ImmoRent

Ausgangssituation:

Wie kann ich den Preis für eine zu vermietende Wohnung in Abhängigkeit diverser Variablen wie Ortschaft, Anz. Zimmer, Wohnfläche, Baujahr, Stockwert etc. schätzen?

Ziel dieser Arbeit ist die Schätzung des monatlichen Mietpreises von Wohnungen. Mietpreise für Gewerbeflächen werden bei Möglichkeit in Betracht gezogen. Mieter können ihr aktuelles oder potentielles Mietobjekt mit einem verlässlichen Wert vergleichen, um zu evaluieren, ob Sie eher zu viel oder zu wenig Miete bezahlen. Des Weiteren kann das Modell dazu benutzt werden, um Regionen der Schweiz zu evaluieren, wo die eingegebene Wohnung innerhalb der definierten Kostengrenze liegt. Vermieter können mit diesem Modell einen Richtwert für das zu vermietende Objekt bekommen, um Marktgerechte Mieten anzubieten.

Als Datengrundlage dient https://www.immoscout24.ch. Diese Website gibt neben dem Mietpreis eines Objektes Auskunft über Attribute wie Adresse, PLZ & Wohnort, Miete / Monat, Zimmer, Wohnfläche, Objektart, Stockwerk, Baujahr. Diese Daten sollen noch mit weiteren Merkmalen, wie z.B. Distanz eines Objektes bis zum nächsten Bahnhof, von anderen Quellen angereichert werden, um die Mietpreisschätzung zu verbessern.

Ansatz:

In einem ersten Schritt müssen die Daten für die Erstellung der Modelle gesammelt werden. Diese werden von den benötigten Webseiten gesammelt und in ein File abgespeichert.

Die gesammelten Attribute werden qualitativ und quantitativ beschrieben, analysiert und wo es Sinn macht, visualisiert.

Die Mietpreisschätzung soll über ein flaches Neuronales Netz (NN) erfolgen, welches in Python implementiert wird. Für dessen Erstellung sollen keine Algorithmen aus Machine Learning-Libraries verwendet werden, d.h. die einzelnen Neuronen, Layers und Trainings-Algorithmen werden weitgehend selbst programmiert.

Zusätzlich soll das erstellte NN mit einem linearen Regressionsmodell verglichen werden, welches ebenfalls in Python implementiert wird. Für dieses Modell dürfen Werkzeuge aus beliebigen Libraries verwendet werden.

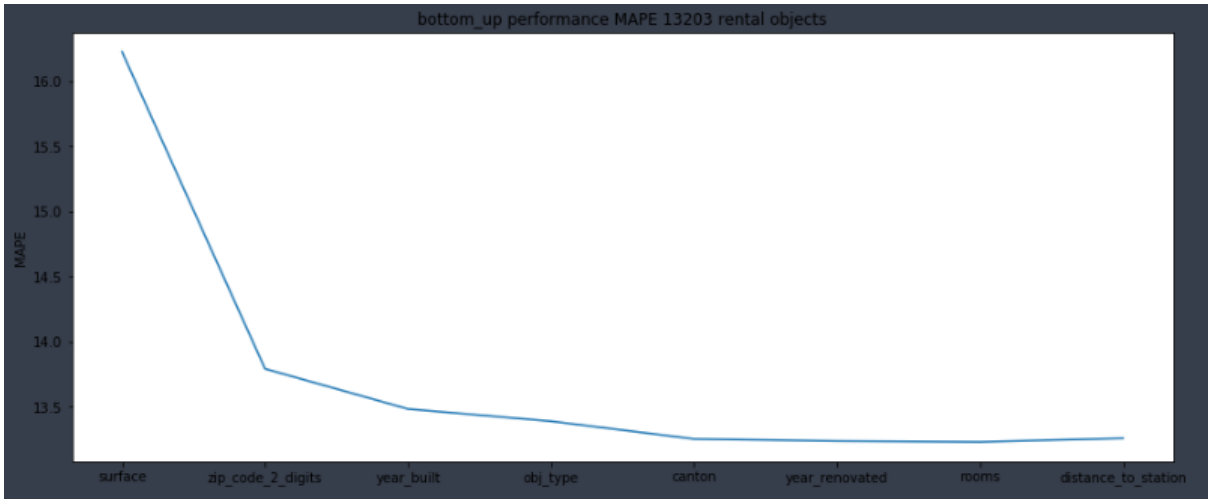
Ergebnisse Lineare Regression:

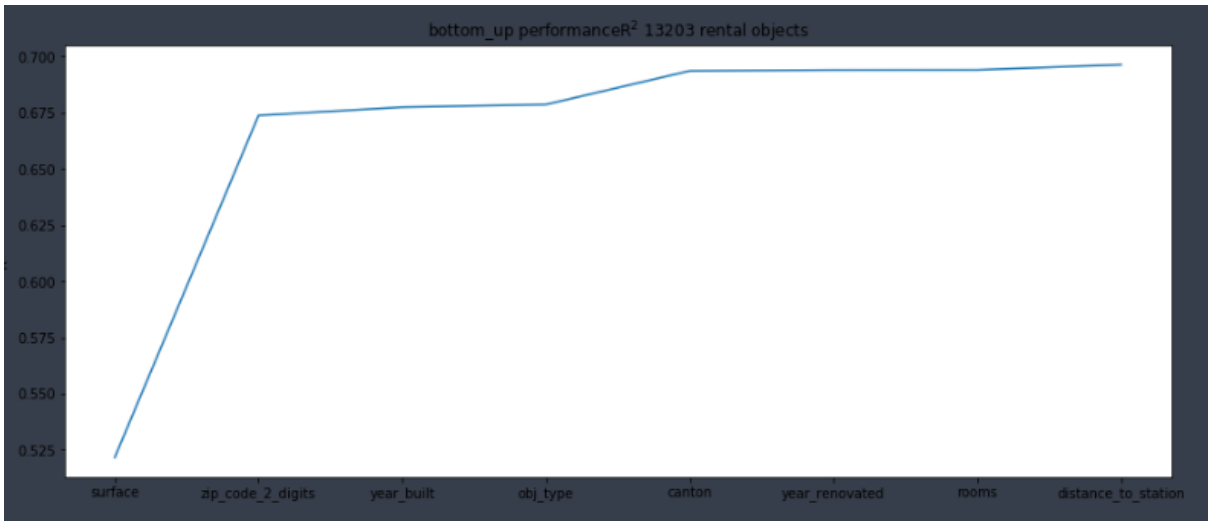
Das lineare Regressionsmodell erzielt einen MAPE von ca. 13.2 und ein R^2 von ca. 0.69 auf dem Testset. Dieses Ergebnis wurde mit einem greedy feature-selection Verfahren evaluiert.

Verwendet wurde ein bottom-up und ein top-down Verfahren. Beim bottom-up wird das erste Feature gewählt, welches allein den kleinsten Fehler erzeugt. Die Restlichen werden nach dem gleichen Prinzip hinzugefügt. Beim top-down Verfahren werden alle Features für die Erstellung des Modells verwendet. Dabei werden nacheinander diejenigen Features entfernt, welche das Modell verbessern oder am wenigsten verschlechtern.

Bottom-Up:

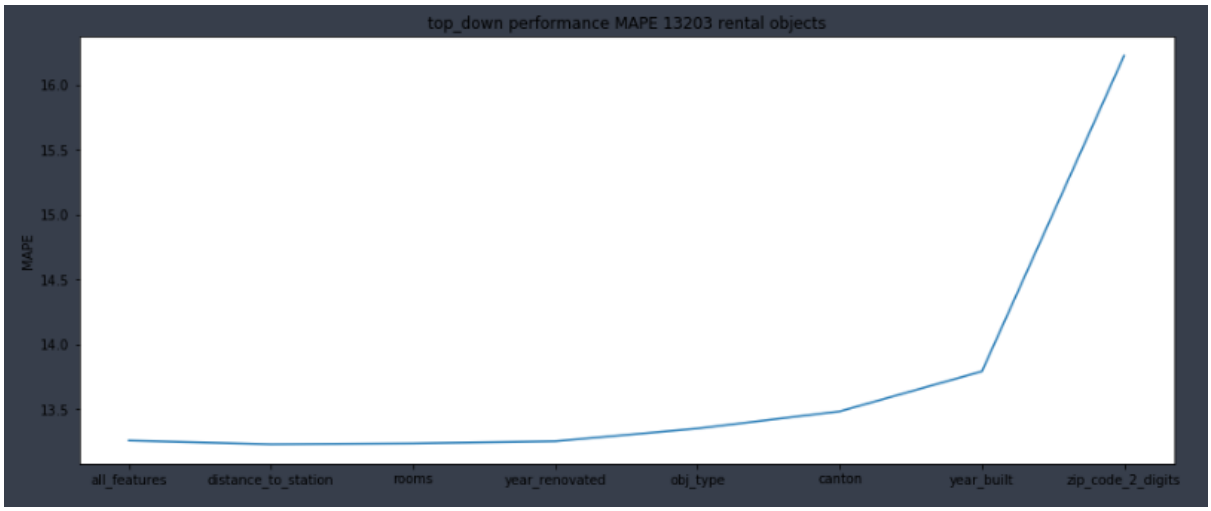
	feature	mape	r2
0	surface	16.221399	0.521657
1	zip_code_2_digits	13.792151	0.673795
2	year_built	13.484028	0.677488
3	obj_type	13.390053	0.678692
4	canton	13.254276	0.693643
5	year_renovated	13.238133	0.694038
6	rooms	13.231294	0.694077
7	distance_to_station	13.259534	0.696417

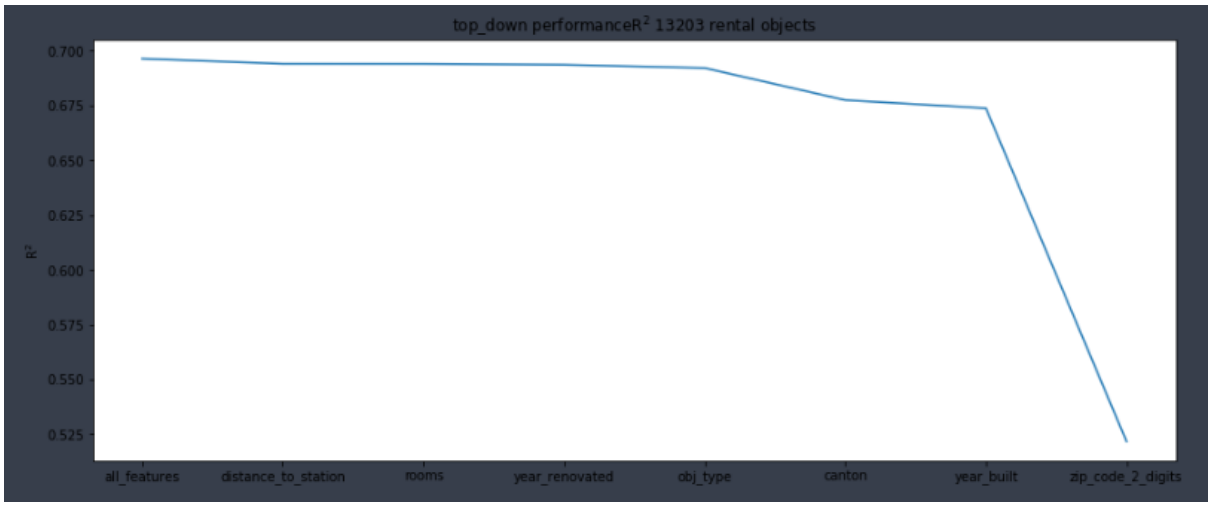




Top-Down:

	feature	mape	r2
0	all_features	13.259534	0.696417
1	distance_to_station	13.231294	0.694077
2	rooms	13.238133	0.694038
3	year_renovated	13.254276	0.693643
4	obj_type	13.352463	0.692148
5	canton	13.484328	0.677488
6	year_built	13.792151	0.673795
7	zip_code_2_digits	16.221399	0.521657

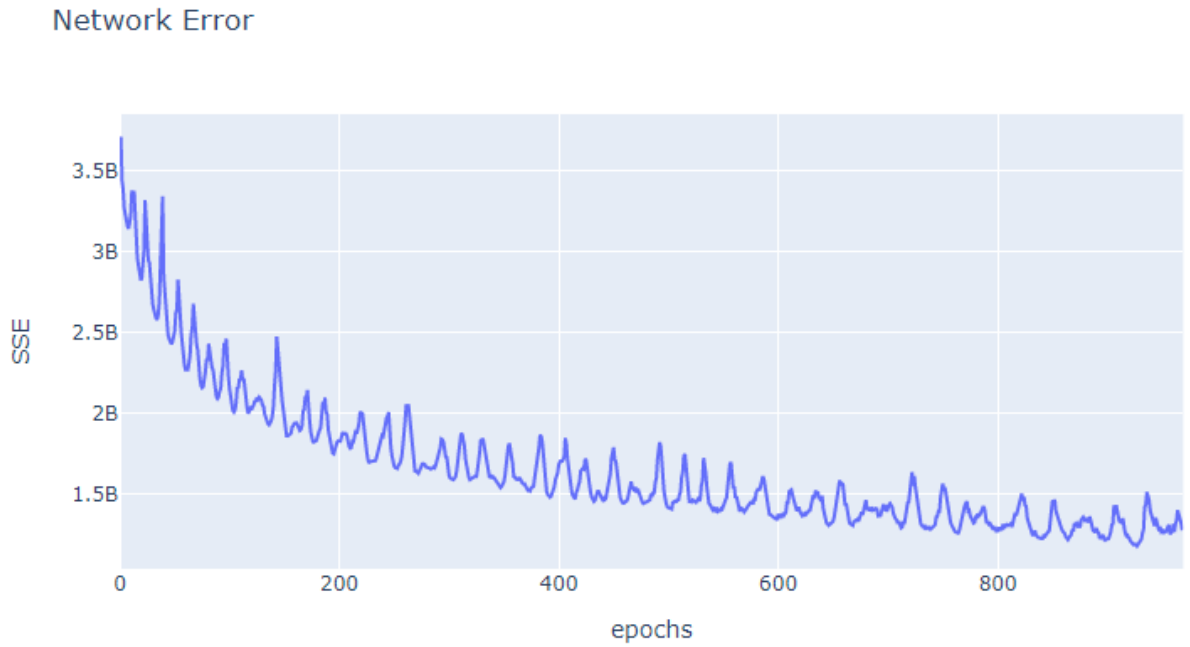




Ergebnisse Neuronales Netzwerk:

Das beste Ergebnis des künstlichen Neuronalen Netzwerks (kNN) erzielt einen MAPE von 10.6 auf dem Testset. Es wurde ein Hidden-Layer der Grösse 80 verwendet. Als Regularisierungsparameter L2 wurde alpha = 0.001 genommen. Des Weiteren trainierte das Modell mit einer konstanten Learning-Rate von 0.01. Die Anzahl der Trainingsiterationen sind auf 1000 beschränkt. Die Lernkurve zeigt auch nach 1000 Iterationen eine stetige Verbesserung der Kosten. Aus Zeitgründen jedoch wurde auf eine viel höhere Anzahl an Iterationen verzichtet.

Optimiert wurde mit dem 'gradient descent' Verfahren, wobei aufgrund der relativ geringen Datenmenge und Netzwerkgrösse auf Methoden wie 'stochastic gradient descent' oder 'minibatch' verzichtet wurde.



	features	layers	learning_rate	alpha	batch_size	residuals	MAPE
0	[obj_type, canton, zip_code_2_digits, surface,...	(80,)	0.01	0.000	none	[-244.48175387548167, -307.262058828574, -116....	11.849378
1	[obj_type, canton, zip_code_2_digits, surface,...	(80,)	0.01	0.000	none	[-203.2536971220245, 6.011481906470408, -210.0...	11.435508
2	[obj_type, canton, zip_code_2_digits, surface,...	(80,)	0.01	0.000	none	[-216.5675113669447, -159.86110521480646, -101...	11.308059
3	[obj_type, canton, zip_code_2_digits, surface,...	(80,)	0.01	0.001	none	[-179.822233826912, -36.175313266719286, -339....	11.768765
4	[obj_type, canton, zip_code_2_digits, surface,...	(80,)	0.01	0.001	none	[-251.0008088320269, -175.47334544745172, -296...	11.352812
5	[obj_type, canton, zip_code_2_digits, surface,...	(80,)	0.01	0.001	none	[-212.62448131932172, 58.46136521125368, -109....	10.606158

Endergebnis:

Somit schlägt das selbst gebaute kNN das von sklearn verwendete lineare Regressionsmodell. Ich erachte es möglich, durch eine bessere Wahl der Hyperparameter des KNNs an noch bessere Resultate zu kommen.

Obwohl das KNN das bessere Modell ist, finde ich einen MAPE von über 10 als relativ hoch und nicht für die Praxis zu gebrauchen. Einerseits könnte das an den gesammelten Daten liegen, da nur sichere falsche Werte herausgefiltert wurden. Andererseits ist es auch gut möglich, dass ein Tiefes Neuronales Netzwerk bessere Ergebnisse liefert (sofern die Datenmenge ausreichend ist).