Data Science 1 – Datenvorhersage zur Bepreisung von Immobilien in Deutschland

Projektbericht

eingereicht bei

Dr. Karsten Tolle

Professur für Datenbanken und Informationssysteme

Fachbereich Informatik

Goethe-Universität Frankfurt am Main

Eingereicht von: Julius Rubbe (Matrikelnummer: 7127061)

Timo Wehner (Matrikelnummer: 5972638)

Frederik Hering (Matrikelnummer: 7213283)

Ausführliche Beschreibung der Analyseskripte mit jeweiligem Code in Github:
https://github.com/juliusrubbe/DS_1

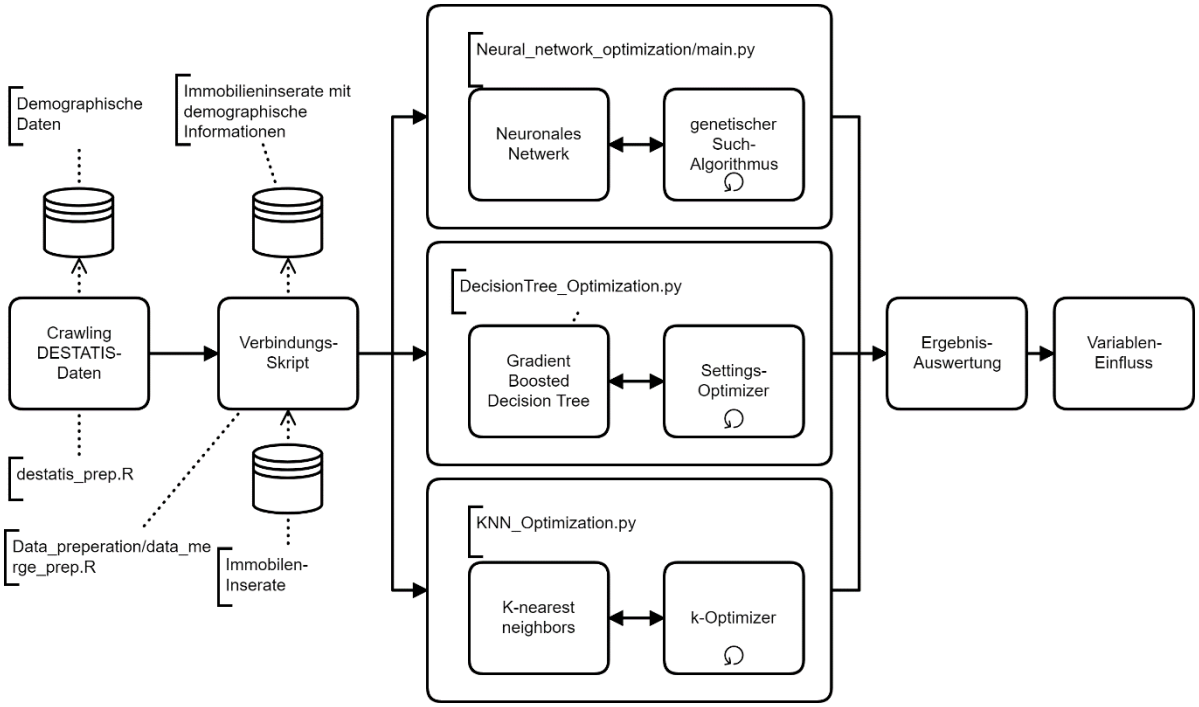
Master of Science in Wirtschaftsinformatik

Abgabetermin: 26.06.2020

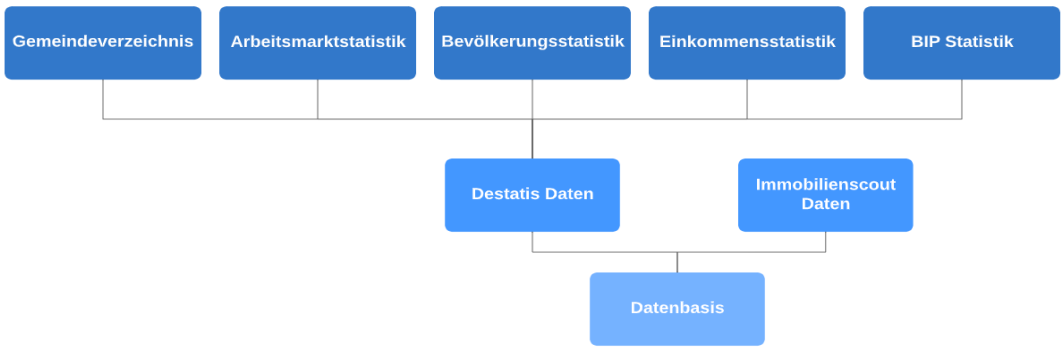
1. Einleitung

Die Immobilienpreise von Bestandswohnungen sind in den letzten 15 Jahren um über 60% gestiegen [1]. Durch dieses hohe Wachstum ist davon auszugehen, dass es speziell für private Vermieter schwierig ist, den Überblick über die aktuelle Preislage zu behalten. Als mögliche Hilfestellung hierzu überprüfen wir ein Vorhersagemodell, mit dem abhängig von regionalen Kennzahlen und den Wohnungsdetails der angemessene Quadratmeterpreis bestimmt werden kann. Dieses Modell könnte z.B. bei einem Wohnungsportal eingespeist werden: Dabei könnten die Kunden ihre Wohnungsdaten manuell eintragen und das System gibt unter Einbezug der regionalen und demografischen Informationen automatisch einen empfohlenen Preis aus.

Die Vorgehensweise des Projektes wird in der folgenden Abbildung dargestellt.

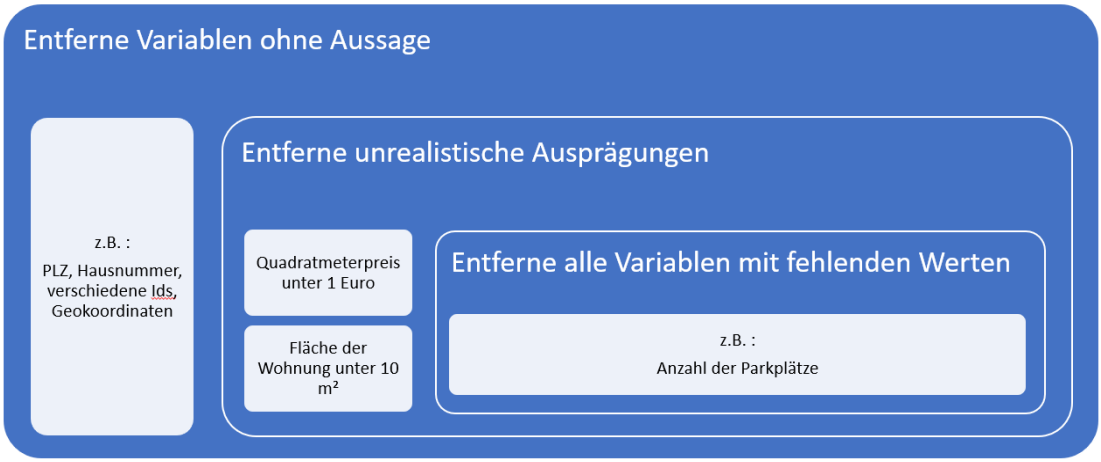


2. Datensätze



Es wurden zwei Datensätze in der Implementierung verwendet. Der erste Datensatz ist von kaggle.com und beinhaltet Informationen von Immobilienscout24.de. Der zweite Datensatz enthält Daten von destatis.de, in der obigen Abbildung ist zu erkennen wie sich die Datenbasis zusammensetzt.

Dabei mussten die Destatis Daten zunächst aufbereitet werden, um eine gemeinsame Datenbasis zu schaffen. Im Anschluss wurden die Destatis Daten und die Daten von Immobilienscout über den *Gemeindename* (Destatis) und den *geo_krs* (Immobilienscout) gemerged, wobei 56% aller Beobachtungen zugeordnet werden konnte. Dabei fällt auf, dass ein Großteil der Beobachtungen auf Ostdeutschland entfällt. Die weiteren Aufbereitungsschritte mit der gemergten Datenbasis sind in der folgenden Abbildung dargestellt.

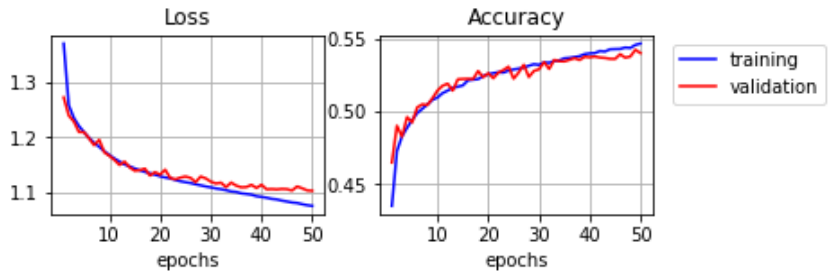


Nach diesen Aufbereitungsschritten beinhaltet der Datensatz 25 Variablen und 150.003 Beobachtungen. Um die Algorithmen im nächsten Schritt miteinander vergleichen zu können, wurden 20% des Datensatzes mit einem festgelegten Seed für die Validierung zurückgehalten. Für die Vorhersage wurde der Kaltmietpreis pro m² in folgende Outputklassen unterteilt:

Klasse	1	2	3	4	5	6	7	8
Quadratmeterpreis	<= 5	<= 7	<= 9	<= 11	<= 13	<= 15	<= 17	> 17

3. Algorithmen

3.1 Künstliche neuronale Netzwerke (NN)



Künstliche neuronale Netzwerke konnten sich in den letzten Jahren als State-of-the-Art Technologie für hochkomplexe Machine Learning Probleme etablieren [2]. So konnten NN Algorithmen immer wieder neue Bestwerte in ML Wettbewerben aufstellen [3]. Aufgrund ihrer Komplexität und der aufwändigen Modellierung einer guten neuronalen Netzwerkarchitektur braucht es viel Zeit und Expertenwissen [4]. Für die Suche nach einer geeigneten Architektur wurde ein genetischer Suchalgorithmus herangezogen, da diese die Performance einer klassischen Gridsuche übertreffen [5]. So können sich, ähnlich der biologischen Regeln „Survival-of-the-fittest“, die besten Parameter über die verschiedenen Generationen hinweg etablieren. Dieser Ansatz folgt den Arbeiten von [6] und [7]. Bei Betrachtung der Loss- und Accuracy-Kurven in der obigen Abbildung erkennt man, dass die Trainings- und Validierungskurve in die gleiche Richtung verlaufen. Dies kann als Anzeichen dafür gesehen werden, dass das Model nicht überschätzt, d.h. zu stark auf die Trainingsdaten ausgerichtet ist.

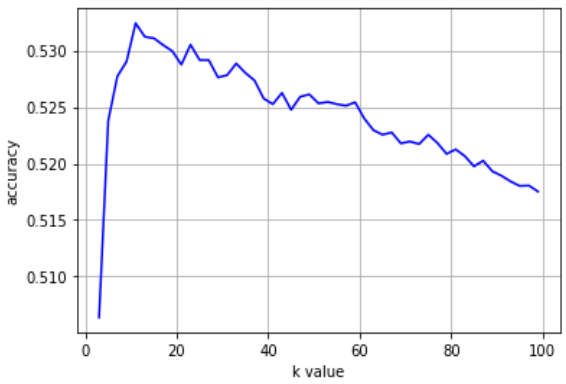
3.2 Gradient boosted decision tree

Auf Entscheidungsbäumen basierende Algorithmen zählen zu den meist verwendeten Algorithmen bei Klassifikationsaufgaben auf Kaggle – der größten Plattform für Data Science Wettbewerbe [8]. Wir haben uns konkret für "(Gradient-) Boosted Decision Trees" (GBD) entschieden, da sich diese insbesondere für Multi-Klassen-Probleme eignen [9]. Bei GBD's werden zwischen 20 und 1000 Entscheidungsbäume sequenziell aufgebaut [10]. Dabei optimiert sich das Modell von Baum zu Baum, da jeder Baum von den vorherigen bereits getestet und aufgebauten Bäumen lernt [10]. Die Verbesserung wird bei GPD's durch die Anpassung von Modell-Eigenschaften erreicht. Die verschiedenen konkreten Einstellungen, sowie kurze Beschreibungen, können in der nächsten Grafik gefunden werden. Diese Einstellungen werden in allen möglichen Kombinationen ausgeführt, wodurch insgesamt 135 verschiedene Modelle aufgebaut werden.

Einstellung	Beschreibung	Spezifikationen
Anzahl der Bäume	Anzahl der Bäume, die sequenziell hintereinander aufgebaut werden	[20, 100, 1000]
Max. Tiefe der Bäume	Die Bäume sind binär, die Tiefe gibt somit an wie häufig sich der Baum maximal teilen darf	[4, 5, 6, 7, 8]
Max. Anzahl der Variablen pro Baum	Die maximale Anzahl der Variablen, die pro Baum für die Teilung verwendet werden darf	[3, 7, 13]
Learning Rate	Ein Faktor mit dem gesteuert wird, wie hoch der Einfluss der vorigen Bäume auf einen Baum ist	[0.1, 0.25, 0.5]

3.3 K-Nearest-Neighbors (KNN)

Der KNN-Algorithmus ist ein relativ einfacher Klassifizierungsalgorithmus, der dem Ansatz Lazy Learnings folgt [11]. Dabei konnte sich dieser Algorithmus vor allem im e-Commerce Umfeld für die Kundensegmentierung etablieren [12] [13]. Die Vorteile des Algorithmus liegen in der einfachen Berechnung und der guten Interpretierbarkeit der Ergebnisse.



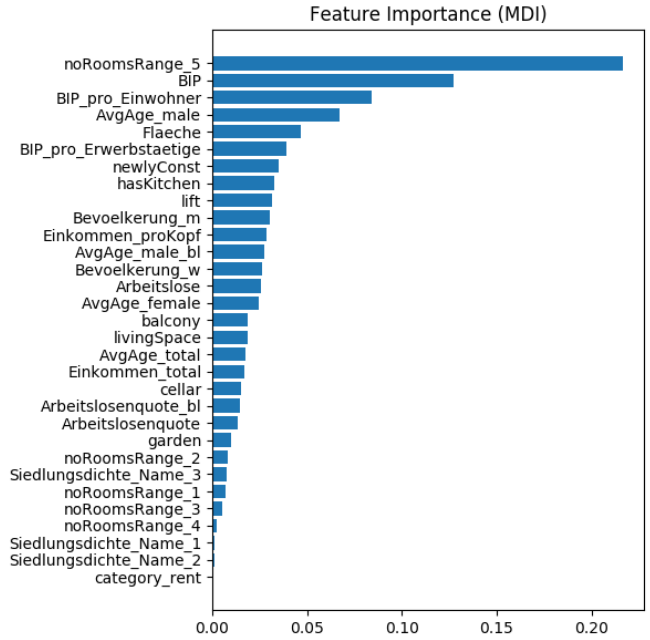
Der Algorithmus misst die Distanz von jeder Beobachtung jeweils zu allen anderen Beobachtungen. Im nächsten Schritt werden die k-ähnlichsten Beobachtungen, d.h. die mit der geringsten Distanz ausgewählt und bilden ein Cluster. Für die Vorhersage werden dann neue Beobachtungen dem nächstgelegenen Cluster zugeordnet und die Beobachtung nimmt die in diesem Cluster mehrheitlich vertretene Kategorie an.

Die Wahl eines geeigneten k-Wertes ist dabei nicht trivial. Deswegen wurde ein Optimierungsverfahren angewendet, bei dem ein optimaler Wert von *k=11* gefunden wurde. Die Accuracy für die einzelnen k-Werte ist in obiger Abbildung dargestellt.

4. Ergebnisse

	Algorithmus	Validation Acccuracy
1.	GBD	56.30%
2.	NN	53.78%
3.	KNN	53.07%

In der Tabelle ist die Performance der verschiedenen Algorithmen nach der Optimierung aufgezeigt. Dabei wird deutlich, dass GBD die anderen Algorithmen klar übertrifft und im Vergleich zu zufälligem Raten ein um 43.8 Prozentpunkte besseres Ergebnis erzielt. Für weitere Interpretationsschritte wird deswegen ausschließlich auf GBD eingegangen. In folgender Abbildung sind die wichtigsten Variablen des GBD Models aufgelistet, was mit Hilfe der Mean Decrease Impurity (MDI) erreicht wurde. Die MDI gibt die relative Häufigkeit einer Variablen im GDB an [14]. Dabei ist erkenntlich, dass die demografischen und ökonomischen Kennzahlen - wie z.B. das BIP - einen sehr großen Einfluss auf die Vorhersage des Modells haben. Kumuliert machen diese 69% des MDI - und damit des Modells - aus. Weniger Intuitiv ist der große Einfluss des Durchschnittsalters der Männer in der jeweiligen Gemeinde (AvgAge_male).



Zusammenfassend kann somit gesagt werden, dass sich die Kombination aus Wohnungsinformationen, demographischen und ökonomischen Kennzahlen zum Vorhersagen des Wohnungspreises eignet. Eine Applikation, welche Wohnungsinserierenden bei der Preisbestimmung unterstützt, könnte somit eine passende Ergänzung für Wohnungsplattformen sein.

An dieser Stelle muss zusätzlich noch auf eine Einschränkung der Datenbasis eingegangen werden, die Beobachtungen stammen schwerpunktmäßig aus Ostdeutschland. Hier kann es zu Verzerrungen gekommen sein.

5. Ausblick

Eine mögliche Verbesserung des Models könnte eine andere Verteilung der Klasseneinteilung von Quadratmeterpreisen sein. Beim Betrachten der gewählten Klassen gibt es deutliche Unterschiede in der Anzahl der Einträge. Eine der gewählten Klassen besitzt somit 30% der Daten, eine andere wiederum 3%. Hierbei könnte eine (relative-) Gleichverteilung (12.5% pro Klasse) zwischen den Klassen das Ergebnis entscheidend beeinflussen.

Klasse	Absolut	Relativ
1 (<= 5)	14759	10%
2 (<= 7)	45485	30%
3 (<= 9)	28641	19%
4 (<= 11)	20452	14%
5 (<= 13)	14021	9%
6 (<= 15)	8865	6%
7 (<= 17)	5152	3%
8 (> 17)	12628	8%

Zusätzlich könnte eine Betrachtung mit einer geringeren Anzahl an Klassen interessant sein. Darauf deutet hin, dass unser bester Algorithmus in 32,3 Prozentpunkten nur eine Klasse daneben lag (graue Felder in der nebenstehenden Abbildung). Somit erreichen wir eine Accuracy von 88,8%, wenn wir eine Abweichung von maximal einer Klasse als akzeptiertes Ergebnis annehmen. Bei einer geringeren Anzahl von Klassen kann mit einer noch besseren Performance gerechnet werden. In der rechts befindlichen Confusion Matrix des GBD, sind die Felder grau markiert, wenn der GBD eine Klasse daneben lag und gelb, wenn die Vorhersage richtig war. Final, könnte mehr Zeit in die Datenaufbereitung gesteckt werden, damit mehr Inserate mit regionalen Daten kombiniert werden können.

		Prediction							
		1	2	3	4	5	6	7	8
Actual	1	1560	1297	65	6	1	0	0	0
	2	731	7050	1204	157	28	16	2	9
	3	40	1624	2987	835	176	51	16	32
	4	14	269	1049	1739	604	173	33	135
	5	3	48	274	870	1000	303	74	226
	6	0	23	72	293	391	488	120	346
	7	0	8	15	137	150	174	202	348
	8	0	5	20	121	155	182	142	1908

Eine ausführliche Beschreibung der Analyseskripte mit jeweiligem Code kann in Github unter folgendem Link gefunden werden: **https://github.com/juliusrubbe/DS_1**

6. Literaturverzeichnis

[1] Deutschland in Zahlen - Wirtschaft & Gesellschaft — gezählt, gewogen, gewichtet, *Tabelle: Immobilienpreisindex - Index 1990=100*. [Online]. Available: https://www.deutschlandinzahlen.de/tab/deutschland/finanzen/preise/immobilienpreisindex (accessed: Jun. 24 2020).

[2] M. I. Jordan and T. M. Mitchell, “Machine learning: Trends, perspectives, and prospects,” *Science*, no. 349, pp. 255–260, 2015.

[3] J. Schmidhuber, “Deep Learning in neural networks: An overview,” *Neural Networks*, no. 61, pp. 85–117, 2015.

[4] B. Zoph and Q. V. Le, *Neural Architecture Search with Reinforcement Learning*. [Online]. Available: http://arxiv.org/abs/1611.01578 (accessed: Jun. 24 2020).

[5] J. Bergstra, J. and and Y. Bengio, Y., “Random search for hyper-parameter optimization,” *Journal of Machine Learning Research*, no. 13, 2012.

[6] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*. Reading, Mass., Wokingham: Addison-Wesley, 1989.

[7] D. T. Pham and D. Karaboga, *Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*. London: Springer London, 2000.

[8] M. He, J. Duan, and S. Zheng, *Kaggle Competition: Product Classification*. Computer Engineering Project.

[9] M. Saberian, P. Delgado, and Y. Raimond, “Gradient Boosted Decision Tree Neural Network,” Oct. 2019. [Online]. Available: http://arxiv.org/pdf/1910.09340v2

[10] T. Hastie, R. Tibshirani, and J. H. Friedman, *The elements of statistical learning: Data mining, inference, and prediction / Trevor Hastie, Robert Tibshirani, Jerome Friedman*, 2nd ed. New York: Springer, 2009.

[11] P. Hart, “The condensed nearest neighbor rule (Corresp.),” *IEEE Trans. Inform. Theory*, vol. 14, no. 3, pp. 515–516, 1968, doi: 10.1109/TIT.1968.1054155.

[12] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Analysis of recommendation algorithms for e-commerce,” in *EC'00: Proceedings of the 2nd ACM conference on electronic commerce*, Minneapolis, Minnesota, United States, 2000, pp. 158–167.

[13] R. Y. Du and W. A. Kamakura, “Measuring Contagion in the Diffusion of Consumer Packaged Goods,” *Journal of Marketing Research*, vol. 48, no. 1, pp. 28–47, 2011, doi: 10.1509/jmkr.48.1.28.

[14] C. Lee, *Feature Importance Measures for Tree Models - Part I: An Incomplete Review*. [Online]. Available: https://medium.com/the-artificial-impostor/feature-importance-measures-for-tree-models-part-i-47f187c1a2c3 (accessed: Jun. 24 2020).