenhänge für eine Preisbewertung erkennen konnte.

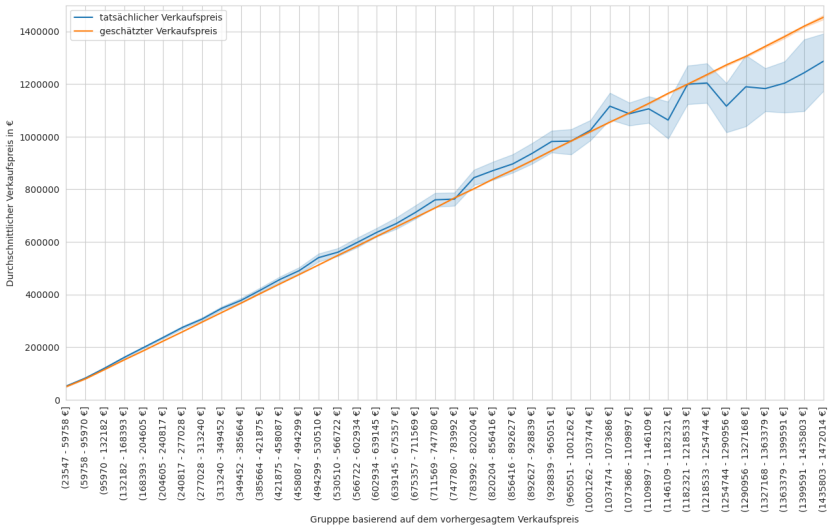


Abbildung 4.3: Vergleich der vorhergesagten Immobilienpreise des eXtreme Gradient Boosting Regression Tree mit den tatsächlichen Immobilienpreisen

Das spiegelt sich in der Abbildung 4.3 wider. In der Darstellung werden die Ergebnisse des Testdatensatzes basierend auf den geschätzten Verkaufspreisen in Gruppen unterteilt. Anschließend werden die geschätzten Verkaufspreise und die tatsächlichen Verkaufspreise für jede Gruppe visualisiert. Die Linie der geschätzten Verkaufspreise ist dabei linear, da die Verkaufspreise die Grundlage für die Gruppierung sind und als Vergleichswert dienen. Die hellblau hinterlegten Bereiche stellen die Abweichungen der tatsächlichen Verkaufspreise dar.

Beobachtet werden kann, dass für Immobilien mit einem Preis bis ca. 750 000 € das Modell sehr akkurate Preisschätzungen durchführt. Von ca. 750 000 € bis ca. 1 100 000 € unterschätzt das Modell den Preis im Durchschnitt um ca. 75 000 €. Über den Preis von ca. 1 100 000 € hinaus überschätzt das Modell den Preis im Durchschnitt um ca. 100 000 €.

Insgesamt lässt sich sagen, dass der eXtreme Gradient Boosting Regression Tree mit nur 14 Attributen (siehe Abschnitt 3.1.4) akkurate Ergebnisse erzielen konnte. Das Modell führt ungenauere Einschätzung durch, wenn der Verkaufspreis der Immobilie über 1 100 000 € liegt. Dies könnte darauf zurückzuführen sein, dass für höherpreisige Immobilien die Schätzung schwieriger wird, wie im Expertenwissen in Abschnitt 2.1.1 aufgeführt.

4.2.2 Multi-Layer Perceptron Regressor

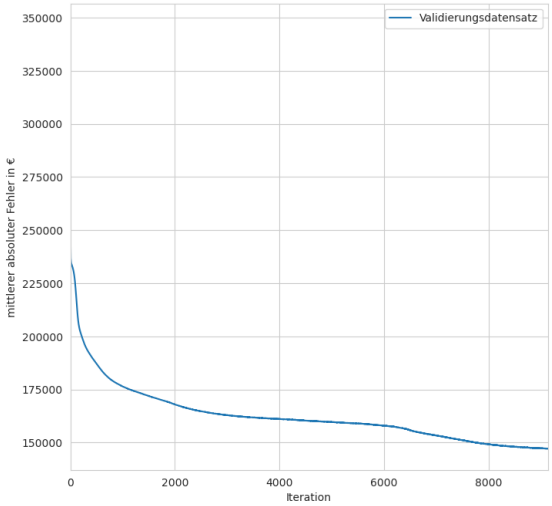


Abbildung 4.4: Mittlerer absoluter Fehler des Multi-Layer Perceptron Regressor im Verlauf des Trainings

Der Trainingsverlauf des Multi-Layer Perceptron Regressor ist in Abbildung 4.4 dargestellt. Die Iterationen bestehen jeweils aus einem Mini-Batch der Größe 200. Innerhalb der ersten 500 Iterationen verringert sich der mittlere absolute Fehler (mean absolute error) auf ca. 190 000 €. Anschließend ist in den Iterationen 500 bis ca. 3 000 eine nahezu lineare Reduktion des Fehlers beim Validierungsdatensatz zu beobachten. Bis ca. 6 000 Iterationen verläuft die Reduktion etwas langsamer. Ab diesem Zeitpunkt reduziert sich

der mittlere absolute Fehler erneut schneller bis zu ca. 8 000 Iterationen. Danach flacht das Training erneut ab und wird nach 9 133 Iterationen bei 149 335 € durch das Early Stopping beendet, damit Overfitting verhindert wird.

Die finalen Metriken nach dem Training des Multi-Layer Perceptron Regressor sind folgende:

- Mittlerer absoluter Fehler (mean absolute error): 149 335 €
- Median der absoluten Fehler (median absolute error): 107 552 €
- R^2 Score: 0, 466
- Trainingszeit: 2 767 Sekunden
- Iteration: 9 133

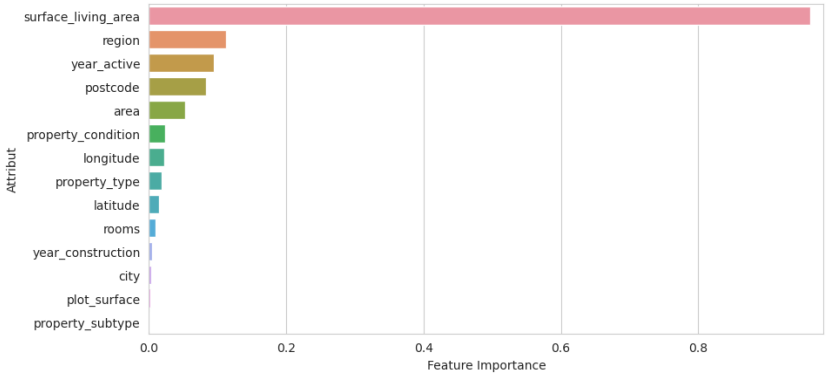


Abbildung 4.5: Feature Importances des Multi-Layer Perceptron Regressor für jedes der 14 verwendeten Attribute

Bei der Feature Importance des Multi-Layer Perceptron Regressor lässt sich ableiten, dass für das Modell die Wohnfläche der Immobilie ausschlaggebend für die Preisschätzung ist. Die Wohnfläche ist mit 0, 541 das Attribut mit der höchsten Feature Importance. Nach größerem Abstand kommen die Attribute zu der Lage der Immobilie. Die Region besitzt den Wert 0, 09, die Area 0, 058 und die Postleitzahl 0, 055. Das Verkaufsjahr verfügt über eine Feature Importance von (0, 08). Der Abbildung ist zu entnehmen, dass dem Zustand der Immobilie eine Relevanz von 0, 031 zugeschrieben wird.

Mit Bezug zum Expertenwissen in Abschnitt 2.1.1 lässt sich erkennen, dass die Wohnfläche der entscheidende Faktor ist. Die Feature Importance der Lage wurde auch als hoch

eingeschätzt. Da das Modell diese Faktoren identifizierte, lässt sich darauf schließen, dass das Modell aus den Daten lernt.

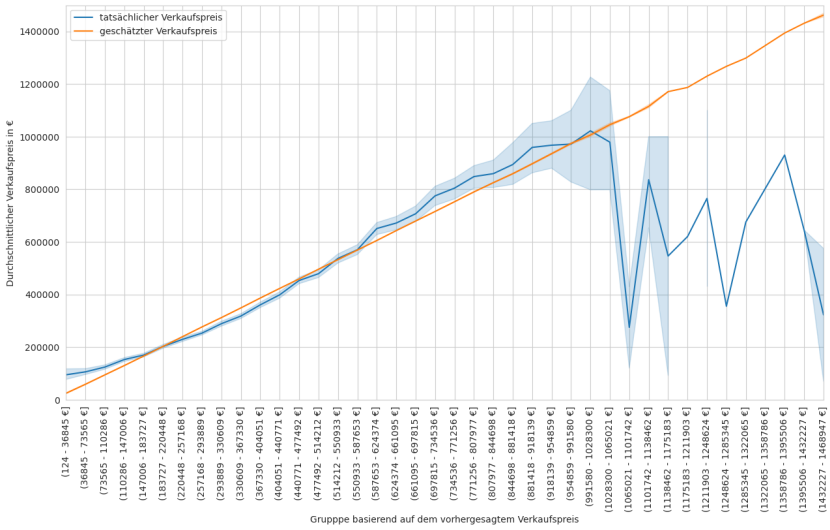


Abbildung 4.6: Vergleich der vorhergesagten Immobilienpreise des Multi-Layer Perceptron Regressor mit den tatsächlichen Immobilienpreisen

In der Abbildung 4.6 werden die Ergebnisse des Testdatensatzes basierend auf den geschätzten Verkaufspreisen in Gruppen unterteilt. Anschließend werden die geschätzten Verkaufspreise und die tatsächlichen Verkaufspreise für jede Gruppe visualisiert. Die Linie der geschätzten Verkaufspreise ist dabei linear, da die Verkaufspreise die Grundlage für die Gruppierung sind und als Vergleichswert dienen. Die hellblau hinterlegten Bereiche der tatsächlichen Verkaufspreise stellt die Abweichungen dar.

Für Immobilien mit einem Preis bis ca. 250 000 € unterschätzt das Modell den Preis leicht. Von ca. 250 000 € bis 550 000 € sind die Ergebnisse nah an dem tatsächlichen Verkaufspreis. Ab ca. 550 000 € überschätzt das Modell die Verkaufspreise tendenziell um ca. 75 000 €. Zwar gibt es um den Wert von 825 000 € einige zutreffende Ergebnisse, jedoch beträgt die Abweichung bei höheren Preisen als 1 000 000 € größtenteils bis zu 300 000 €.

Über die gesamte Preisspanne zeigt sich, dass in einigen Bereichen zutreffende Ergebnisse erzielt werden konnten, jedoch in hochpreisigen Bereichen deutliche Schwächen zu verzeichnen sind.

4.3 Multiple lineare Regression

Bei der multiplen linearen Regression gibt es kein Modell, welches im Laufe des Trainings optimiert wird. Das Training beschreibt die Suche nach den optimalen Koeffizienten, um eine möglichst genaue lineare Annäherung an die tatsächlichen Messwerte zu erzielen.

Die finalen Metriken nach dem Training der multiplen linearen Regression sind folgende:

- Mittlerer absoluter Fehler (mean absolute error): 169 311 €
- Median der absoluten Fehler (median absolute error): 127 687 €
- R^2 Score: $0,317$
- Trainingszeit: 4 Sekunden

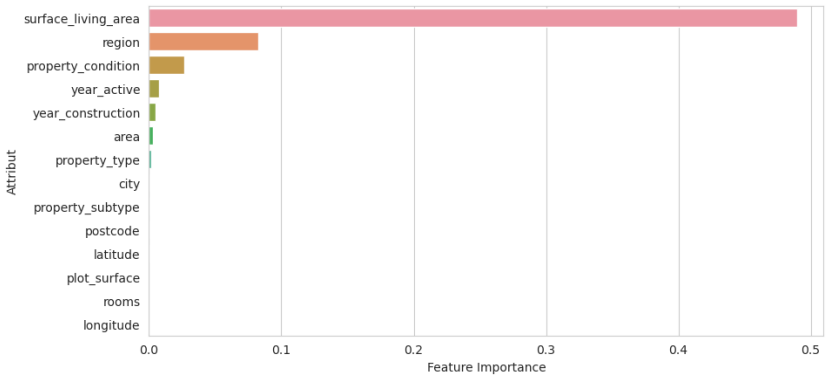


Abbildung 4.7: Feature Importances der multiplen linearen Regression für jedes der 14 verwendeten Attribute

In Abbildung 4.7 wird die Feature Importance für jedes der 14 Attribute dargestellt. Dabei erzielt die Wohnfläche mit $0,49$ die mit Abstand höchste Feature Importance. Die Region wird mit einem Wert von $0,083$ angegeben und der Immobilienzustand mit $0,027$. Die anderen Attribute haben alle eine Feature Importance von unter $0,01$.

Im Vergleich zum Expertenwissen in Abschnitt 2.1.1 haben viele Attribute für das Modell weniger Relevanz. Es wird nicht hinreichend die Komplexität einer Immobilienbewertung erfasst.

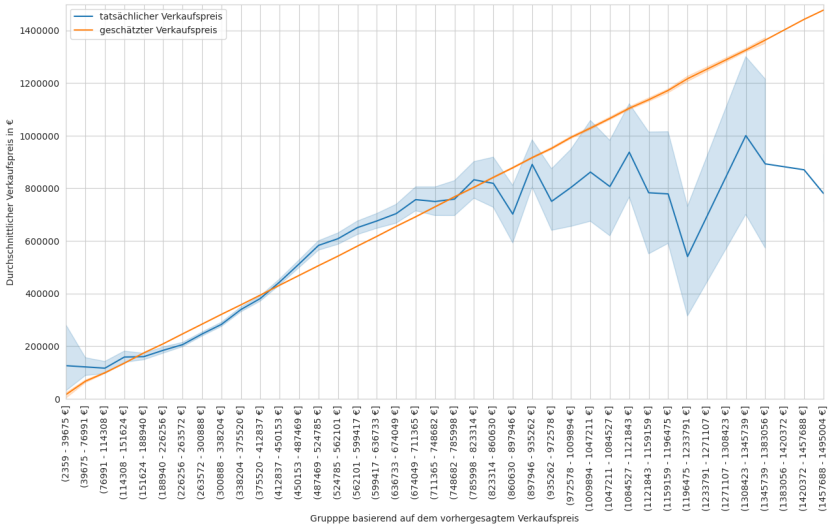


Abbildung 4.8: Vergleich der vorhergesagten Immobilienpreise der multiplen linearen Regression mit den tatsächlichen Immobilienpreisen

Die Ergebnisse des Testdatensatzes werden in Abbildung 4.8 basierend auf den geschätzten Verkaufspreisen in Gruppen unterteilt. Diese Gruppen und die tatsächlichen Verkaufspreise werden anschließend visualisiert. Dabei ist erkennbar, dass das Modell der multiplen linearen Regression im Bereich von 85 000 € bis 850 000 € eine durchschnittliche Abweichung von 100 000 € besitzt. Ab 850 000 € überschätzt das Modell die Preise meistens und die durchschnittliche Abweichung liegt teilweise bei über 400 000 €.

Die multiple lineare Regression erzielt ungenaue Ergebnisse für eine Vorhersage der Immobilienpreise im Bereich ab 850 000 €. Auch im niedrigen Preisbereichen unter 85 000 € sind die Schätzungen ungenau.

4.4 Vergleich der Modelle

In Tabelle 4.1 werden der eXtreme Gradient Boosting Regression Tree (XGBRegressor), der Multi-Layer Perceptron Regressor (MLPRegressor) und die multiple lineare Regression (MLR) miteinander verglichen. Dazu werden verschiedene Metriken genutzt.

Betrachtet man den mittleren absoluten Fehler, so ist zu sehen, dass der XGBRegressor durchschnittlich um 67 582 € besser schätzt als der MLPRegressor. Dies entspricht einem Unterschied von 82,6 %. Beim Vergleich des Median der absoluten Fehler ist sogar ein

Metrik	**XGBRegressor**	**MLPRegressor**	**MLR**
Mittlerer absoluter Fehler	81 773 €	149 335 €	169 311 €
Median der absoluten Fehler	45 527 €	107 552 €	127 687 €
R^2 Score	0,781	0,466	0,317
Trainingszeit	62 Sekunden	2 767 Sekunden	4 Sekunden
Iterationen	789	9 133	×

Tabelle 4.1: Vergleich mehrerer Metriken zwischen eXtreme Gradient Boosting Regression Tree (XGBRegressor), Multi-Layer Perceptron Regressor (MLPRegressor) und der multiplen linearen Regression (MLR)

Unterschied von 136,2 % zu verzeichnen. Im Vergleich zur MLR sind die Preisschätzungen des XGBRegressor sogar um durchschnittlich 87 538 € besser. Das entspricht einem Unterschied von 207,05 %. Insbesondere für hochpreisige Immobilien ab 850 000 € wurde gezeigt, dass der XGBRegressor deutlich genauere Vorhersagen treffen kann.

Zudem ist der XGBRegressor dem MLPRegressor bei der zeitlichen Betrachtung überlegen. Die Trainingszeit des XGBRegressor betrug 62 Sekunden. Im Vergleich dazu brauchte der MLPRegressor 2 767 Sekunden. Somit ist der XGBRegressor im Training 4 363 % schneller. Durch die schnellere Trainingszeit ist sowohl die automatisierte Parametersuche (siehe Abschnitt 3.2) als auch die manuelle Optimierung deutlich schneller. Beim MLPRegressor ist dies mit deutlich längeren Wartezeiten von 46 Minuten verbunden. Deshalb wurde beim MLPRegressor meist ein Modell mit weniger Iterationen während der Optimierungsphase genutzt. Die multiple lineare Regression ist dabei mit 4 Sekunden erheblich schneller. Diese ist somit 1550 % schneller als der XGBRegressor.

Der R^2 Score ist ebenfalls beim XGBRegressor mit einem Wert von 0,781 gegenüber dem MLPRegressor um 0,315 erhöht. Im Vergleich zur MLR ist sogar eine Erhöhung von 0,464 beim XGBRegressor festzustellen.

In Abbildung 4.9 kann der Trainingsverlauf beider Modelle des maschinellen Lernens im direkten Vergleich nachvollzogen werden. Dabei wird auf der x-Achse die Anzahl der Iterationen und auf der y-Achse der mittlere absolute Fehler in € (mean absolute error 4.1) abgebildet. Die Anzahl der Iterationen ist aufgrund von unterschiedlichen Trainingsarten nicht direkt miteinander zu vergleichen, bietet allerdings einen Richtwert. In Blau ist der Verlauf des eXtreme Gradient Boosting Regression Tree (XGBRegressor) und in Gelb der des Multi-Layer Perceptron Regressor (MLPRegressor) zu sehen.

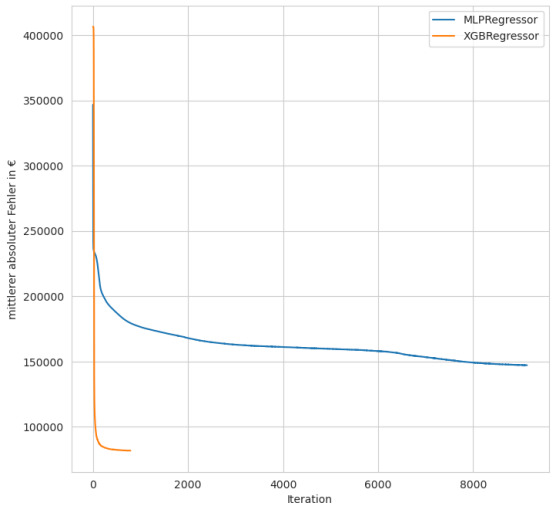


Abbildung 4.9: Vergleich des mittleren absoluten Fehlers zwischen XGBRegressor und MLPRegressor über den Verlauf des Trainings

Es ist zu erkennen, dass der XGBRegressor bereits nach ca. 100 Iterationen auf ein Niveau von unter 100 000 € gelangt. Nach dem schnellen Abstieg verläuft der XGBRegressor annähernd linear bis zu 81 773 €. Dann beendet das Early Stopping das Training.

Beim MLPRegressor sinkt der mittlere absolute Fehler anfangs schnell auf ca. 175 000 € bei 750 Iterationen. Danach verläuft das Training in einem Zeitraum von 8 000 Iterationen annähernd linear. Der große Unterschied besteht allerdings darin, dass der MLPRegressor einen deutlich höheren Fehler sowohl nach der anfänglichen schnellen Phase als auch nach dem gesamten Trainingsverlauf besitzt. Nach 9 133 Iterationen wird dieses Training durch das Early Stopping beendet.

Da die multiple lineare Regression kein iterativer Lernalgorithmus ist, ist eine Darstellung des Trainingsverlaufs nicht möglich.

Beim Vergleich der Feature Importance lässt sich erkennen, dass beim eXtreme Gradient Boosting Regression Tree mehr Attribute eine höhere Gewichtung haben. Beim Multi-Layer Perceptron Regressor und insbesondere bei der multiplen linearen Regression haben viele Attribute eine niedrige Gewichtung. Mit Bezug auf das Expertenwissen in Abschnitt 2.1.1 ist zu folgern, dass der eXtreme Gradient Boosting Regression Tree die zutreffendste Verteilung der Attribute bei der Feature Importance besitzt.

Insgesamt ist in Anbetracht aller Metriken der eXtreme Gradient Boosting Regression Tree dem Multi-Layer Perceptron Regressor und der multiplen linearen Regression über-

legen. Mit Ausnahme der Trainingszeit konnte der eXtreme Gradient Boosting Regression Tree bei allen Metriken die besten Ergebnisse erzielen. Der Multi-Layer Perceptron Regressor konnte ebenfalls bessere Ergebnisse als die multiple lineare Regression erreichen.

Aus den Ergebnissen der multiplen linearen Regression lässt sich deuten, dass diese die Komplexität der Immobilienbewertung nicht hinreichend abbilden kann.

Die Auswertung ergab insgesamt, dass die Modelle des maschinellen Lernens der multiplen linearen Regression bei Immobilienpreisschätzungen überlegen sind. Insbesondere der verwendete eXtreme Gradient Boosting Regression Tree konnte im Vergleich zu der multiplen linearen Regression über die gesamte Preisspanne bessere Ergebnisse erzielen.

4.5 Praktische Anwendung

4.5.1 Backend

Um die Praktikabilität des Backend in Form eines API Server beurteilen zu können wird dieser mit einem Stresstest geprüft. Der Stresstest wird lokal auf einem MacBook Pro 2020 (Intel) durchgeführt. [24] Für den Stresstest wird das Open Source Framework Locust verwendet. [22]

Wie in Abbildung A.1 zu erkennen, beträgt die durchschnittliche Antwortzeit ca. 600 Millisekunden. Diese wird von dem Absenden der Anfrage bis hin zur Antwort gemessen. Zu beachten ist, dass in dieser Zeit die Preisschätzungen für alle Modelle aus Abschnitt 3.3.1, 3.3.2 und 3.4 durchgeführt werden, welche insgesamt ca. 100 Millisekunden benötigen.

Dabei ist in Abbildung A.2 dargestellt, dass ca. 15 Anfragen pro Sekunde ohne Probleme bearbeitet werden können. Falls mehr als 15 Immobilienbewertungen pro Sekunde durchgeführt werden müssten, könnte der Server sowohl vertikal als auch horizontal skaliert werden. Dadurch ist es möglich das Backend für den Anwendungsfall angemessen zu skalieren.

Für eine interne Verwendung in einem produktiven Umfeld müssten die Benutzer authentifiziert und autorisiert werden.

4.5.2 Benutzeroberfläche

Die Benutzeroberfläche wurde mit modernen Technologien umgesetzt und ist leicht an neue Anforderungen anpassbar. Bei internen Betrachtungen von potenziellen Nutzern und Mitarbeitern wurde die visuelle Gestaltung gelobt. Die klare Trennung zwischen Eingaben und Ausgaben ist positiv aufgefasst worden.

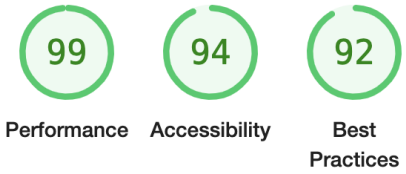


Abbildung 4.10: Lighthouse Report der Benutzeroberfläche

Mithilfe eines automatisierten „Lighthouse Report“ wurde die Performance und Barrierefreiheit analysiert. Die Ergebnisse in Abbildung 4.10 wurden mit einem Google Chrome Browser in der Version 105 durchgeführt. In allen Kategorien des Tests sind sehr gute Ergebnisse erzielt worden. Alle Eingaben sind ausschließend über die Tastatur möglich. Für Screenreader sind die erforderlichen Hinweise korrekt gesetzt. Nutzer mit Seheinschränkungen können sich folglich den Inhalt der Website vorlesen lassen. [20]

In der Benutzeroberfläche werden die Zeiten für die Berechnung der Preisschätzungen auf der rechten Seite in Abbildung 3.6 dargestellt. Diese Informationen sind während der Entwicklung interessant, um zu evaluieren, in welcher Größenordnung die Berechnungen stattfinden. Die Berechnungen benötigen meist zwischen 20 und 50 Millisekunden beim XGBRegressor. Beim MLPRegressor und der multiplen linearen Regression beträgt die Zeit zwischen 20 und 35 Millisekunden. Für den Endnutzer des Systems sind diese Informationen irrelevant und könnten ausgeblendet werden.

4.5.3 Mögliche Anwendungsgebiete

Ziel dieser Arbeit war es nicht ein fertiges System für die praktische Anwendung zu entwickeln. Dennoch ist dieser Abschnitt den Möglichkeiten auf Grundlage dieser Arbeit gewidmet.

Zunächst besteht die Möglichkeit, das System als Unterstützung für Makler einzusetzen. Dazu könnte im Bewertungsprozess einer Immobilie das System als einer von vielen

Faktoren genutzt werden. Unerfahreneren Maklern kann es dabei als automatisiertes Werkzeug bei der Immobilienbewertung helfen. Für eine Anwendung in einem Produktionsumfeld müssten die Mitarbeiter darüber aufgeklärt werden, dass die Ergebnisse nicht ausschließlich als korrekte Vorhersagen zu interpretieren sind.

Ebenfalls könnte das System für die Ersteinschätzung durch Immobilienbesitzer von Nutzen sein. Laien könnten sich so selbst vor Betrug schützen. Die Anwendung durch Immobilienbesitzer könnte zusätzlich von Firmen zur Akquise (Kundengewinnung) verwendet werden. Dazu wäre es möglich eine Website zur automatisierten Bewertung von Immobilien zu erstellen. Auf dieser könnte ein Verweis auf einen Makler oder ein anderes Unternehmen gegeben werden.

Zudem könnte eine automatisierte Bewertung auch in anderen wirtschaftlichen Bereichen Anwendung finden können, in denen die Werte von Immobilien verwendet werden.

Es ist jedoch zu beachten, dass diese Arbeit keine Automationen beinhaltet, die die Modelle fortlaufend an verändernde Gegebenheiten und neue Daten anpassen. Dafür müssten erneut alle Daten aus der Datenbank extrahiert und für das Training verwendet werden. Die Automation davon ist jedoch mit geringem Aufwand möglich. Damit könnten beispielsweise einmal im Monat die Modelle neu trainiert werden.

Für eine praktische Nutzung könnte die Leistung der Modelle weiter optimiert werden. Dazu könnte beispielsweise die Beschaffung zusätzlicher Daten für das Training nützlich sein.

5 Zusammenfassung

In diesem Kapitel wird basierend auf den Ergebnissen rückblickend ein Fazit mit Bezug auf die Zielsetzung aus der Einleitung gezogen. Abschließend wird ein Ausblick auf zukünftige Arbeit gegeben.

5.1 Fazit

Ziel dieser Arbeit war die Umsetzung und Evaluation von maschinellem Lernen für die automatisierte Bewertung von Immobilien in Deutschland. Hierfür sollten zwei Varianten des maschinellen Lernens durch zwei Modelle verglichen werden. Diese beiden Modelle sollten zudem mit der multiplen linearen Regression verglichen werden, welche bereits in anderen Ländern für die Schätzung von Immobilienpreisen eingesetzt wird. Zudem war eine prototypische Umsetzung in Form einer Benutzeroberfläche und eines Backends vorgesehen.

Mithilfe des maschinellen Lernens konnten sehr gute Ergebnisse für die automatisierte Bewertung von Immobilien in Deutschland erzielt werden. Der eXtreme Gradient Boosting Regression Tree aus Abschnitt 3.3.1 war dem Multi-Layer Perceptron Regressor aus Abschnitt 3.3.2 in allen Metriken überlegen. Beide Modelle des maschinellen Lernens, aber insbesondere der eXtreme Gradient Boosting Regression Tree, sind der multiplen linearen Regression aus Abschnitt 3.4 deutlich überlegen. Dabei wurde beschrieben, dass automatisierte Immobilienbewertungen keine perfekten Ergebnisse erzielen können, jedoch einen annähernden Richtwert bieten. Besonders für Immobilien mit einem Preis unter 1 250 000 € waren die Ergebnisse nah an den tatsächlichen Verkaufspreisen.

Der eXtreme Gradient Boosting Regression Tree konnte eine mittlere Abweichung von 81 773 € und einen Median der absoluten Abweichungen von 45 527 € erzielen. Ein R^2 von $0,781$ wurde dabei erreicht. Das Training wurde mit 789 Iterationen in 62 Sekunden durchgeführt.

Die als Ziel gesetzte prototypische Umsetzung in Form einer Benutzeroberfläche und eines Backend wurde in Abschnitt 3.6 implementiert. Auswertungen dazu haben gezeigt, dass die Benutzeroberfläche von Nutzern als leicht verständlich und übersichtlich gestaltet eingeschätzt wird. Das Backend aus Abschnitt 3.6.1 kann lokal ca. 15 Anfragen pro Sekunde bearbeiten.

Zusätzlich wurde ein Ausblick auf den Nutzen der Forschungsergebnisse in der praktischen Anwendung in Abschnitt 4.5.3 gegeben. Dabei wurde deutlich, dass die Forschungsergebnisse Grundlage für eine Vielzahl von Umsetzungen sein könnten. So könnte es sowohl für Makler als Unterstützung als auch für Einzelpersonen zur Ersteinschätzung bei der Immobilienbewertung von Nutzen sein.

Alles in allem könnte die Immobilienbewertung in der Zukunft entscheidend durch die Anwendung von künstlicher Intelligenz und Big Data verändert werden.

5.2 Zukünftige Arbeit

Mikrolage ausführlich analysieren

Für eine zukünftige Arbeit könnten weitere Verbesserungen in der Preisschätzung erzielt werden, indem die Mikrolage der Immobilie analysiert wird. Dazu könnten Informationen verwenden werden, die in Abschnitt 2.1 aufgelistet wurden. Diese lassen sich beliebig weit verfeinern und erweitern.

Es könnten beispielsweise Informationen über Schnittstellen von Kartendiensten entnommen werden. Diese könnten aufbereitet als Attribute für das Training und die Anwendung von Modellen Verwendung finden. Somit müsste der Nutzer keine weiteren Informationen zu der Lage eingeben, da die Längen- und Breitengrade bereits eingegeben werden können.

Es ist zu erwarten, dass dies die Immobilienbewertung weiter verbessern würde.

Natural Language Processing

Für Immobilien gibt es meist Texte, die verwendet werden, wenn die Immobilie beworben wird oder ein Exposé erstellt wird. Diese Texte könnten kodiert und den Modellen als

zusätzliche Attribute zur Verfügung gestellt werden. Dabei ist eine Verbesserung der Immobilienbewertung zu erwarten.

In dieser Arbeit lag jedoch der Fokus auf einer Simplifizierung der Benutzeroberfläche durch die Reduktion der Attribute.

Feedback von Maklern

Bei jeder Preisschätzung einer Immobilie könnte Feedback von den Maklern gegeben werden, wie zutreffend die jeweilige Schätzung ist. Diese Informationen könnten dazu verwendet werden die Algorithmen zu verbessern. Aus Feedback könnten neue Trainings- und Testdaten entstehen, die für das Training zusätzlich zur Verfügung gestellt werden. Außerdem sind diese Informationen hilfreich, um die Verwendung in einem produktiven Umfeld zu überwachen.

Mietpreise

Diese Arbeit hat sich nur mit der Schätzung von Verkaufspreisen befasst. Zukünftig könnte auch die Schätzung von Mietpreisen erforscht und evaluiert werden.

Ethik

Diese Arbeit befasst sich nicht mit den ethischen Herausforderungen des maschinellen Lernens. In einer zukünftigen Arbeit könnten die Auswirkungen einer automatisierten Preisbewertung von Immobilien mit Hinblick auf die ethischen Aspekte erforscht werden.

Literaturverzeichnis

[1] *BigQuery*. https://cloud.google.com/bigquery. 2022. – [Online; Zuletzt zugegriffen am 20.09.2022]

[2] Bonat, Wagner H. ; Kokonendji, Cé lestin C.: Flexible Tweedie regression models for continuous data. In: *Journal of Statistical Computation and Simulation* 87 (2017), apr, Nr. 11, S. 2138–2152. – URL https://doi.org/10.1080%2F00949655.2017.1318876

[3] Buodd, Martin F. ; Derås, Erlend J.: *Machine learning for property valuation : an empirical study of how property price predictions can improve property tax estimations in Norway*. 2020

[4] Chen, Tianqi ; Guestrin, Carlos: XGBoost: A Scalable Tree Boosting System. In: *CoRR* abs/1603.02754 (2016). – URL http://arxiv.org/abs/1603.02754

[5] Diederichs, Claus J.: *Immobilienmanagement im Lebenszyklus*. Springer-Verlag, 2006. – URL https://doi.org/10.1007/3-540-30965-9

[6] *Docker: Develop faster. Run anywhere*. https://docker.com/. 2022. – [Online; Zuletzt zugegriffen am 08.09.2022]

[7] Fahrmeir, Ludwig ; Kneib, Thomas ; Lang, Stefan: *Regression*. Springer Berlin Heidelberg, 2009. – URL https://doi.org/10.1007/978-3-642-01837-4

[8] Fasel, Daniel ; Meier, Andreas: *Big Data*. Wiesbaden : Springer Vieweg, 2016. – URL https://doi.org/10.1007/978-3-658-11589-0. – ISBN 9783658115890

[9] *FastAPI: FastAPI framework, high performance, easy to learn, fast to code, ready for production*. https://fastapi.tiangolo.com/. 2022. – [Online; Zuletzt zugegriffen am 30.08.2022]

[10] Friedman, Jerome H.: Greedy function approximation: A gradient boosting machine. In: *The Annals of Statistics* 29 (2001), Nr. 5, S. 1189 – 1232. – URL https://doi.org/10.1214/aos/1013203451

[11] Friedrichsen, Stefanie: *Immobilienbewertung*. Springer Fachmedien Wiesbaden, 2021. – URL https://doi.org/10.1007/978-3-658-32257-1

[12] Frost, Irasianty: *Einfache lineare Regression*. Springer Fachmedien Wiesbaden, 2018. – URL https://doi.org/10.1007/978-3-658-19732-2

[13] Gruhn, Volker (Hrsg.) ; Striemer, Rüdiger (Hrsg.): *The Essence of Software Engineering*. Springer International Publishing, 2018. – URL https://doi.org/10.1007/978-3-319-73897-0

[14] Hassairi, Abdelhamid: *Riesz Probability Distributions*. Berlin, Boston : De Gruyter, 2021. – URL https://doi.org/10.1515/9783110713374. – ISBN 9783110713374

[15] *Joblib: running Python functions as pipeline jobs*. https://joblib.readthedocs.io/en/latest/. – [Online; Zuletzt zugegriffen am 30.08.2022]

[16] Kingma, Diederik P. ; Ba, Jimmy: *Adam: A Method for Stochastic Optimization*. 2014. – URL https://arxiv.org/abs/1412.6980

[17] Kinnebrock, Werner: *Neuronale Netze*. De Gruyter, Dezember 1994. – URL https://doi.org/10.1515/9783486786361

[18] Kruse, Rudolf ; Borgelt, Christian ; Braune, Christian ; Klawonn, Frank ; Moewes, Christian ; Steinbrecher, Matthias: *Computational Intelligence*. Wiesbaden : Springer Vieweg, 2015. – URL https://doi.org/10.1007/978-3-658-10904-2. – ISBN 9783658109042

[19] Lämmel, Uwe ; Cleve, Jürgen: *Künstliche Intelligenz*. Carl Hanser Verlag GmbH & Co. KG, März 2020. – URL https://doi.org/10.3139/9783446463639

[20] *Lighthouse: Automated auditing, performance metrics, and best practices for the web*. https://github.com/GoogleChrome/lighthouse. 2022. – [Online; Zuletzt zugegriffen am 16.09.2022]

[21] Limsombunchai, Visit ; Gan, Christopher ; Lee, Minsoo: House Price Prediction: Hedonic Price Model vs. Artificial Neural Network. In: *American Journal of Applied Sciences* 1 (2004), 03

[22] *Locust: An open source load testing tool.* https://locust.io/. 2022. – [Online; Zuletzt zugegriffen am 04.09.2022]

[23] López, Osval Antonio M. ; López, Abelardo M. ; Crossa, José: *Multivariate Statistical Machine Learning Methods for Genomic Prediction.* Springer Cham, 2022. – URL https://doi.org/10.1007/978-3-030-89010-0. – ISBN 9783030890100

[24] Inc., Apple: *MacBook Pro: 13 Zoll, 2020, Vier Thunderbolt 3 Anschlüsse.* https://support.apple.com/kb/SP819. 2022. – [Online; Zuletzt zugegriffen am 02.09.2022]

[25] Nair, Vinod ; Hinton, Geoffrey: Rectified Linear Units Improve Restricted Boltzmann Machines Vinod Nair. In: *Proceedings of ICML* 27 (2010), 06

[26] *NextJS: The React Framework for Production.* https://nextjs.org/. 2022. – [Online; Zuletzt zugegriffen am 27.08.2022]

[27] Pathan, Mukaddim ; Buyya, Rajkumar ; Vakali, Athena: *Content Delivery Networks: State of the Art, Insights, and Imperatives.* In: Buyya, Rajkumar (Hrsg.) ; Pathan, Mukaddim (Hrsg.) ; Vakali, Athena (Hrsg.): *Content Delivery Networks.* Berlin, Heidelberg : Springer Berlin Heidelberg, 2008. – URL https://doi.org/10.1007/978-3-540-77887-5_1. – ISBN 978-3-540-77887-5

[28] Pokropp, Fritz: *Lineare Regression und Varianzanalyse.* Berlin, Boston : Oldenbourg Wissenschaftsverlag, 2015. – URL https://doi.org/10.1515/9783486786682. – ISBN 9783486786682

[29] *React: A JavaScript library for building user interfaces.* https://reactjs.org/. 2022. – [Online; Zuletzt zugegriffen am 27.08.2022]

[30] Rehkugler, Heinz: *Die Immobilien-AG: Bewertung und Marktattraktivität.* Berlin, Boston : Oldenbourg Wissenschaftsverlag, 2014. – URL https://doi.org/10.1515/9783486813807. – ISBN 9783486813807

[31] Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. In: *Psychological Review* 65 (1958), Nr. 6, S. 386–408. – URL http://dx.doi.org/10.1037/h0042519. – ISSN 0033-295X

[32] Russell, Stuart ; Norvig, Peter: *Artificial Intelligence: A Modern Approach.* 3. Prentice Hall, 2010

[33] *scikit-learn: Machine Learning in Python*. https://scikit-learn.org/. 2022. – [Online; Zuletzt zugegriffen am 25.08.2022]

[34] Sonnet, Daniel: *Neuronale Netze kompakt*. Wiesbaden : Springer Vieweg, 2022. – URL https://doi.org/10.1007/978-3-658-29081-8. – ISBN 9783658290818

[35] Stanton, Jeffrey M.: Galton, Pearson, and the Peas: A Brief History of Linear Regression for Statistics Instructors. In: *Journal of Statistics Education* 9 (2001), Nr. 3, S. null. – URL https://doi.org/10.1080/10691898.2001.11910537

[36] *TailwindCSS: Rapidly build modern websites without ever leaving your HTML*. https://tailwindcss.com/. 2022. – [Online; Zuletzt zugegriffen am 28.08.2022]

[37] Vandeput, Nicolas: *Data Science for Supply Chain Forecasting*. Berlin, Boston : De Gruyter, 2021. – URL https://doi.org/10.1515/9783110671124. – ISBN 9783110671124

[38] Wang, Shawn X. (Hrsg.): *Current Trends in Computer Science and Mechanical Automation Vol.1*. Warsaw, Poland : De Gruyter Open Poland, 2018. – URL https://doi.org/10.1515/9783110584974. – ISBN 9783110584974

[39] *XGBoost: eXtreme Gradient Boosting in Python*. https://xgboost.readthedocs.io/en/stable/. 2022. – [Online; Zuletzt zugegriffen am 07.09.2022]

[40] *XGBoost: Machine Learning Challenge Winning Solutions*. https://github.com/dmlc/xgboost/tree/master/demo#machine-learning-challenge-winning-solutions. 2022. – [Online; Zuletzt zugegriffen am 07.09.2022]

[41] Zucchini, Walter ; Schlegel, Andreas ; Nenadić, Oleg ; Sperlich, Stefan: *Statistik für Bachelor- und Masterstudenten*. Springer Berlin Heidelberg, 2009. – URL https://doi.org/10.1007/978-3-540-88987-8

A Anhang

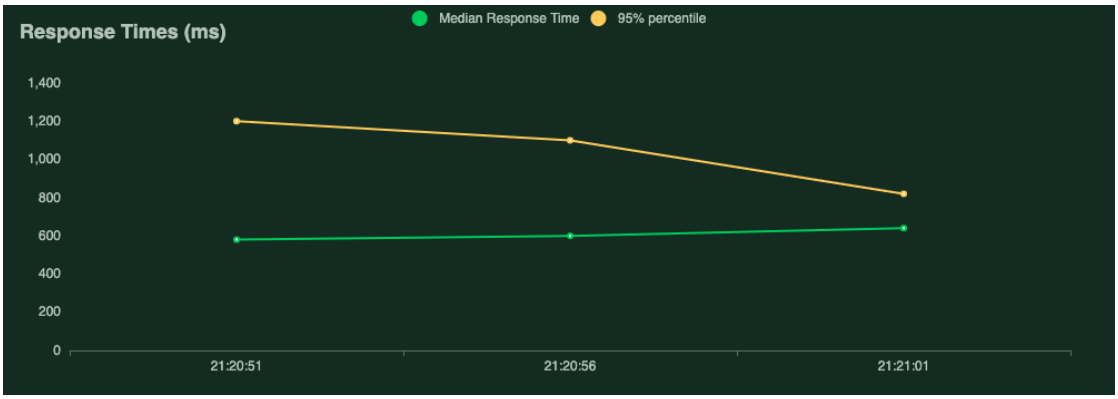


Abbildung A.1: Backend Stresstest: Antwortzeiten. In dem Testzeitraum von ca. 30 Sekunden wurde eine durchschnittliche Antwortzeit von ca. 600 Millisekunden erreicht. Die Antwortzeit im 95 % Perzentil sinkt von ca. 1200 Millisekunden auf ca. 800 Millisekunden ab. Auswertung in Abschnitt 4.5.1.

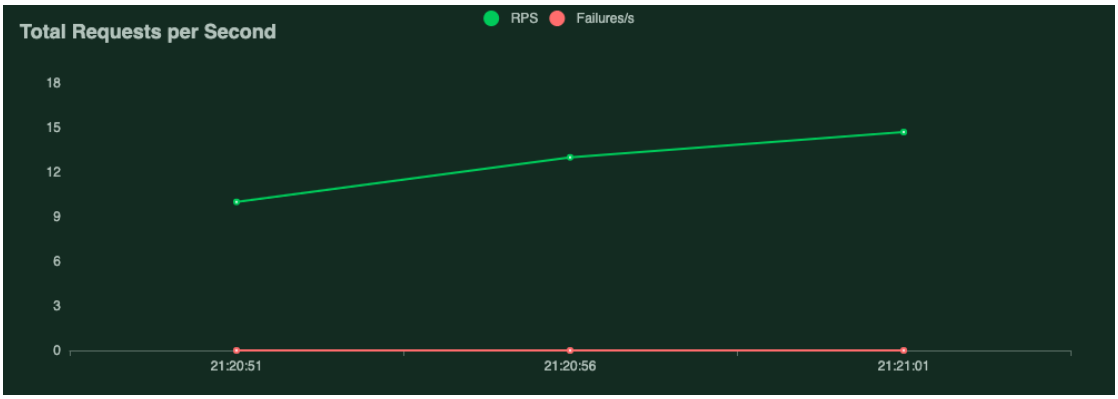


Abbildung A.2: Backend Stresstest: Gleichzeitige Anfragen. In dem Testzeitraum von ca. 30 Sekunden steigen die Anfragen pro Sekunde von 10 auf ungefähr 15. Auswertung in Abschnitt 4.5.1.

Kategorie	Beschreibung	Datentyp	Attribut
Preis	Netto Verkaufspreis	Gleitkommazahl	soldprice_net
Allgemein	Immobilientyp	String	country
	Immobilienuntertyp	String	property_subtype
	Jahr des Inserats	Zeitstempel	year_active
Lokalisierung	Landescode	String	country_iso
	Bundesland	String	region
	Stadt	String	city
	Gebiet	String	area
	Postleitzahl	String	postcode
	Breitengrad	Gleitkommazahl	latitude
	Längengrad	Gleitkommazahl	longitude
Eigenschaften	Grundstücksfläche	Gleitkommazahl	plot_surface
	Gesamtfläche	Gleitkommazahl	total_surface
	Wohnfläche	Gleitkommazahl	surface_living_area
	Baujahr	Ganze Zahlen	year_construction
	Immobilienzustand	String	property_condition
	Zimmer	Ganze Zahlen	rooms
	Schlafzimmer	Ganze Zahlen	bedrooms
	Badezimmer	Ganze Zahlen	bathrooms
	Etagen	Gleitkommazahl	stories
	Fahrstuhl vorhanden?	Boolean	elevator
	Balkone	Ganze Zahlen	balcony
	Pool vorhanden?	Boolean	swimmingpool
	Anzahl an Terrassen	Ganze Zahlen	terrace
	Fläche der Terrassen	Gleitkommazahl	terrace_size
	Himmelsrichtung	String	orientation
Energie	Energielabel	String	energy_label
	Energieeffizienzklasse	String	energy_efficiency_rating
	CO_2 Emissionsklasse	Zeitstempel	co2_emission_rating

Tabelle A.1: Alle 29 verfügbaren Attribute im Datensatz. Zu jedem Attribut ist die Kategorie, eine deutsche Beschreibung, der Datentyp und der interne Name des Attributs angegeben.

Glossar

Early Stopping Early Stopping ist eine Methodik der Regularisierung, die versucht Overfitting zu verhindern. Dabei wird das Training beendet, wenn sich die Leistung des Validierungsdatensatzes nicht weiter verbessert.

Engel & Völkers Engel & Völkers ist eine Unternehmensgruppe, die weltweit als Immobilienmakler Wohn- und Gewerbeimmobilien vermittelt.

Feature Importance Feature Importance gibt an, wie wichtig ein Attribut für die Vorhersagen eines Modells durchschnittlich ist. Dieser Wert wird meist zwischen $0,0$ und $1,0$ angegeben.

Gradient Boosting Gradient Boosting ist eine Lernmethode des maschinellen Lernens. Sie ist ein sogenanntes Ensemble Modell und verwendet meist Entscheidungsbäume.

JavaScript JavaScript ist eine Programmiersprache, welche primär für die Webentwicklung entwickelt wurde und eingesetzt wird. Der Kern der Sprache wurde als ECMAScript standardisiert.

Overfitting Overfitting bezeichnet die Überanpassung des Modells an den Trainingsdatensatz beim maschinellen Lernen.

Python Python ist eine Programmiersprache, welche unter anderem in der Datenanalyse und dem maschinellen Lernen weit verbreitet ist.

Regression Regression beschreibt die Vorhersage von stetigen und numerischen Werten beim maschinellen Lernen.

TypeScript TypeScript ist eine Programmiersprache, welche in JavaScript kompiliert wird und JavaScript um Typisierung und zusätzliche Funktionen erweitert.

Erklärung zur selbstständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original