

Immobilien Challenge

Kevin Wartmann, Denis Schatzmann

12. Januar 2025

Abstract

In dieser Challenge wurde ein umfangreicher Immobilien-Datensatz, gescrappt von Immoscout und Homegate, analysiert und modelliert. Ziel war es, verschiedene Ansätze zur Vorhersage von Immobilienpreisen und zur Klassifizierung von Gebäudetypen zu entwickeln und zu evaluieren. Zunächst wurde ein einfaches lineares Regressionsmodell implementiert, um den Gebäudepreis basierend auf der Wohnfläche vorherzusagen. Daraufhin folgte die Teilnahme an einem Kaggle-Wettbewerb, in dem komplexere Regressionsmodelle optimiert wurden. Die Modelle wurden anhand der Mean Absolute Percentage Error (MAPE) bewertet, um die Vorhersagegenauigkeit zu maximieren. Ergänzend wurde ein Webserver entwickelt, der es ermöglicht, den Preis einer Immobilie basierend auf grundlegenden Gebäudeinformationen in Echtzeit vorherzusagen. Abschliessend wurden verschiedene Klassifikationsmodelle getestet, um den Gebäudetyp zuverlässig zu bestimmen. Diese umfassende Herangehensweise kombinierte Datenanalyse, Modelloptimierung und Webentwicklung zu einer praxisnahen Lösung im Bereich der Immobilienbewertung.

Inhaltsverzeichnis

Abstract	1
Inhaltsverzeichnis	2
1 Einleitung	4
1.1 Thema	4
2 Hauptteil	6
2.1 Daten	6
2.1.1 Übersicht	6
2.1.2 Herkunft der Daten	7
2.2 Explorative Datenanalyse	8
2.2.1 Preis	8
2.2.2 Wohnfläche	9
2.2.3 Immobilientypen und Preisverteilung	10
2.2.4 Anzahl Zimmer	10
2.3 Datenselektion	12
2.3.1 Herausforderungen mit den Daten	12
2.3.2 Mögliche Bias	12
2.3.3 Entfernte Spalten	14
2.3.4 Feature engineering	20
2.4 Imputation	22
2.4.1 Kein Imputer	22
2.4.2 Simple Imputer	22
2.4.3 KNN-Imputer	22
2.5 Modelle Regression	23
2.5.1 Metrik	23
2.5.2 Lineare Regression	23
2.5.3 Splines	25
2.5.4 Random Forest	27
2.5.5 Hist-Gradient-Boosting	28

2.5.6	K-Nearest Neighbors	29
2.5.7	XGBoost	31
2.5.8	MLP	33
2.6	Modelle zur Klassifikation	36
2.6.1	Preprocessing	36
2.6.2	Metrik: Gewichteter Recall	37
2.6.3	Logistische Regression	37
2.6.4	Desischon trees	38
2.6.5	Random Forest	38
2.6.6	XGBoost-Classifier	40
2.6.7	Zusammenfassung	41
2.7	Webauftritt	42
2.7.1	Voraussetzungen	42
2.7.2	Bedienung der Startseite	43
2.7.3	Ablauf des Vorhersageprozesses	43
2.7.4	Installation	44
2.8	Projektstruktur	45
2.8.1	Projektmanagement	45
2.8.2	Versionskontrolle	45
2.9	Fazit	47
2.10	Ausblick	47

Kapitel 1

Einleitung

1.1 Thema

In der [Immobilienrechner-Challenge](#) beschäftigen wir uns mit der Vorhersage von Immobilienpreisen in der Schweiz sowie der Klassifikation von Immobilientypen. Immobilien sind laut der Deutschen Nationalbibliothek (n.d.) unbewegliche Sachgüter, die Grundstücke und darauf befindliche Gebäude umfassen. Der Wert einer Immobilie wird durch zahlreiche Faktoren wie Standort, Grösse, Zustand und Nutzungsmöglichkeiten beeinflusst. Mit Hilfe von Machine Learning und explorativer Datenanalyse wollen wir diese komplexen Zusammenhänge besser verstehen und mehrere Modelle entwickeln, die bessere Vorhersagen ermöglicht.

Der Datensatz, den wir analysieren, umfasst Immobilieninformationen aus der Schweiz, ergänzt durch Gemeinde-Daten. Ziel ist es, den Wert der Immobilien so genau wie möglich vorherzusagen, wobei als Metrik der MAPE gilt. Gleichzeitig wollen wir den Einfluss der verschiedenen Attribute auf den Preis analysieren, um nachvollziehbare und erklärbare Modelle zu schaffen.

In der Immobilienbewertung gibt es verschiedene etablierte Ansätze. Traditionelle Methoden wie die Vergleichswertmethode oder die Ertragskapitalisierung (Loepfe, 2017) haben ihre Berechtigung, sind jedoch oft zeitaufwendig und schwer skalierbar. Durch datengetriebene Ansätze wie die hedonische Methode (Noth, 2025) können wir den Prozess automatisieren und gleichzeitig die Genauigkeit verbessern. Mit unserem Projekt wollen wir die Anwendung solcher Modelle auf Schweizer Immobilien evaluieren und verbessern.

Diese Challenge bietet uns die Möglichkeit, praxisnah an einem datenintensi-

ven Problem zu arbeiten. Dabei entwickeln wir nicht nur unsere technischen Fähigkeiten weiter, sondern erhalten auch Einblicke in die Komplexität und Relevanz der Immobilienbewertung. Unser Ziel ist es, ein Modell zu schaffen, das sowohl präzise als auch praktikabel ist.

Kapitel 2

Hauptteil

2.1 Daten

2.1.1 Übersicht

Wir haben die Daten als .csv-Datei von F. Benites erhalten. Sie umfassen 22481 Datenpunkte und 134 Spalten.

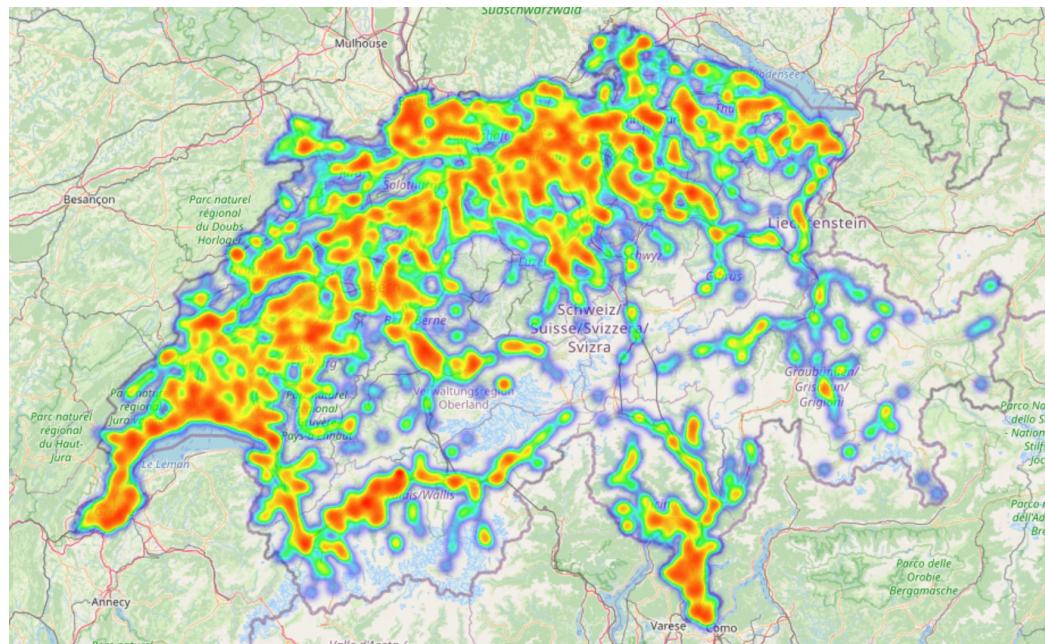


Abbildung 2.1: Verteilung der Datenpunkte Schweiz

2.1.2 Herkunft der Daten

Die Daten wurden von F. Benites gescraped. Die beiden Quellen sind Immoscout24.ch mit 12'044 Datenpunkten und homegate.ch mit 8785 Datenpunkten. Die Daten wurden im Jahr 2022 zwischen August und Dezember gesammelt. Einige Spalten wurden durch Gemeindespezifische oder Standortspezifische Informationen ergänzt (bspw. Steuerfuss oder Distanz zum nächsten Bahnhof).

<https://spaces.technik.fhnw.ch/spaces/immobilienrechner-1/lernmaterialien>

2.2 Explorative Datenanalyse

Der untersuchte Datensatz enthält insgesamt 22.481 Beobachtungen und 132 Spalten. Die wichtigsten Ergebnisse der ersten Analyse sind:

- **Keine Duplikate:** Im Datensatz wurden keine doppelten Einträge gefunden.
- **Fehlende Werte:** Insgesamt liegen 1.179.856 fehlende Werte vor.
- **Spalten ohne fehlende Werte:** 48 von 132 Spalten enthalten keine fehlenden Werte.
- **Kombination der Spalten:** Die Zusammenführung verschiedener Spalten wurde in Kapitel [2.3.3](#) erläutert.

Einige Spalten, wie zum Beispiel *Minimum floor space*, wiesen sehr viele fehlende Werte (z. B. 22.479) auf, die nicht sinnvoll aufgefüllt werden konnten. Daher wurden solche Spalten aus dem Datensatz entfernt.

2.2.1 Preis

Für einen Überblick über die Preisspanne und deren Verteilung wurde die Preisverteilung der Immobilien genauer untersucht.

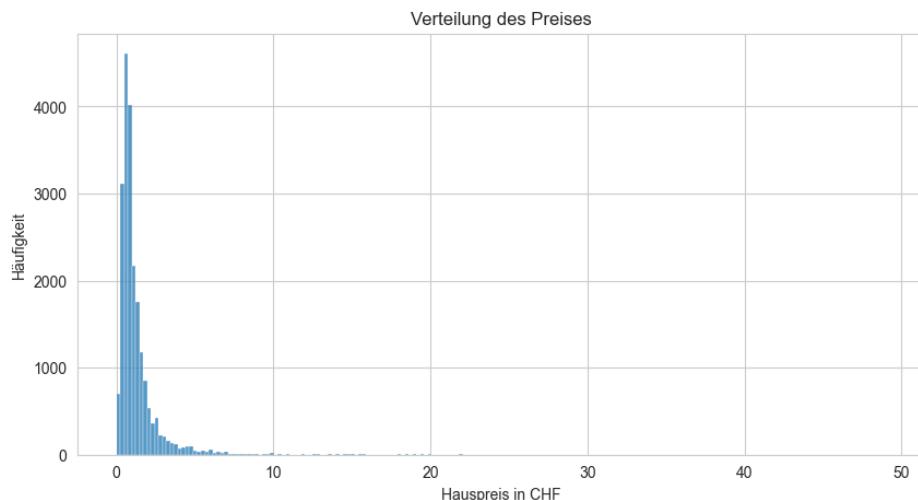


Abbildung 2.2: Verteilung der Spalte `price_cleaned`

In Abbildung [2.2](#) ist eine stark linksgeneigte (linkssteile) Verteilung zu erkennen: Die meisten Immobilienpreise liegen im unteren Bereich, während nur wenige Objekte sehr hochpreisig sind.

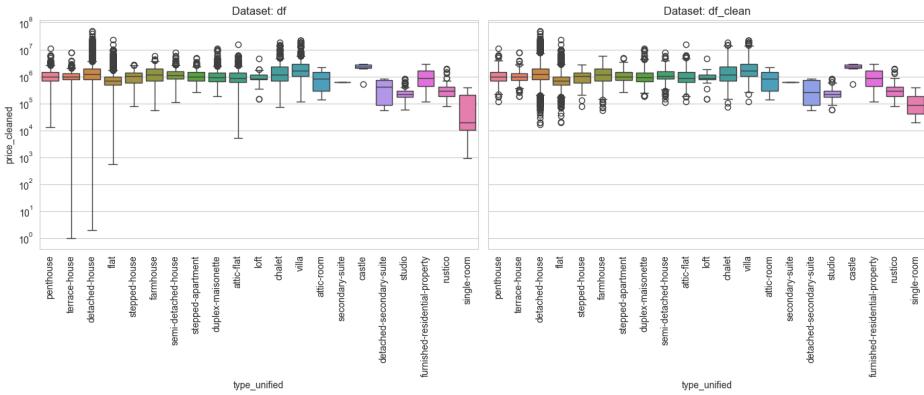


Abbildung 2.3: Bereinigung des Preises

Die Boxplot-Darstellung (Abbildung 2.3) verdeutlicht, dass es Ausreißer im sehr hochpreisigen Bereich gibt. Im Zuge der Datenbereinigung wurden Immobilien mit einem Preis unter 16.000 CHF entfernt, da insbesondere extrem kleine Werte (etwa 1 CHF) als Platzhalter für neue, noch unbepreiste Immobilien eingetragen waren.

2.2.2 Wohnfläche

Ein weiterer wichtiger Faktor ist die Verteilung der Wohnflächen.

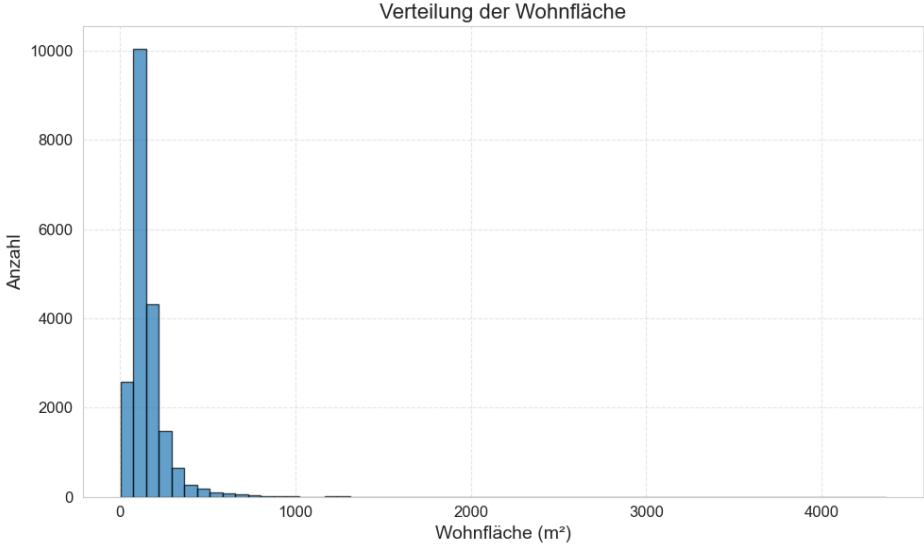


Abbildung 2.4: Verteilung der Wohnfläche

Das Histogramm in Abbildung 2.4 zeigt, dass die meisten Immobilien eine eher kleine Wohnfläche besitzen. Auch hier ist eine linkssteile Verteilung zu erkennen, da nur wenige Immobilien über eine sehr grosse Wohnfläche verfügen.

2.2.3 Immobilientypen und Preisverteilung

Zur weiteren Analyse wurde die Preisverteilung nach verschiedenen Immobilientypen untersucht. Die Boxplot-Darstellung in Abbildung 2.3 (siehe oben) zeigt, dass die bereinigten Preise der jeweiligen Typen nah beieinander liegen. Dies deutet auf eine gelungene Bereinigung hin, da unplausibel niedrige und eindeutig zu hohe Werte entfernt wurden.

2.2.4 Anzahl Zimmer

Um Wohnfläche und Anzahl Zimmer zu vervollständigen, wurde die Spalte `detail` hinzugezogen. Dies führte beinahe zu Fehlern, da in manchen Fällen die angegebene Fläche statt der Wohnfläche nur die Nutzfläche war. Dennoch konnten rund 500 fehlende Werte bei der Anzahl Zimmer auf diese Weise erfolgreich ergänzt werden.

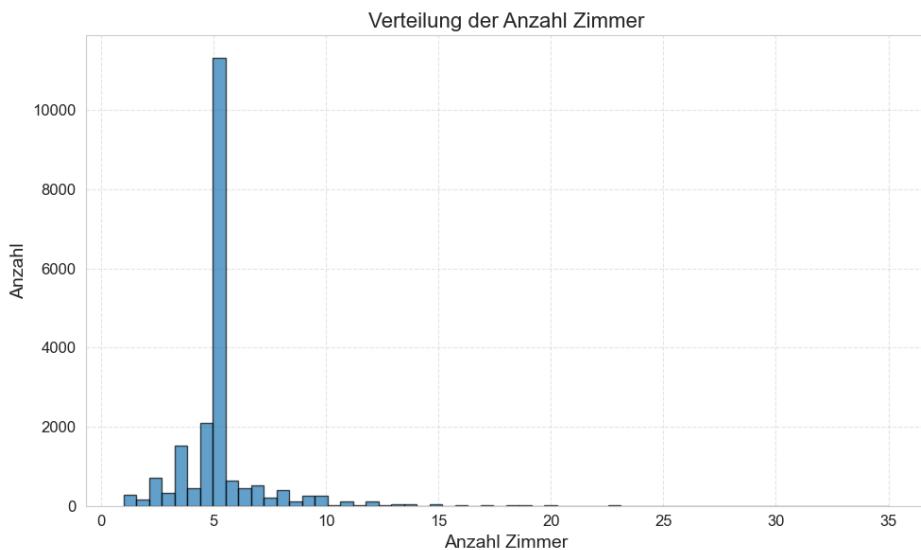


Abbildung 2.5: Verteilung der Anzahl Zimmer

Abbildung 2.5 veranschaulicht die Verteilung der Anzahl Zimmer im Datensatz.

Fazit

Abschliessend lässt sich festhalten, dass die Datenbereinigung und Analyse dank detaillierter Verfahren weitreichende Einblicke in die Immobilienpreise, Wohnflächen und Zimmeranzahlen liefern konnte. Da es in diesem Datensatz jedoch sehr viele Spalten gibt, können zusätzliche Analysen und weiterführende Ergebnisse in unserem GitHub-Repository eingesehen werden.

https://github.com/staldenhoefler/immo_challenge

2.3 Datenselektion

2.3.1 Herausforderungen mit den Daten

Die Qualität des Datensatzes ist grundsätzlich gut. Wir sind jedoch auf einige Herausforderungen gestossen.

Zum einen hat die Schweiz einige verschiedene Landessprachen. Dies führt dazu, dass bis zu sechs Spalten mit unterschiedlichen Namen existieren, welche aber das gleiche Aussagen. Ein Beispiel wäre hier die Spalten Availability, detail_responsive#available_from, Available_merged, Verfügbarkeit, Disponibilité und Disponibilità.

Ein zweites Problem sind die vielen leeren Zellen. Selbst nach dem Zusammenführen der gleichen Spalten haben wir immer noch sehr viele NaN-Values. Das sind Spalten wie Beispielsweise "Number of Apartments". Da fehlen fast 20'000 der insgesamt 20'829 Datenpunkten.

2.3.2 Mögliche Bias

Zielvariable \neq Marktwert

Da die Daten von Verkaufsportalen stammen, ist anzunehmen, dass die Immobilien welche darauf zu finden sind, noch nicht verkauft wurden. Da wir keine Angaben dazu haben, wie lange die Objekte jeweils schon ausgeschrieben sind, können wir aber dazu keine fundierte Statistik erstellen. Unsere These ist jedoch, dass die Preise der Immobilien bei allen Hausern über dem Marktwert liegen.

Region

Wir haben in der West- und Nordwest-Schweiz viel mehr Datenpunkte wie Beispielsweise in der Inner- oder Ost-Schweiz. Siehe [Abb. 2.1] Dies führt dazu, dass unsere Modelle ihre Präzision in den Regionen mit wenig Daten verringern. Mögliche Gegenstrategien wäre das Gruppieren der vorhandenen Datenpunkte in gleichgroße Cluster.

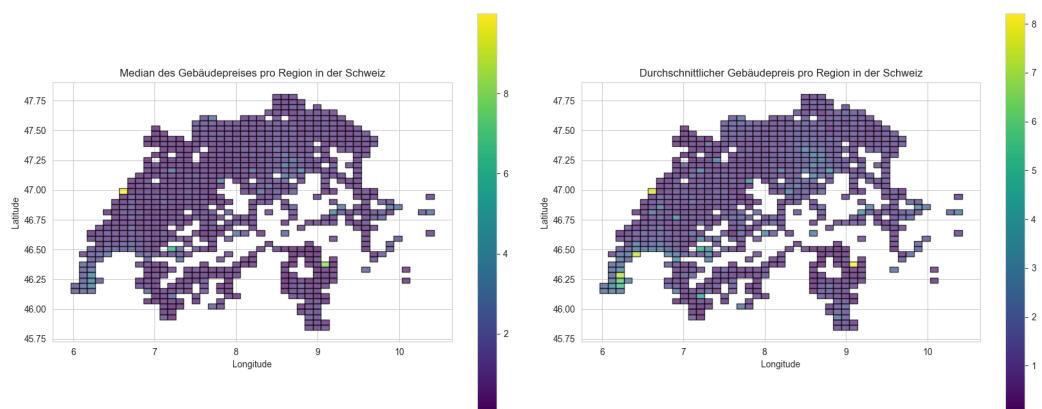


Abbildung 2.6: Verteilung der Gebäudepreise nach Region

Plattformabhängigkeit

2.3.3 Entfernte Spalten

Spalte	Aktion	Begründung
Unnamed: 0.1	Entfernt	Ist ein Index ohne wertvolle Information
Unnamed: 0	Entfernt	Ist ein Index ohne wertvolle Information
Municipality	Entfernt	Gemeindeinformationen verwenden wir mit der Postleitzahl
Living space	Entfernt	Infos bereits in Space extracted
Plot area	Entfernt	Infos bereits in Plot_area_unified
Floor space	Entfernt	Infos bereits in Floor_space_merged
Availability	Behalten	Wird ergänzt mit div. anderen Spalten
location	Entfernt	Gemeindeinformationen verwenden wir mit der Postleitzahl
description	Entfernt	Zu unstrukturierte Informationen
detailed_description	Entfernt	Zu unstrukturierte Informationen
url	Entfernt	Link ausgelaufen provider separat vorhanden
table	Entfernt	F.Benites hat alles in sep. Spalten extrahiert
Floor	Behalten	Wird ergänzt mit div. anderen Spalten
detail_responsive #municipality	Entfernt	Gemeindeinformationen verwenden wir mit der Postleitzahl
detail_responsive #surface_living	Entfernt	Ergänzt die Spalte Space extracted
detail_responsive #floor	Entfernt	Ergänzt die Spalte Floor
detail_responsive #available_from	Entfernt	Ergänzt die Spalte Availability

Gemeinde	Entfernt	Gemeindeinformationen verwenden wir mit der Postleitzahl
Wohnfläche	Entfernt	Infos bereits in Space extracted
Stockwerk	Entfernt	Infos bereits in Floor
Nutzfläche	Entfernt	Infos bereits in detail_responsive #surface_usable
Verfügbarkeit	Entfernt	Ergänzt die Spalte Availability
Grundstücksfläche	Entfernt	Infos bereits in Plot_area_unified
detail_responsive #surface_property	Entfernt	Ergänzt die Spalte Plot_area_unified
detail_responsive #surface_usable	Behalten	Nutzfläche
Commune	Entfernt	Gemeindeinformationen verwenden wir mit der Postleitzahl
Surface habitable	Entfernt	Infos bereits in Space extracted
Surface du terrain	Entfernt	Infos bereits in Plot_area_unified
Surface utile	Entfernt	Infos bereits in detail_responsive#surface_usable
Disponibilité	Entfernt	Ergänzt die Spalte Availability
Étage	Entfernt	Infos bereits in Floor
Comune	Entfernt	Gemeindeinformationen verwenden wir mit der Postleitzahl
Superficie abitabile	Entfernt	Infos bereits in Space extracted
Disponibilità	Entfernt	Ergänzt die Spalte Availability
Gross return	Entfernt	Zu viele fehlende Werte
Piano	Entfernt	Infos bereits in Floor
Superficie del terreno	Entfernt	Infos bereits in Plot_area_unified
Superficie utile	Entfernt	Infos bereits in detail_responsive #surface_usable

Municipality_merged	Entfernt	Lon Lat wird verwendet, da mehr vorhanden sind
Floor_merged		
Living_space_merged	Entfernt	Zum auffüllen in Space extracted gebraucht
Floor_space_merged	Behalten	Stockwerksfläche
Plot_area_merged	Entfernt	Wir zum auffüllen in Plot_area_unified gebraucht
Availability_merged	Behalten	Status der Immobilien
location_parsed	Entfernt	Daten wurden bereits extrahiert
title	Entfernt	Textuelle Daten
details	Entfernt	Extrahiert in anderen Spalten
address	Entfernt	Unvollständig und keine Verwendung
price	Entfernt	Ist zusammengefasst in price_cleaned
link	Entfernt	Extrahiert in type
details_structured	Entfernt	Extrahierte inforamtionen bereits in anderen Spalten
lat	Behalten	Standort des Haus
lon	Behalten	Standort des Haus
index	Entfernt	Reihenfolge des Scripts, hat keine informationen
ForestDensityL	Behalten	Nützliche Umgebungsinformation
ForestDensityM	Behalten	Nützliche Umgebungsinformation
ForestDensityS	Behalten	Nützliche Umgebungsinformation
Latitude	Entfernt	lat ist genauer
Locality	Entfernt	Nicht gebraucht da mit Postleitzahl gearbeitet wird
Longitude	Entfernt	lon ist detaillierter
NoisePollutionRailwayL	Behalten	Nützliche Umgebungsinformation
NoisePollutionRailwayM	Behalten	Nützliche Umgebungsinformation

NoisePollutionRailwayS	Behalten	Nützliche Umgebungsinformation
NoisePollutionRoadL	Behalten	Nützliche Umgebungsinformation
NoisePollutionRoadM	Behalten	Nützliche Umgebungsinformation
NoisePollutionRoadS	Behalten	Nützliche Umgebungsinformation
PopulationDensityL	Behalten	Nützliche Umgebungsinformation
PopulationDensityM	Behalten	Nützliche Umgebungsinformation
PopulationDensityS	Behalten	Nützliche Umgebungsinformation
RiversAndLakesL	Behalten	Nützliche Umgebungsinformation
RiversAndLakesM	Behalten	Nützliche Umgebungsinformation
RiversAndLakesS	Behalten	Nützliche Umgebungsinformation
WorkplaceDensityL	Behalten	Nützliche Umgebungsinformation
WorkplaceDensityM	Behalten	Nützliche Umgebungsinformation
WorkplaceDensityS	Behalten	Nützliche Umgebungsinformation
Zip	Entfernt	Postleitzahl wird gebraucht
distanceToTrainStation	Behalten	Nützliche Umgebungsinformation
gde_area_agriculture percentage	Behalten	Nützliche Umgebungsinformation
gde_area_forest percentage	Behalten	Nützliche Umgebungsinformation
gde_area_nonproductive percentage	Behalten	Nützliche Umgebungsinformation
gde_area_settlement percentage	Behalten	Nützliche Umgebungsinformation
gde_average_house_hold	Behalten	Nützliche Umgebungsinformation
gde_empty_apartments	Behalten	Nützliche Umgebungsinformation

gde_foreigners percentage	Behalten	Nützliche Umgebungsinformation
gde_new_homes_per_1000	Behalten	Nützliche Umgebungsinformation
gde_politics_bdp	Behalten	Nützliche Umgebungsinformation
gde_politics_cvp	Behalten	Nützliche Umgebungsinformation
gde_politics_evp	Behalten	Nützliche Umgebungsinformation
gde_politics_fdp	Behalten	Nützliche Umgebungsinformation
gde_politics_glp	Behalten	Nützliche Umgebungsinformation
gde_politics_gps	Behalten	Nützliche Umgebungsinformation
gde_politics_pda	Behalten	Nützliche Umgebungsinformation
gde_politics_rights	Behalten	Nützliche Umgebungsinformation
gde_politics_sp	Behalten	Nützliche Umgebungsinformation
gde_politics_svp	Behalten	Nützliche Umgebungsinformation
gde_pop_per_km2	Behalten	Nützliche Umgebungsinformation
gde_population	Behalten	Nützliche Umgebungsinformation
gde_private_apartments	Behalten	Nützliche Umgebungsinformation
gde_social_help_quota	Behalten	Nützliche Umgebungsinformation
gde_tax	Behalten	
gde_workers_sector1	Behalten	Nützliche Umgebungsinformation
gde_workers_sector2	Behalten	Nützliche Umgebungsinformation
gde_workers_sector3	Behalten	Nützliche Umgebungsinformation
gde_workers_total	Behalten	Nützliche Umgebungsinformation

price_cleaned	Behalten	Zielvariable
type	entfernt	Bessere Gruppierung in type_unified
Space extracted	behalten	Wohnfläche
rooms	Entfernt	Zum auffüllen in No. of rooms verwendet
plz_parsed	Entfernt	Anstelle von 4400 Postleitzahlen wurden nach lon und lat gruppiert
type_unified	Behalten	Gebäudetyp
Floor_unified	Entfernt	Stockwerkfläche
Plot_area_unified	Behalten	Grundstücksfläche
Living_area_unified	Entfernt	Zum auffüllen in Space extracted verwendet
provider	Entfernt	Keine nützliche Information
space	Entfernt	Zum auffüllen in Space extracted verwendet
price_s	Entfernt	Zum auffüllen in price_cleaned gebraucht
address_s	Entfernt	Zu wenige Observationen
No. of rooms:	Behalten	Anzahl Zimmer
Number of apartments:	Behalten	Anzahl Wohnungen
Surface living:	Entfernt	Zum auffüllen in Space extracted verwendet
Land area:	Entfernt	Gebraucht um detail responsive #surface_property aufzufüllen
Room height:	Entfernt	Zu wenige Einträge
Last refurbishment:	Behalten	Letzte Renovation
Year built:	Entfernt	Benutzt um Last refurbishment aufzufüllen
features	Behalten	Als dummy-Variable für zusätzliche Information
description_detailed	Entfernt	Wörtliche Informationen
Floor space:	Entfernt	Zu Wenige Observationen
Number of floors:	Behalten	Anzahl Stockwerke
Volume:	Entfernt	Zu Wenige Observationen
plz	Entfernt	Lon lat verwendet um nicht 4400 dummy-Variablen zu haben
Number of toilets:	Entfernt	Zu Wenige Observationen
Gross yield:	Entfernt 19	Möglicher Rückschlüsse auf den Preis
Minimum floor space:	Entfernt	Zu Wenige Observationen
space_cleaned	Entfernt	Zu Wenige Observationen

Tabelle 2.1: Informationen zu den Spalten

2.3.4 Feature engineering

Auffüllen

Einige Spalten enthalten das gleiche Feature, sind jedoch nicht zusammengeführt worden. Dies haben wir in der Data-Pipeline vorgenommen. In der Tabelle 2.1 ist dieser Prozess beschrieben.

KMeans-Clustering

Um besser mit den Location-Informationen umgehen zu können, haben wir die Spalten 'lon' und 'lat' mit einem K-Means Algorithmus geclustert.

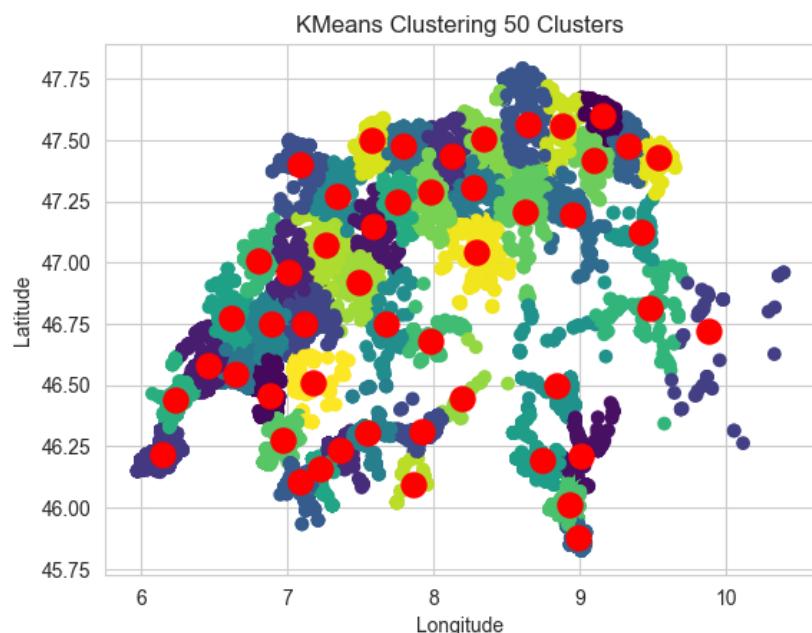


Abbildung 2.7: KMeans-Clustering

Aus diesen Clustern wurden dann Dummy-Variablen, welche die Modelle besser interpretieren konnten wie z.B. Postleitzahlen oder Longitude und Latitude.

[Hier](#) das dazugehörige Notebook.

BERT-Modell

Um aus dem 'detailed_description'-Feature effektiv nutzbare Informationen herauszuziehen, haben wir versucht mit [BERT](#), einem vortrainierten Enco-

der der Firma Google, ein Regressor zu schreiben welcher nur aufgrund dieser einen Spalte direkt den Immobilienpreis vorhersagt.

Die beiden Versuche sind [hier](#) zu finden. Der Trainingsprozess des Hugging Face Modells ist in diesem [WanDB-Projekt](#) zu finden.

Die lange Trainings- sowie Pipeplinelaufzeit haben jedoch dazu beigetragen, dass dieses daraus entstandene Feature nicht wirklich in die Runs miteingebaut wurden.

2.4 Imputation

2.4.1 Kein Imputer

Wenn ein Modell mit fehlenden Werten umgehen konnte, haben wir die Imputation nicht vorgenommen. Um unsere Pipeline zu 'überlisten' haben wir diesen Imputer verwendet:

```
imputer = SimpleImputer(strategy='constant', fill_value=None)
```

2.4.2 Simple Imputer

Um einfachere Modelle oder die Pipeline zu testen, haben wir einen einfachen und schnellen Imputer verwendet. Der SimpleImputer von sklearn hat verschiedene 'Strategien' wie dieser die Daten Imputieren soll. Wir haben meist 'mean' oder 'median' verwendet.

2.4.3 KNN-Imputer

Der KNNImputer von sklearn hilft uns, fehlende Werte in unseren Daten zu ersetzen. Er nutzt dafür die Methode der k-nächsten Nachbarn. Das bedeutet: Für jede fehlende Zahl schaut der Imputer auf die Werte der nächsten k ähnlichen Datenpunkte und berechnet daraus einen Durchschnitt. Mit dem Parameter n_neighbors können wir einstellen, wie viele Nachbarn berücksichtigt werden sollen. Diese Methode funktioniert gut, wenn unsere Daten Ähnlichkeiten in kleinen Gruppen aufweisen. Wir haben meistens ein k zwischen 5 und 10 gewählt.

Dieser Imputer ist jedoch sehr ressourcenintensiv und kann je nach Anzahl der Features bis zu 5 Minuten rechnen.

2.5 Modelle Regression

2.5.1 Metrik

Der Mean Absolute Percentage Error (MAPE) ist eine Metrik, die die durchschnittliche prozentuale Abweichung zwischen vorhergesagten und tatsächlichen Werten misst. Sie wird oft verwendet, um die Genauigkeit von Prognosen zu bewerten, insbesondere in Preisvorhersagen.

$$MAPE = 100 - \frac{1}{n} \sum_{t=1}^n \left| \frac{A_t - F_t}{A_t} \right|$$

Vorteile

Der MAPE ist leicht zu verstehen und ermöglicht eine intuitive Interpretation, da sie in Prozent ausgedrückt wird. Dies macht es einfach, die Abweichung unabhängig von der Skala der Werte zu bewerten. Außerdem ist der MAPE ein Standardmaß für Prognosen in Bereichen wie der Preisvorhersage, da es eine verständliche und vergleichbare Einschätzung der Modellleistung liefert.

Nachteile

Trotz seiner Einfachheit hat MAPE mehrere Schwächen. Erstens führt die Division durch kleine oder null Werte zu Problemen, da sie unendlich grosse Fehler erzeugen kann. Zweitens ist der MAPE asymmetrisch: Sie bestraft negative Fehler stärker als positive Fehler. Dies kann dazu führen, dass Modelle bevorzugt werden, die systematisch zu niedrige Vorhersagen machen. Zudem hat der MAPE keinen oberen Grenzwert für grosse positive Fehler, was die Vergleichbarkeit erschwert.

2.5.2 Lineare Regression

Einfache lineare Regression

Im Rahmen dieser Analyse wurde ein einfaches lineares Regressionsmodell entwickelt, um den bereinigten Immobilienpreis (*price_cleaned*) basierend auf der Wohnfläche (*Space extracted*) vorherzusagen. Ziel war es, die Modellannahmen zu überprüfen, mögliche Verbesserungen durch Variablentransformationen zu identifizieren und die Vorhersagegenauigkeit mithilfe geeigneter Metriken zu bewerten.

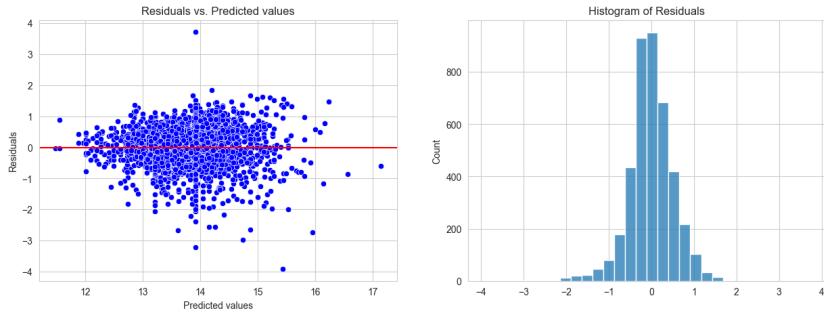


Abbildung 2.8: Residuenanalyse des einfachen linearen Modells

Die Analyse ergab, dass das Modell die besten Ergebnisse liefert, wenn sowohl der Preis als auch die Wohnfläche logarithmisch transformiert werden. Diese Transformation ist sinnvoll, da beide Variablen viele grosse Ausprägungen aufweisen, die durch die Log-Transformation näher an eine Normalverteilung herangeführt werden. Wie in Abbildung 2.8 zu erkennen ist, erscheinen die Residuen annähernd normalverteilt.

Auffällig ist jedoch, dass einige wenige Vorhersagen stark von den tatsächlichen Werten abweichen. Der berechnete Mean Absolute Percentage Error (MAPE) lag bei 47 %, was für ein einfaches Regressionsmodell als solide Leistung betrachtet werden kann.

Bestes Lineares-Modell

Das Lineare Modell wurde mit Lasso trainiert, um Overfitting zu verhindern. Die beste Regularisierung war 0.01. Wie beim einfachen Linearen-Modell wurde die Zeilvariable (price_cleaned) und die Wohnfläche Log-transformiert. Die fehlende Werte wurden mit knn-Imputing aufgefüllt, da dieser bei den Test die besten Ergebnisse lieferte.

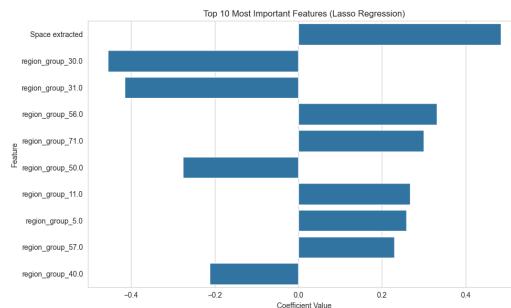


Abbildung 2.9: Top-10 Koeffizienten Lasso-Regression

Das Ergebnis zeigte, dass wir durch die Integration zusätzlicher Features das Modell um 10% in Bezug auf den MAPE (Mean Absolute Percentage Error) verbessern konnten. Der Test-MAPE erreichte schliesslich einen Wert von 38%. Diese Verbesserung lässt sich insbesondere anhand der Koeffizienten auf die Bedeutung der Standorte zurückführen.

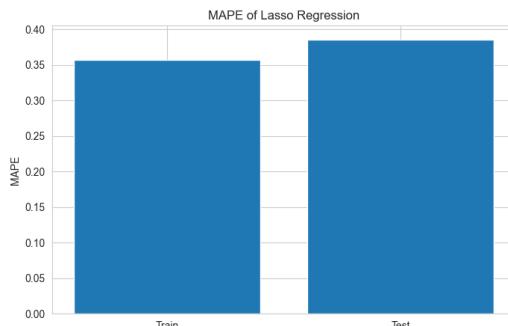


Abbildung 2.10:

2.5.3 Splines

Aus dem Modul [STL](#) haben wir von Splines erfahren und wollten diese ausprobieren. Wir versuchen den Ansatz mit einem General Additive Model (GAM).

Features

Wir haben 116 Features verwendet um dieses Modell zu trainieren. Dazu gehören die Cluster-Groups sowie type_unified und die dummyfied features.

Modell

Wir haben das LinearGAM von [pyGAM](#) verwendet. Die Hyperparameter haben wir mit Gridsearch getunet. Die Additionsterme haben wir auf 'auto' gelassen. Die Modellgrösse beträgt 837.7 Effective Degrees of Freedom (EDoF).

Ergebnisse

Feature-Analyse Der Vorteil eines GAMs ist, das es einiger Massen interpretierbar bleibt. Deshalb können wir den Einfluss einzelner Features auf das Ergebnis inspizieren.

Hier können wir zum Beispiel sehen, dass wenig Wohnfläche, einen negativen Einfluss auf den Immobilienpreis hat.

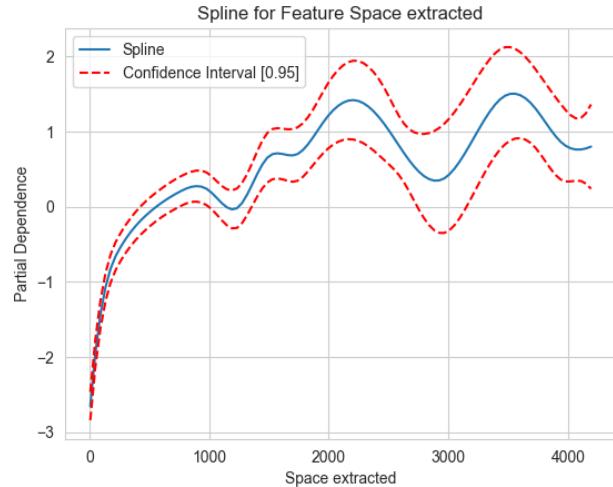


Abbildung 2.11: Spline-Regression für das Feature Space extracted

Spannend ist noch das Feature Year built. Das 95%-Konfidenzintervall (Rote Linien) ist in frühen Jahren sehr weit von der Vorhersage (Blaue Linie) weg. Dies liegt daran, dass wir viel mehr Daten von neueren Immobilien haben, wie von alten.

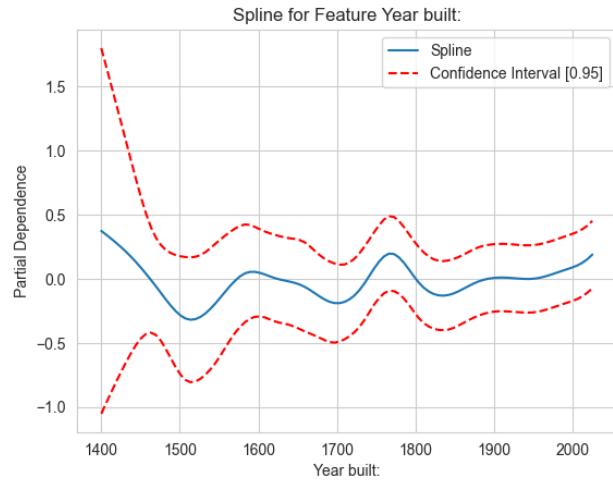


Abbildung 2.12: Spline-Regression für das Feature year built

Vorhersagen Die Ergebnisse sind nicht umwerfend, jedoch besser als Erwartet.

Train-MAPE	Test-MAPE
27.81	26.49

Tabelle 2.2: Ergebnisse Train-Test-MAPE

Zu sehen ist, dass niedrige Immobilienpreise einen sehr grossen Einfluss auf den MAPE haben. Die teuren Immobilien haben kaum einen Fehler über 100.

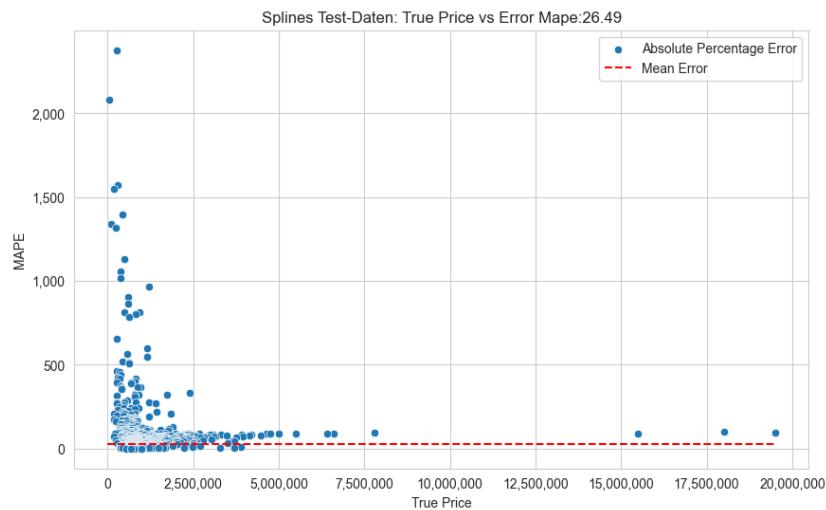


Abbildung 2.13: MAPE vs. Preis

Fazit

Das Splines-Modell ist ein interessanter Ansatz für ein interpretierbares Modell. Das Ergebniss ist ebenfalls gut. Jedoch haben wir mit unserem Splines-Modell auf dem Kaggle-Datensatz nur ein MAPE von 51% erreicht. Deswegen haben wir diesen Ansatz nicht weiter verfolgt. Weitere Experimente und Analysen sind [hier](#) zu finden.

2.5.4 Random Forest

Das Hyperparametertuning wurde mithilfe eines Bayes-Optimierers durchgeführt. Besonders hervorzuheben ist dabei der Parameter ccp-alpha, der das Modell effektiv vor Overfitting schützte. Bei der Optimierung mit wenigen Parametern schien zunächst die max-depth den grössten Einfluss auf die Modelleistung zu haben. Allerdings zeigte die Kreuzvalidierung, dass die Ergebnisse dabei stark schwankten.

Für das RandomForest-Modell stellte sich die Wohnfläche als das wichtigste Feature heraus. Interessanterweise wurde die Nutzfläche als zweitwichtigste Variable identifiziert, was auf einen unerwartet starken Einfluss dieser Eigenschaft auf das Modell hindeutet.

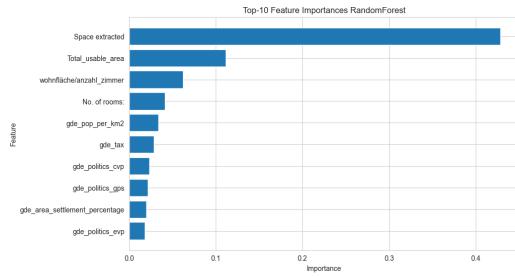


Abbildung 2.14: Top-10 wichtigste Features RandomForest

Das Ergebnis des Random Forest Modells war sehr gut, der Test-MAPE betrug 28%. Durch die gezielte Anpassung des Parameters `ccp_alpha` konnte das Modell effektiv vor Overfitting geschützt werden, was zu einer besseren Generalisierungsfähigkeit führte. Darüber hinaus trug die Log-Transformation der Zielvariable massgeblich zur Leistungssteigerung des Modells bei. Insgesamt führte die Kombination aus Hyperparametertuning und der Log-Transformation zu einer signifikanten Verbesserung der Modellgüte.

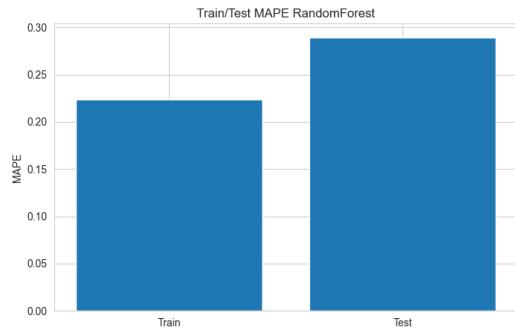


Abbildung 2.15: Bestes Resultat-Mape RandomForest

2.5.5 Hist-Gradient-Boosting

Das Hyperparametertuning des HistGradientBoosting-Modells wurde ebenfalls mit dem Bayes-Optimizer durchgeführt. Fehlende Werte mussten nicht ersetzt und kategoriale Variablen nicht umgewandelt werden, da das HistGradientBoosting-Modell diese Aufgaben eigenständig übernimmt. Auch hier wurde die Zielvariable log-transformiert, da dies die Modelleistung erheblich verbesserte.

Auffällig ist, dass der HistGradientBoosting-Algorithmus die Regionengruppen als zweitwichtigstes Merkmal identifiziert. Das wichtigste Merkmal war, wie bereits zuvor, die Wohnfläche.

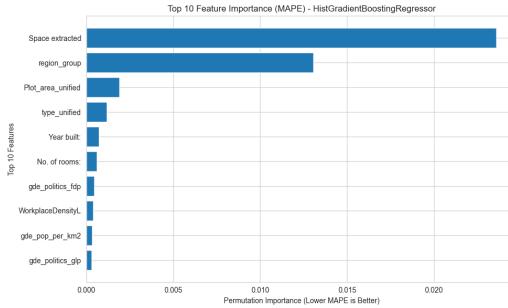


Abbildung 2.16: Top-10 wichtigste Features des HistGradientBoosting-Modells

Das Ergebnis des HistGradientBoosting-Modells überraschte mich sehr. Der Test-MAPE erreichte einen bemerkenswert niedrigen Wert von 23%. Besonders beeindruckend war jedoch die Trainingsgeschwindigkeit des Modells. Es trainierte in aussergewöhnlich kurzer Zeit und erzielte dabei einen sehr guten MAPE-Wert.

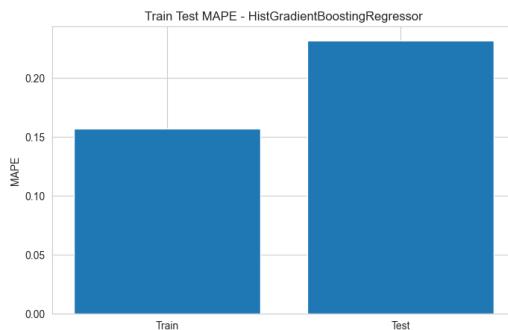


Abbildung 2.17: Bestes Resultat-Mape HistGradientBoosting

2.5.6 K-Nearest Neighbors

Das KNN-Modell ist sehr einfach zu implementieren und zu tunen. Jedoch ist es stark anfällig für Overfitting. Dies, da die Vorhersage auf Basis der Trainingsdaten gemacht werden.

Features

Wir haben das Modell einmal mit vielen (149) und wenigen (77 davon sind 50 Cluster Groups und 20 type_unified) Features ausprobiert.

Modell

Wir haben uns für eine Anzahl von 10 Nachbarn entschieden. Ebenfalls haben wir die Gewichte nach Distanz und einem p (Exponent der Distanz) von 1 gewählt.

Ergebnisse

Das Modell ist beide Modelle Overfitten sehr stark. Da die KNN-Modelle keine Regularisierungsmechanismen haben, ist es schwierig aus diesem Overfit herauszukommen.

Train-MAPE	Test-MAPE
0.317	37.86

Tabelle 2.3: Ergebnisse Train-Test-MAPE

Auch bei diesem Modell befindet sich der grösste Fehler bei den günstigen Immobilien.

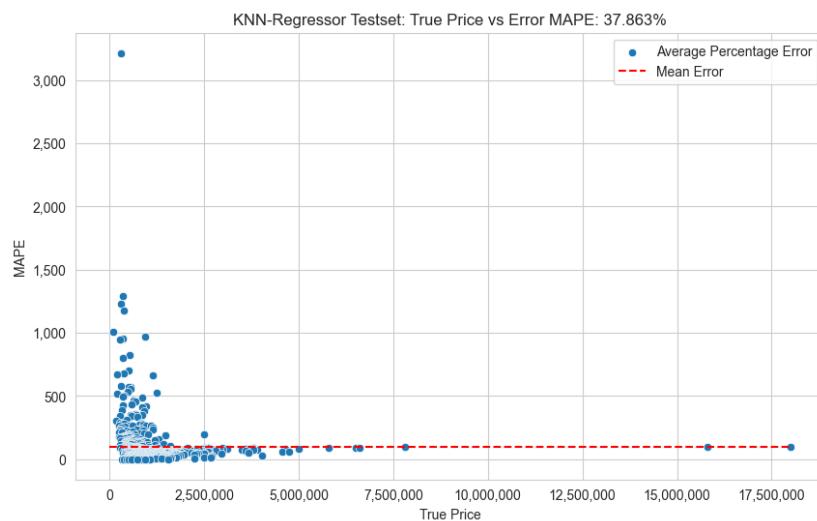


Abbildung 2.18: MAPE vs. Preis

Fazit

Der KNN-Regressor funktioniert erstaunlich gut. Jedoch ist er sehr stark von Overfitting betroffen, gegen welches wir kein Mittel gefunden haben. Der MAPE auf dem Kaggle-Datensatz war 46.53%. Da es nur wenig Raum für Optimierung bietet, haben wir diesen Ansatz nicht weiter verfolgt. Weitere Experimente und Informationen sind [hier](#) zu finden.

2.5.7 XGBoost

Dieses Modell ist laut Recherchen !!!QUELLE!!! das vielversprechendste Modell um Immobilienpreise zu bestimmen. Deswegen haben wir sehr viel Zeit mit diesem Modell verbracht.

Features

Wir haben eine grosse Bandbreite an Features und Feature-Kombinationen ausprobiert. Schlussendlich haben wir festgestellt, dass mehr Features das Modell grundsätzlich besser machen oder zumindest nicht schlechter.

Da beim MAPE die tiefen Werte einen grösseren Fehler verursachen, haben wir die Daten beim Training gewichtet. Daten welche einen Immobilienpreis unter 50'000 Franken haben, wurden 50 mal mehr gewichtet wie jene über 1.2 Millionen Franken. Die genaue Stückelung kann im Notebook nachgelesen werden.

Modell

Auch hier haben wir sehr viele verschiedene Hyperparameter ausprobiert. Im unteren Teil des [XGBoost-Notebooks](#) ist ein Grid-Search mit 01:30:00 Stunden Laufzeit zu finden.

Grundsätzlich overfittet XGBoost sehr stark. Deswegen haben wir eine starke Regularisierung verwendet. Für uns wichtige Regularisierungsparameter sind:

- `reg_lambda = 1.5` (L1-Regularisierung)
- `reg_alpha = 8` (L2-Regularisierung)
- `min_split_loss = 5` (Mindestgewicht für einen Split)
- `min_child_weight = 20` (Mindestgewicht für eine neue Node)

- `subsample` = 0.9 (% von Daten welche pro Run fürs Training verwendet werden)
- `max_delta_step` = 3 (Maximale Schrittänge pro Trainingsschritt)
- `colsample_bylevel` = 0.8 (% von Features welche pro Run fürs Training von einem Layer verwendet werden)
- `colsample_bytree` = 0.8 (% von Features welche pro Run fürs Training von einem Baum verwendet werden)
- `colsample_bynode` = 0.8 (% von Features welche pro Run fürs Training von einer Node verwendet werden)

sowie eine tiefe `learning_rate` mit 0.005 und eine `max_depth` von 9 haben gegen das Overfitting geholfen.

Ergebnisse

Train-MAPE	Test-MAPE
22.03	27.38

Tabelle 2.4: Ergebnisse Train-Test-MAPE

Das XGBoost-Modell ist auch bei uns das best-performancste Modell. Jedoch haben wir mit starkem overfitting zu kämpfen.

Auch in diesem Fall sind die günstigen Immobilien das Problem. Und dies obwohl wir beim Trainieren die Gewichtung dieser extra deutlich erhöht haben.

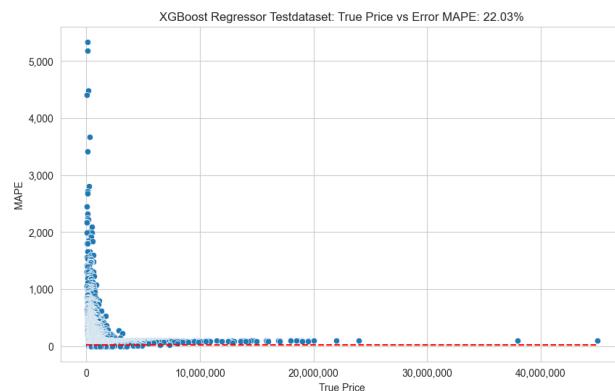


Abbildung 2.19: XGBoost Fehler pro Preis

Feature Importance Bei XGBoost kann man eine Feature Importance extrahieren. Diese entspricht in etwa den Erwartungen und weicht nicht stark von derer anderer Modelle ab. Überraschend ist die Wichtigkeit von bestimmten region_groups. Dies könnte beispielsweise daran liegen das bestimmte Cluster genau auf einer teureren Stadt liegen.

Merkmal	Feature Importance
Space extracted	0.05
region_group_16	0.04
No. of rooms:	0.03
region_group_19	0.03
type_unified_studio	0.02
type_unified_villa	0.02
gde_pop_per_km2	0.02
type_unified_farmhouse	0.02
type_unified_detached-house	0.02
region_group_43	0.02

Tabelle 2.5: Wichtige Merkmale und deren Bedeutung

Weitere Experimente und Informationen sind [hier](#) zu finden.

Fazit

Das ist unser bestes Modell mit einem MAPE von 33.48 auf dem Private-Kaggle-Datensatz. Leider haben wir den starken Overfit nicht gut beseitigen können.

2.5.8 MLP

Da Tabellarische sowie ein grosser Datensatz vorhanden ist, haben wir ein Multy-Layer-Perceptron ausprobiert. Alle Runs sind auf [WanDB](#) zu finden.

Features

Da dies unser erstes Modell war, hat sich die Datenpipeline über die Zeit stark verändert. Die Runs sind also nicht alle auf dem gleichen Datenset. Ab etwa der Hälfte der Runs haben wir die verwendeten Spalten mitgeloggt.

Modell

Für die Modelle haben wir die Library Pytorch verwendet. Mit dieser ist die Implementation deutlich einfacher. Ebenfalls wurde ein Teil des Codes für

dieses Modell (hauptsächlich Trainingsloop) wurden aus der [DeepLearning Minichallenge](#) übernommen.

Das beste Modell hat fünf Fully-Connected Layers mit einer Batch-Size von 16 und einer Learning Rate von 0.0001.

Ergebnisse

Train-MAPE	Test-MAPE
18.82	20.49

Tabelle 2.6: Ergebnisse Train-Test-MAPE

[Report](#) der besten MLP-Runs.

Die Performance des MLP war erstaunlich gut. Jedoch ist stark davon auszugehen, dass es sich dabei um Overfitting handelt. Das Modell könnte man beispielsweise mit L1, L2 oder Dropout regularisieren.

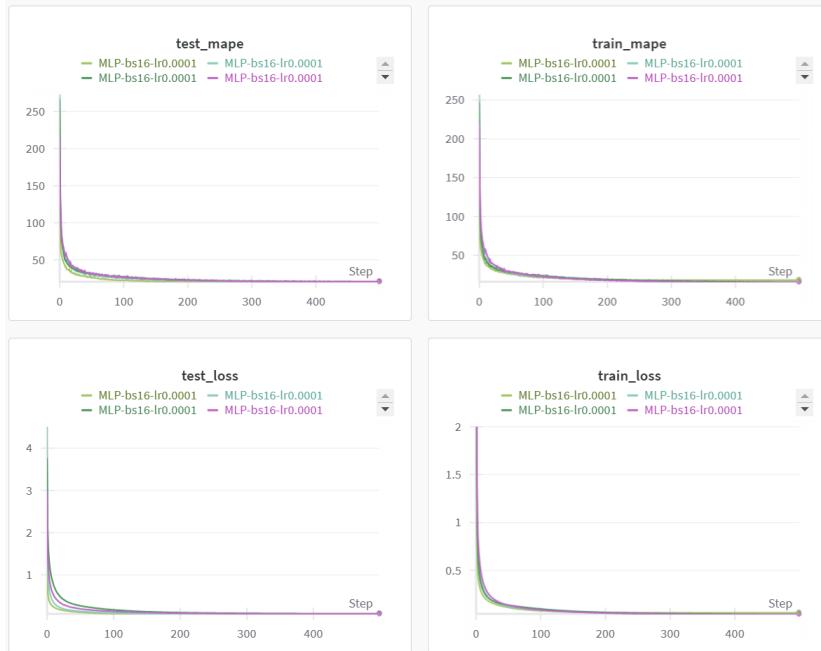


Abbildung 2.20: MLP Trainingskurven

Die Trainingszeit war bei diesem Modell sehr lange.
Weitere Experimente und Informationen sind [hier](#) zu finden.

Fazit

Der MAPE auf dem Kaggle-Datensatz war mit 63.24% sehr schlecht und bestätigt die Vermutung des starken Overfittings. Nach Absprache mit Fernando Benites haben wir uns nicht weiter in dieses Modell vertieft.

2.6 Modelle zur Klassifikation

Unsere Aufgabe bestand darin, den Typ der Gebäude vorherzusagen. Da es zwei Spalten `Type` und `Type_unified` gibt, die den Haustyp beschreiben, haben wir uns entschieden, `Type_unified` als Zielvariable zu verwenden. Der Grund für diese Entscheidung war, dass es bei `Type` viele Gebäudetypen mit weniger als fünf Einträgen gab, was den Vorhersageprozess erschwert hätte. Mit `Type_unified` sagen wir 21 verschiedene Haustypen voraus.

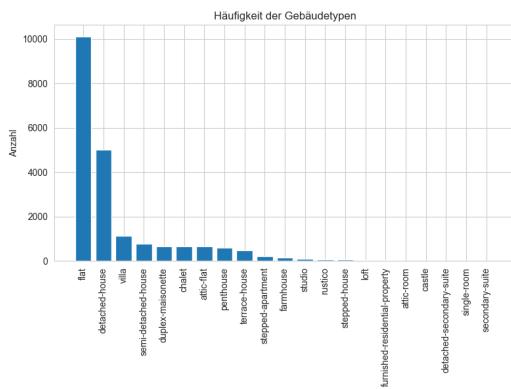


Abbildung 2.21: Wichtigste Features der logistischen Regression

2.6.1 Preprocessing

Bei der Zielvariablen wurden folgende Transformationen durchgeführt:

- `attic-room` wurde `attic-flat` zugewiesen.
- `castle` wurde zu `farmhouse` zusammengefasst.
- `detached-secondary-suite` wurde zu `detached-house` umgewandelt.
- `single-room` wurde zu `studio`.
- `secondary-suite` wurde zu `duplex-maisonette`.

Diese Anpassungen wurden vorgenommen, da diese Kategorien jeweils weniger als zehn Einträge hatten, was die Vorhersagegenauigkeit negativ beeinflusst hätte.

Zusätzlich wurde die Variable `price` auf Werte beschränkt, da niedrigere Preise als unlogisch betrachtet wurden. Ebenso wurden Wohnflächen unter 5 Quadratmetern aus dem Datensatz entfernt, da diese als fehlerhafte Einträge angesehen wurden.

2.6.2 Metrik: Gewichteter Recall

Für die Bewertung unseres Klassifikationsmodells haben wir den **gewichteten Recall** gewählt. Diese Metrik ist besonders geeignet, wenn die Klassenverteilung unausgeglichen ist, was in unserem Fall zutraf. Da einige Haustypen seltener vertreten sind, war es entscheidend, dass unser Modell auch diese Klassen zuverlässig erkennt. Wie man in der Abbildung 2.21 sieht.

Der Recall für eine einzelne Klasse berechnet sich nach folgender Formel:

$$\text{Recall}_i = \frac{\text{True Positives}_i}{\text{True Positives}_i + \text{False Negatives}_i} \quad (2.1)$$

Der **gewichtete Recall** berücksichtigt die Häufigkeit jeder Klasse und berechnet sich nach folgender Formel:

$$\text{Gewichteter Recall} = \sum_{i=1}^n \frac{|\text{Klasse}_i|}{N} \cdot \text{Recall}_i \quad (2.2)$$

Dabei ist $|\text{Klasse}_i|$ die Anzahl der Instanzen der Klasse i , N die Gesamtanzahl aller Instanzen und Recall_i der Recall für Klasse i .

Ein hoher gewichteter Recall-Wert zeigt, dass das Modell in der Lage ist, alle Klassen – insbesondere die selteneren – angemessen zu erkennen. Dies ist bei einer unbalancierten Datenverteilung besonders wichtig, da sonst Minderheitsklassen leicht übersehen werden könnten.

2.6.3 Logistische Regression

Die logistische Regression wurde auf einem Datensatz mit 59 Spalten und 20786 Zeilen trainiert. Zwei dieser Spalten (*Availability* und *region_group*) wurden in Dummy-Variablen umgewandelt. Fehlende Werte wurden mithilfe eines KNN-Imputers aufgefüllt, da die logistische Regression nicht mit fehlenden Werten umgehen kann. Die numerischen Werte wurden mit einem StandardScaler normalisiert, wodurch sie einen Wertebereich von etwa und einen Mittelwert von 0 erhielten.

Die wichtigsten Einflussfaktoren im Modell waren, wie bereits bei der Vorhersage des Preises, vor allem die Wohnfläche, die sich als das bedeutendste Feature erwies. Auffällig war, dass das Stockwerk des Gebäudes als zweitwichtigstes Merkmal identifiziert wurde. Dies erscheint plausibel, da Wohnungen in verschiedenen Stockwerken unterschiedliche Eigenschaften und damit Einfluss auf die Zielvariable haben können. Die Verteilung der wichtigsten Merkmale ist in Abbildung 2.22 dargestellt.

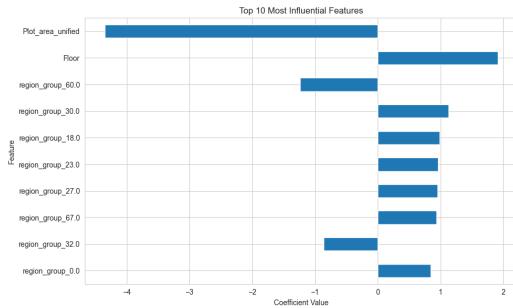


Abbildung 2.22: Wichtigste Features der logistischen Regression

Das Modell zeigte eine gute Generalisierungsfähigkeit, da die Metriken für Trainings- und Testdaten eng beieinander lagen. Besonders wichtig war für uns der Recall, der auf den Testdaten bei 0,68 lag, wie auf bei 2.23 dargestellt. Zur Verbesserung dieses Ergebnisses setzten wir SMOTE (Synthetic Minority Over-sampling Technique) zum Upsampling ein. Obwohl der SMOTE-Upsampler auf den Trainingsdaten vielversprechend erschien, zeigte er auf den Testdaten eine deutlich schlechtere Leistung mit einem Recall von nur noch 0,38. Dieses Ergebnis erklären wir uns damit, dass der SMOTE-Upsampler aufgrund einer zu kleinen Ausgangsstichprobe keine qualitativ hochwertigen synthetischen Daten erzeugen konnte.

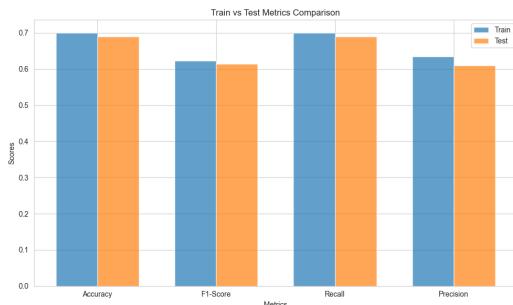


Abbildung 2.23: Bestes Resultat Logistische-Regression

2.6.4 Desischon trees

Kevin

2.6.5 Random Forest

Wie bei der logistischen Regression wurde das Random-Forest-Modell auf einem Datensatz mit 59 Spalten und 20.786 Zeilen trainiert. Zwei dieser Spalten

(*Availability* und *region_group*) wurden in Dummy-Variablen umgewandelt, um kategoriale Daten korrekt zu verarbeiten. Fehlende Werte im Datensatz wurden mithilfe des *knn_Imputers* aufgefüllt.

Zur Optimierung der Hyperparameter kam der Bayesian Optimizer zum Einsatz. Dieser verwendet ein probabilistisches Modell zur Approximation der Zielfunktion und bestimmt auf Basis dieser Schätzung die nächste zu testende Parameterkombination. Durch die Nutzung einer sogenannten Akquisitionsfunktion kann der Bayesian Optimizer gezielt und effizient optimale Hyperparameter finden, was insbesondere bei rechenintensiven Modellen wie dem Random Forest von Vorteil ist.

Die Analyse der Feature-Importanz (siehe Abbildung 2.24) zeigte, dass die Grundstücksfläche das wichtigste Merkmal für den Random Forest darstellt, gefolgt vom Stockwerk. Diese beiden Merkmale sind besonders interessant, da sie wahrscheinlich den Unterschied zwischen einer Wohnung und einem Haus widerspiegeln. Überraschend rangiert der Preis erst an vierter Stelle, obwohl erwartet wurde, dass dieser einen grösseren Einfluss auf die Vorhersagen hat.

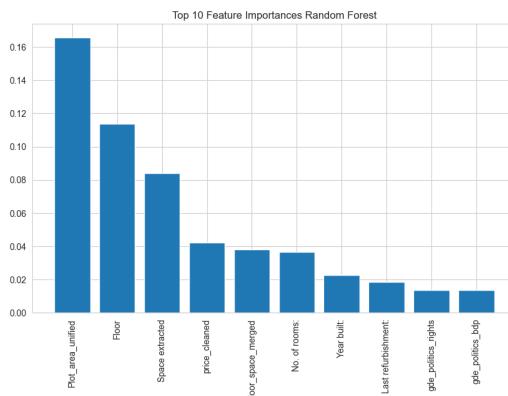


Abbildung 2.24: Wichtigste Features des Random Forest

Mit dem Random-Forest-Modell wurde ein Recall von 0,74 erreicht, was eine deutliche Verbesserung gegenüber dem Modell der logistischen Regression darstellt. Obwohl der Random Forest tendenziell etwas schlechter generalisiert, liefert er insgesamt bessere Ergebnisse über alle Bewertungsmaetriken hinweg.

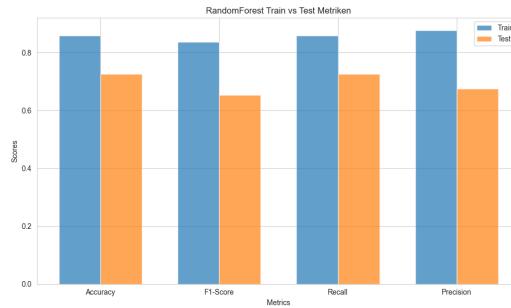


Abbildung 2.25: Bestes Ergebnis des Random Forest

Um das Ergebnis weiter zu verbessern, wurde auch hier versucht, mit dem SMOTE-Oversampler bessere Resultate zu erzielen. Wie bereits beim logistischen Modell war dies nicht erfolgreich, was auf die geringe Anzahl an Stichproben in bestimmten Kategorien zurückgeführt wird.

2.6.6 XGBoost-Classifier

Das Modell stammt aus der gleichen Library wie der XGBoost-Regressor welchen wir für die Regression des Immobilienpreises verwendet haben. Dieser funktioniert auch für unsere Klassifikationsaufgabe relativ gut.

Features

Wir haben für dieses Modell sehr viele Features (84) verwendet. Die Zielvariable wurde wie oben bereits beschrieben angepasst.

Modell

Wir haben ähnliche Parameter wie beim XGBoost-Regressor verwendet. Jedoch haben wir die Regularisierung etwas weniger stark gemacht.

Ergebnisse

Train-Recall	Test-Recall
0.773	0.691

Tabelle 2.7: Ergebnisse Train-Test-Recall

Das Modell Schnitt nicht so gut ab wie erwartet. Vor allem im Vergleich mit dem Random-Forest-Modell ist der XGBoost-Classifier leider nicht besser. Wir haben nicht herausgefunden woran dies liegen könnte.

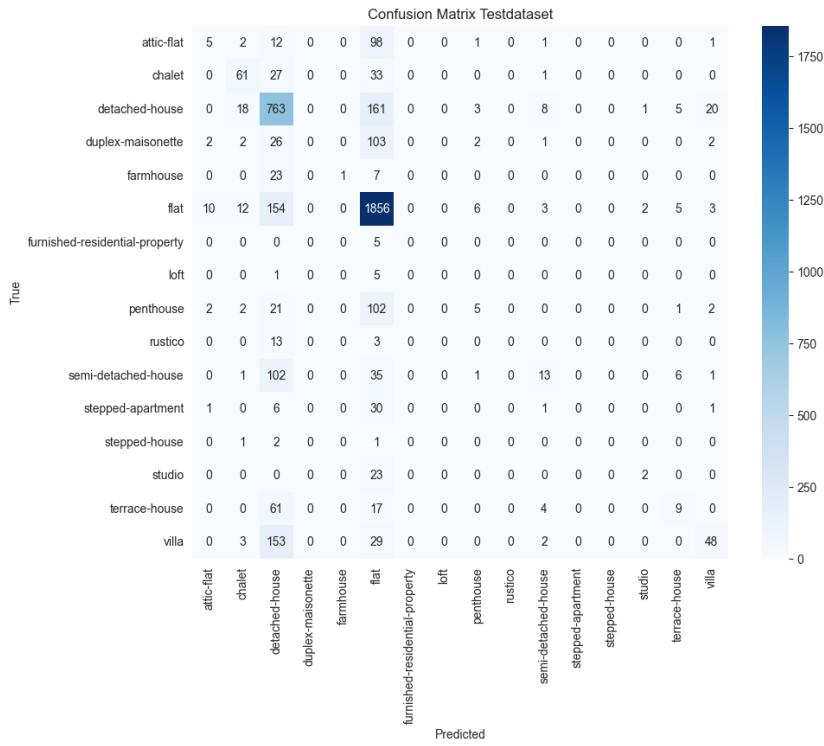


Abbildung 2.26: XGBoost Classifier Confusion Matrix

Das Modell hat viele der Immobilien einfach als 'flat' oder 'detached-house' vorhergesagt.

2.6.7 Zusammenfassung

Unsere Analysen zeigen, dass die stark unausgeglichenen Gebäudetypen die Vorhersage erschweren. Zwar haben wir versucht, durch SMOTE-Oversampling die Minderheitsklassen zu verstärken, doch führte dies zu schlechteren Ergebnissen. Eine mögliche Ursache dafür ist die sehr geringe Anzahl an Ausgangsdaten für bestimmte Typen, sodass keine qualitativ hochwertigen synthetischen Stichproben erzeugt werden konnten. Nichtsdestotrotz lieferte das Random-Forest-Modell ein überzeugendes Resultat mit einem gewichteten Recall von 0,74 auf dem Testdatensatz. Um die Vorhersagen weiter zu verbessern, könnten fortgeschrittene Strategien zur Behandlung fehlender Werte zum Einsatz kommen oder eine größere Zusammenfassung seltener Haustypen vorgenommen werden. Auf diese Weise liesse sich die Verteilung der Zielvariable ausbalancieren und die Modellleistung insbesondere für kleinere Klassen steigern.

2.7 Webauftritt

Im Rahmen der Entwicklung innovativer digitaler Lösungen wurde eine benutzerfreundliche Webanwendung zur Prognose von Immobilienpreisen konzipiert. Ziel dieser Anwendung ist es, sowohl privaten Käufern als auch Investoren einen einfachen Zugang zu präzisen Preisvorhersagen zu ermöglichen. Der Fokus liegt auf einer intuitiven Bedienbarkeit, die keine speziellen Fachkenntnisse voraussetzt.



Abbildung 2.27: Webserver

2.7.1 Voraussetzungen

Für den Betrieb der Webanwendung sind folgende technische Voraussetzungen notwendig:

1. **Python 3.x** muss auf dem System installiert sein.
2. Installation der erforderlichen Python-Bibliotheken:

```
pip install -r requirements.txt
```

3. Bereitstellung der folgenden Dateien:
 - `home.html`: Benutzeroberfläche für die Dateneingabe.
 - `zip_to_lat_lon(zip_code)`: Funktion zur Umwandlung der Postleitzahl in geografische Koordinaten.
 - `preprocess_and_predict(input_data)`: Verantwortlich für die Datenverarbeitung und Vorhersage.
 - Flask-Routen: Die Hauptseite der Anwendung wird über die Route `/` gesteuert.

2.7.2 Bedienung der Startseite

Nach dem Öffnen der Anwendung im Browser erscheint die Startseite mit einem Eingabeformular. Dort können Nutzer folgende Daten eingeben:

- **Postleitzahl** (Postleitzahl des Standorts)
- **Wohnfläche** (Wohnfläche in Quadratmetern)
- **Stockwerk** (Etage des Objekts)
- **Nutzfläche** (Nutzfläche in Quadratmetern)
- **Anzahl der Stockwerke** (Gesamtanzahl der Etagen im Gebäude)
- **Grundstücksfläche** (Grundstücksgrösse in Quadratmetern)
- **Anzahl der Zimmer** (Zimmeranzahl der Immobilie)
- **Gebäude Typ** (z.B. Penthouse, Villa, Loft, etc.)

2.7.3 Ablauf des Vorhersageprozesses

1. **Dateneingabe:** Der Nutzer füllt das Formular aus und übermittelt die Daten.
2. **Umwandlung der Postleitzahl:** Die eingegebene Postleitzahl wird in Längen- und Breitengrad konvertiert.
3. **Regionale Gruppierung:** Mittels KMeans-Clustering erfolgt die Zuordnung zu einer Region.
4. **Datenaufbereitung:** Die Eingabedaten werden durch eine Vorverarbeitungs-Pipeline vorbereitet.
5. **Vorhersageberechnung:** Das trainierte Modell erstellt eine Preisprognose.
6. **Anzeige des Ergebnisses:** Die Prognose wird auf der Webseite ausgegeben.

2.7.4 Installation

1. **Ordner herunterladen oder navigieren:**

Laden Sie den Ordner **Webserver** herunter oder navigieren Sie im Terminal in das entsprechende Verzeichnis.

2. **Python-Bibliotheken:**

Siehe Abschnitt Voraussetzungen[Unterabschnitt 2.7.1](#)

3. **Applikation starten:**

Starten Sie die Anwendung mit folgendem Befehl im Terminal:

```
python app.py
```

4. **Webseite öffnen:**

Öffnen Sie die angegebene URL oder rufen Sie die Anwendung im Browser unter <http://localhost:5000> auf.

2.8 Projektstruktur

2.8.1 Projektmanagement

Zusammenarbeit

Da wir nur zu Zweit waren, haben wir unser Projektmanagement sehr schlank gehalten. Da wir beide jeden Tag vor Ort waren, konnten wir regelmässige Besprechungen abhalten. Diese fanden etwa zwei mal in der Woche statt. Dabei wurden die Ergebnisse der letzten Tage und die Ziele und Aufgaben für die nächsten Tage besprochen.

Kommunikation

Die Kommunikation fand hauptsächlich in mündlicher Form statt. Alternativ auf einem Kurznachrichtendienst.

Aufgabenmanagement

Als Haupttool für unser Aufgabenmanagement haben wir github-Issues verwendet. Mit diesem Tool konnten wir unsere Aufgaben planen und eine gute Übersicht behalten.

Fazit

Die Zusammenarbeit empfanden wir beide als sehr angenehm. Die Zusammenarbeit vor Ort hat vieles deutlich vereinfacht/effizienter gemacht.

2.8.2 Versionskontrolle

Als Versionskontrolle haben wir [github](#) verwendet. Dabei haben wir Feature-Branching inklusive Code-Reviews, Unit-Tests und Linter-Test angewendet.

Unit-Tests

Für die Daten-Pipeline sowie die Webapp haben wir Unit-Tests geschrieben. Diese Kontrollieren dessen Funktionen.

Die Unit-Tests sind Teil der CI/CD-Pipeline und werden via github-Actions bei jedem Push ausgeführt. Ein Branch kann nur bei erfolgreich bestanden Unit-Tests in den Main-Branch gemitgt werden.

Linter-Tests

Um die Naming- sowie Coding-Standards einzuhalten, haben wir Linter-Tests eingeführt. Diese sind ebenfalls Teil der CI/CD-Pipeline. Sie werden bei jedem Push automatisch ausgeführt und müssen für das Mergen in den Main bestanden werden.

Weights Bias

Die Ergebnisse des MLP-Modells haben wir auf [WanDB](#) geloggt.

2.9 Fazit

Die Analyse und Modellierung des Immobilien-Datensatzes von Immoscout24 und Homegate verdeutlichte die Herausforderungen bei der Bereinigung realer Daten. Zudem wurde deutlich, wie komplex der Aufbau einer flexiblen Datenpipeline ist, die verschiedene Modellarten unterstützt. Lineare Regressionsmodelle erfordern beispielsweise die Erstellung von Dummy-Variablen, während baumbasierte Modelle wie Entscheidungsbäume sowohl mit One-Hot-Encoding als auch teilweise mit fehlenden Werten umgehen können.

Erstaunlicherweise lieferte bereits das einfache lineare Modell gute Ergebnisse, was uns optimistisch stimmte, auch mit komplexeren Modellen noch bessere Resultate zu erzielen. Besonders interessant war der Vergleich verschiedener baumbasierter Modelle. Hierbei erzielte XGBoost die besten Ergebnisse in Bezug auf den MAPE-Wert. Auffällig war zudem, dass sich XGBoost und das HistGradientBoosting-Modell besonders schnell trainieren liessen. Die Vorhersage der Gebäudetypen stellte sich als schwieriger heraus als erwartet, da die Verteilung der Gebäudetypen stark unausgeglichen war. In diesem Zusammenhang überzeugte insbesondere der Random Forest mit seiner robusten Leistung. Zusammenfassend zeigte die Analyse, dass die Wahl des Modells und die Anpassung der Datenaufbereitung massgeblich zur Vorhersagegenauigkeit beitragen. XGBoost erwies sich dabei als besonders leistungsfähig, während der Random Forest seine Stärken bei der Klassifikation von Gebäudetypen ausspielte.

2.10 Ausblick

Um bessere Ergebnisse bei der Klassifikation zu erzielen, könnte man die Gebäudekategorien größer gruppieren. Dies würde beispielsweise die Anwendung von SMOTE (Synthetic Minority Over-sampling Technique) ermöglichen, um das Problem der unausgeglichenen Klassenverteilung zu mildern. Zur Verbesserung der Regressionsmodelle wäre es sinnvoll, sich intensiver mit der Verarbeitung von Textdaten auseinanderzusetzen, um deren zusätzliche Vorhersagekraft besser zu nutzen. Darüber hinaus könnte die Erweiterung des Datensatzes durch zusätzliche Beobachtungen, insbesondere aus den Kantonen Zug und Zürich, die bislang nicht im Datensatz enthalten sind, zu genaueren Ergebnissen führen. Abschliessend können wir festhalten, dass wir viel Freude an der Analyse hatten und zahlreiche Modelle ausprobieren konnten, was uns wertvolle Einblicke und Erfahrungen gebracht hat.