2_2_Bestmögliches_Regressionsmodell

January 11, 2024

cml1 - Immobilienrechner

1 Bestmögliches Regressionsmodell - kaggle-Contest

Aufgabenstellung: Entwickle mit beliebigen Algorithmen das bestmögliche Modell im Sinne des Mean absolute percentage error (MAPE). Vergleiche dabei mindestens drei algorithmische Ansätze, wobei ein multiples lineares Modell Teil davon sein soll als Benchmark. Untersuche die 'Variable Importance' für dein bestes Modell.

1.0.1 MAPE - Metrik

Der Mean Absolute Percentage Error (MAPE) ist eine Metrik zur Beurteilung der Genauigkeit von Vorhersagemodellen. MAPE gibt den durchschnittlichen prozentualen Fehler zwischen den beobachteten (tatsächlichen) Werten und den vorhergesagten Werten an. Er wird häufig in der Zeitreihenanalyse und bei Prognosemodellen verwendet, besonders wenn es wichtig ist, die Genauigkeit von Vorhersagen in Bezug auf die tatsächlichen Werte in Prozent auszudrücken.

Die Formel zur Berechnung von MAPE ist:

$$MAPE = \frac{1}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{y_i} * 100$$

wobei: - y_i der tatsächliche Wert zum Zeitpunkt t ist.

- $\hat{y}_i$ der vorhergesagte Wert zum Zeitpunkt t ist.
- n die Gesamtanzahl der Beobachtungen ist.

1.1 Daten laden und vorbereiten

Wir haben im Notebook "datawrangling.ipynb" den Datensatz bereinigt und lesen dieser hier ein.

```
Anzahl der Zeilen und Spalten: (21466, 57)
```

Wir haben hier eine gründliche Datenverarbeitung durchgeführt, um sicherzustellen, dass unser Datensatz optimal für unsere Analysen und Modelle vorbereitet ist.

Zunächst haben wir die Daten aufgeteilt, indem wir die Spalten mit kategorischen Variablen von den numerischen Variablen unterschieden. Dies ermöglicht es uns, die kategorischen Variablen in Dummy-Variablen umzuwandeln, um sie in unseren Modellen verwenden zu können. Dabei haben wir die `pd.get_dummies`-Funktion verwendet.

Nachdem wir die kategorischen Spalten verarbeitet hatten, haben wir uns auf die numerischen Variablen konzentriert. Hier haben wir uns auf das Entfernen von Ausreissern konzentriert, um die Robustheit unserer Modelle sicherzustellen. Dazu haben wir die 1% und 99% Quantile für jede numerische Spalte berechnet. Alle Datenpunkte, die ausserhalb dieser Quantile lagen, wurden als Ausreisser betrachtet und durch NaN-Werte ersetzt. Dies hilft, extreme Werte zu glätten, die die Modellleistung beeinträchtigen könnten.

Schliesslich haben wir die Zeilen entfernt, die fehlende Werte im Zielvariable `price_cleaned` aufweisen, um sicherzustellen, dass wir mit einem sauberen Datensatz arbeiten.

```
Anzahl der Zeilen und Spalten nach der Vorverarbeitung: (20833, 99)
```

Unser vorbereiteter Datensatz ist jetzt bereit für die weiterführende Analyse und Modellierung.

1.2 Korrelationen

In diesem Abschnitt werden wir hochkorrelierte Merkmale identifizieren. Dies ist ein wichtiger Schritt bei der Auswahl von Merkmalen. Die Logik dahinter ist, dass zwei (oder mehr) Variablen, die hoch korreliert sind, redundante Informationen enthalten, was für einige Algorithmen des maschinellen Lernens, insbesondere die lineare Regression, problematisch sein kann.

Wir berechnen die paarweise Korrelationsmatrix aller Spalten im DataFrame. Anschliessend wird das untere Dreieck dieser Matrix durchlaufen, um Spaltenpaare zu ermitteln, die einen absoluten Korrelationswert von mehr als 0.75 haben. Für jedes solche Paar werden die Namen der beiden Spalten und ihr Korrelationswert in die Liste `highly_correlated_pairs` aufgenommen. Das Ergebnis ist eine Liste von Tupeln, wobei jedes Tupel zwei Spaltennamen enthält, die ein Paar hoch korrelierter Merkmale und deren spezifischen Korrelationskoeffizienten angeben.

```
Anzahl hoch-korrelierte Paare: 26
```

```
[('gde_private_apartments', 'gde_population', 0.9983906151468656),
 ('gde_workers_total', 'gde_workers_sector3', 0.9965264625780591),
 ('gde_private_apartments', 'gde_workers_total', 0.9841766211423276),
 ('gde_private_apartments', 'gde_workers_sector3', 0.9811157323060425),
 ('gde_population', 'gde_workers_total', 0.9796743534569659),
 ('gde_population', 'gde_workers_sector3', 0.9739444811874234),
 ('WorkplaceDensityM', 'WorkplaceDensityL', 0.9209441781880091),
 ('kanton_TI', 'gde_politics_rights', 0.917008926805917),
 ('ForestDensityM', 'ForestDensityL', 0.9154568661789924),
 ('PopulationDensityM', 'PopulationDensityL', 0.907554821625827),
 ('WorkplaceDensityS', 'WorkplaceDensityM', 0.907194688248403),
 ('NoisePollutionRoadM', 'NoisePollutionRoadL', 0.894449390632457),
 ('ForestDensityS', 'ForestDensityM', 0.8896128280449461),
 ('gde_pop_per_km2', 'gde_area_settlement_percentage', 0.8776308057232483),
 ('PopulationDensityS', 'PopulationDensityM', 0.8498122735512645),
 ('Floor_space_merged', 'Living_area_unified', 0.8247898295393741),
 ('gde_workers_total', 'gde_workers_sector2', 0.8078595437630071),
 ('NoisePollutionRoadS', 'NoisePollutionRoadM', 0.804717721529123),
 ('Rooms_new', 'Living_area_unified', 0.7989357833270612),
 ('NoisePollutionRailwayM', 'NoisePollutionRailwayL', 0.7867739807244442),
```

```
('kanton_BE', 'gde_politics_bdp', 0.7600456105133713),
('RiversAndLakesM', 'RiversAndLakesL', 0.7593828416286377),
('WorkplaceDensityS', 'WorkplaceDensityL', 0.7576508482078738),
('gde_population', 'gde_workers_sector2', 0.7560044682363133),
('WorkplaceDensityL', 'PopulationDensityL', 0.7508886345509813),
('ForestDensityS', 'ForestDensityL', 0.7502920622585877)]
```

Wir haben nun festgestellt, dass 26 Paare von Merkmalen eine hohe Korrelation von mehr als 0.75 aufweisen. In solchen Fällen ist es wichtig, die redundanten Informationen zu reduzieren, da hochkorrelierte Merkmale in einigen Machine-Learning-Modellen zu Problemen führen können, insbesondere in linearen Regressionen.

Folgende Optionen stehen uns zur Verfügung:

- **Feature-Auswahl:** Wir können eine der beiden hochkorrelierten Spalten entfernen und nur eine beibehalten. Die Auswahl der zu behaltenden Spalte sollte auf der Relevanz für das Problem und der Interpretierbarkeit basieren. Hier sollte die Feature-Importance-Analyse helfen, welche wir später mit Random Forest und XGBoost durchführen werden.
- **Feature-Transformation:** Wenn beide Spalten wichtig sind und Informationen enthalten, die für das Modell relevant sind, können wir versuchen, die beiden Spalten zu einer neuen, abgeleiteten Spalte zu kombinieren. Dies könnte beispielsweise durch Addition, Subtraktion oder Durchschnittsbildung geschehen.

1.3 Train Test Split

Nun werden die Daten in 80% Trainings- und 20% Testdaten aufgeteilt. Die Zielvariable ist `price_cleaned` und wird ebenfalls getrennt.

```
Anzahl der Zeilen und Spalten von X_train: (16666, 98)
Anzahl der Zeilen und Spalten von y_train: (16666,)
Anzahl der Zeilen und Spalten von X_test: (4167, 98)
Anzahl der Zeilen und Spalten von y_test: (4167,)
```

1.4 Imputation

Bevor wir die Imputation mit KNN und 5 Nachbarn durchführen, werden wir ein Trainings- und Testset erstellen, bei denen keine Imputation durchgeführt wird. Diese Datensätze werden wir später für die Modelle verwenden, um sie mit und ohne Imputation zu vergleichen.

Sonst verwenden wir den `KNNImputer` von `sklearn`. Dieser Imputer verwendet die k-Nearest Neighbors Methode, um fehlende Werte zu imputieren. In unserem Fall betrachten wir die 5 nächsten Nachbarn.

1.5 Standardisierung

Hier verwenden wir den `StandardScaler` von `sklearn`, um die Daten zu standardisieren. Dieser Skalierer standardisiert Merkmale, indem er den Mittelwert auf 0 und die Standardabweichung auf 1 setzt. Wir berücksichtigen dabei nur die numerischen Spalten.

1.6 Modelle

In diesem Abschnitt werden wir verschiedene Modelle trainieren und ihre Leistung vergleichen. Dabei werden wir folgende Modelle verwenden:

1. **Multiple Linear Regression (Benchmark):** Dieses Modell wird als Benchmark dienen und ermöglicht uns, die Leistung der anderen Modelle im Vergleich zu einer einfachen linearen Regression zu bewerten.
2. **Ridge Regression:** Die Ridge-Regression ist eine Variante der linearen Regression, die dazu beiträgt, Overfitting zu reduzieren, indem sie einen L2-Regularisierungsterm hinzufügt. Dies kann nützlich sein, um die Modellgenauigkeit zu verbessern.
3. **Lasso Regression:** Die Lasso-Regression ist ähnlich wie die Ridge-Regression, jedoch mit einem L1-Regularisierungsterm. Sie kann dazu beitragen, irrelevante Features zu eliminieren und das Modell zu vereinfachen.
4. **XGBoost:** XGBoost ist ein leistungsstarker Gradient-Boosting-Algorithmus, der für seine Fähigkeit bekannt ist, komplexe nichtlineare Beziehungen in den Daten zu modellieren.
5. **Random Forest:** Der Random Forest ist ein Ensemble-Modell, das aus einer Menge von Entscheidungsbäumen besteht. Es ist robust gegenüber Overfitting und kann sowohl für Regression als auch für Klassifikation verwendet werden.
6. **Histogram Gradient Boosting:** Das Histogram Gradient Boosting ist eine effiziente Variante des Gradient Boosting-Algorithmus, die in der Lage ist, grosse Datensätze effizient zu verarbeiten.
7. **Gradient Boosting:** Gradient Boosting ist ein leistungsstarker Ensemble-Algorithmus, der schrittweise schwache Modelle zu einem starken Modell kombiniert.

Auch die Log-Transformation auf die Zielvariable wird durchgeführt, um sicherzustellen, dass die Verteilung der Zielvariable näher an einer Normalverteilung liegt. Dies kann die Modellleistung verbessern, da viele Modelle, einschliesslich linearer Modelle, auf Annahmen basieren, die eine Normalverteilung der Residuen voraussetzen. Die Log-Transformation kann dazu beitragen, Ausreisser zu mildern und die Homoskedastizität zu fördern, was die Modellgenauigkeit erhöhen kann.

Wir werden auch bei XGBoost und Random Forest die Feature-Importance-Analyse durchführen, um die wichtigsten Merkmale zu identifizieren, um eventuell irrelevante Merkmale zu eliminieren.

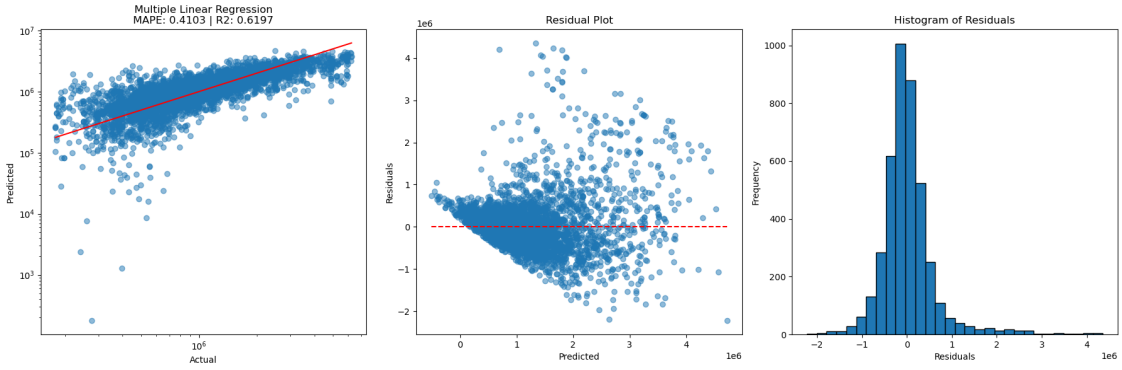
1.6.1 Modell 1: Multiple Linear Regression (Benchmark)

Die Multiple Linear Regression (MLR) ist ein Supervised Learning-Modell, das dazu verwendet wird, die Beziehung zwischen einer abhängigen Zielvariable und mehreren unabhängigen Merkmalen herzustellen. Es modelliert diese Beziehung als lineare Kombination der Merkmale und zielt darauf ab, die besten Gewichtungen für jede Funktion zu finden, um Vorhersagen für die Zielvariable zu treffen. Die MLR wird verwendet, um numerische Werte vorherzusagen und eignet sich gut für einfache lineare Zusammenhänge zwischen Merkmalen und der Zielvariable.

sklearn - Linear Regression

```
Fitting 5 folds for each of 8 candidates, totalling 40 fits
Best parameters:  {'copy_X': True, 'fit_intercept': True, 'positive': False}
```

```
MAPE: 0.4103
R2: 0.6197
```



MAPE gibt her einen durchschnittlichen prozentualen Fehler (MAPE) von 41.03% und einen R^2 von 61.97% zwischen den beobachteten (tatsächlichen) Werten und den vorhergesagten Werten an.

1.6.2 Modell 2: Ridge Regression

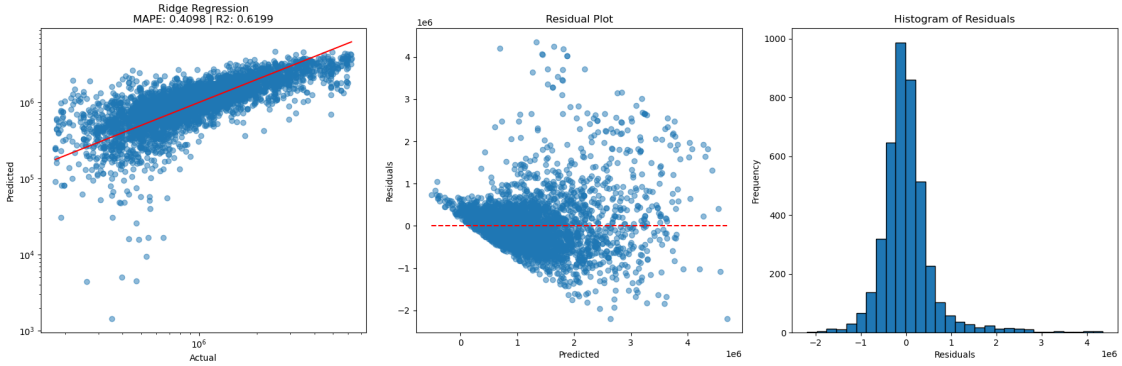
Die Ridge Regression ist eine Variante der linearen Regression, die dazu dient, Overfitting zu reduzieren und die Stabilität des Modells zu verbessern. Sie fügt einen L2-Regularisierungsterm zur linearen Regression hinzu, um die Gewichtungen der Merkmale zu begrenzen, was dazu beiträgt, insbesondere in hochdimensionalen Datensätzen, eine bessere Modellleistung zu erzielen. Ridge Regression ist nützlich, um die Auswirkungen von Multikollinearität zu mildern und das Modell robuster zu machen.

sklearn - Ridge Regression

```
Fitting 5 folds for each of 96 candidates, totalling 480 fits
```

```
/Users/alexanderschilling/anaconda3/lib/python3.11/site-
packages/joblib/externals/loky/process_executor.py:702: UserWarning: A worker
stopped while some jobs were given to the executor. This can be caused by a too
short worker timeout or by a memory leak.
  warnings.warn(
```

```
Best parameters: {'alpha': 10.0, 'fit_intercept': True, 'max_iter': None,
'solver': 'svd'}
MAPE: 0.4098
R2: 0.6199
```

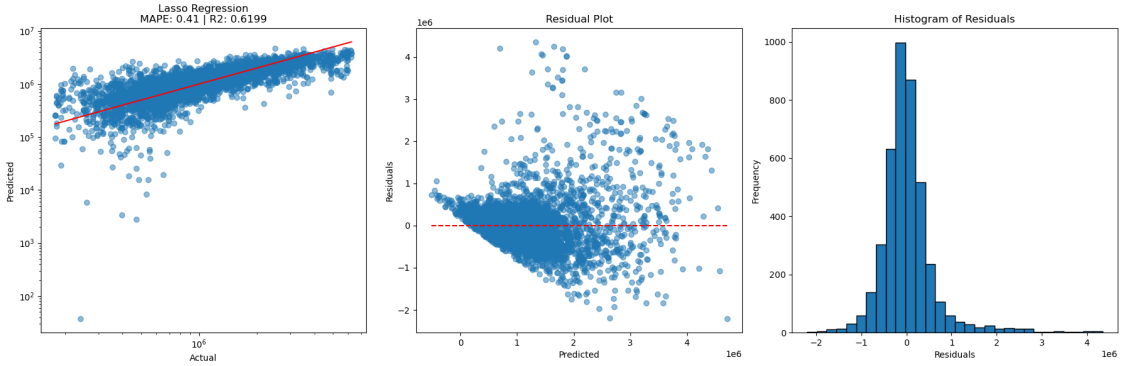


Ridge Regression ist im Vergleich zum Basismodell (Multiple Linear Regression) nur minim besser. Der MAPE beträgt 40.98% und der R^2 61.99%.

1.6.3 Modell 3: Lasso Regression

Lasso Regression funktioniert durch die Anwendung einer Straftechnik, die als L1-Regularisierung bekannt ist, wodurch einige Koeffizienten exakt auf null gesetzt werden. Dies führt zu spärlichen Modellen, in denen nur eine begrenzte Anzahl von Merkmalen gewichtet wird, was die Interpretation und Visualisierung der Daten erleichtert.

sklearn - Lasso Regression



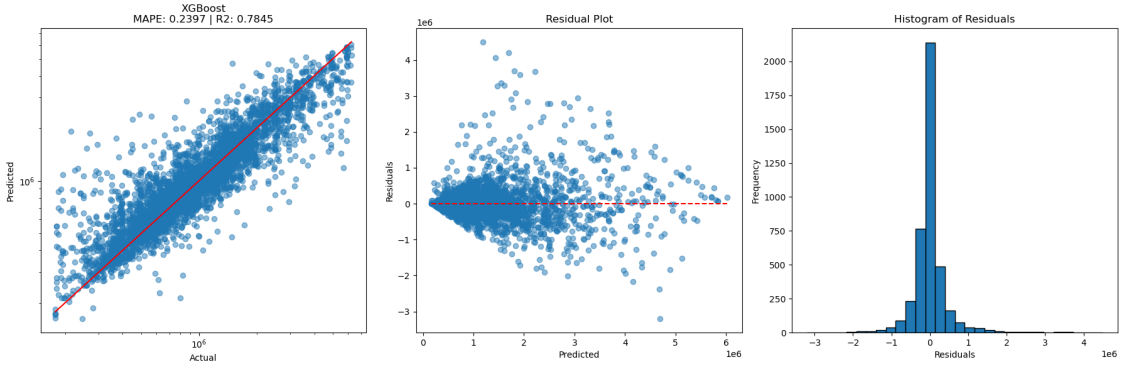
Mittels Lasso Regression wird keine Verbesserung gegenüber dem Basismodell erzielt. Dies Resultate bei den linearen Modellen sind nicht überraschend, da die linearen Modelle nicht in der Lage sind, die komplexen nichtlinearen Beziehungen in den Daten zu modellieren.

1.6.4 Modell 4: XGBoost

XGBoost (Extreme Gradient Boosting) ist eine fortschrittliche Implementierung des Gradient Boosting-Algorithmus, der für seine Effizienz, Leistungsfähigkeit und Flexibilität bekannt ist. XGBoost verwendet eine Reihe von Entscheidungsbäumen, die iterativ trainiert werden, um den Fehler der vorherigen Bäume zu minimieren, wobei jeder nachfolgende Baum versucht, die Fehler seines Vorgängers zu korrigieren. Diese Methode führt zu einem robusten Vorhersagemodell, das sich besonders gut für grosse und komplexe Datensätze eignet und eine hohe Vorhersagegenauigkeit bietet.

XGBoost

```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
Best parameters: {'eval_metric': 'mape', 'learning_rate': 0.1, 'max_depth': 9,
'nthread': -1, 'reg_alpha': 0.1}
MAPE: 0.2397
R2: 0.7845
```

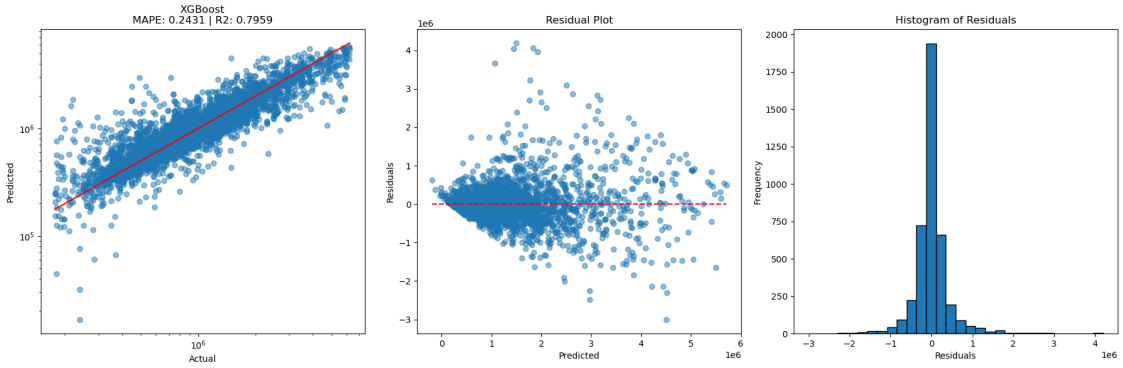


Verglichen mit einem linearen Modell, wie der linearen Regression, ist XGBoost in der Lage, die komplexen nichtlinearen Beziehungen in den Daten besser zu modellieren und wir sehen direkt eine erhebliche Verbesserung der Resultate. Der $MAPE$ ist nun auf 23.97% gesunken und der R^2 auf 78.45% gestiegen.

1.6.5 Modell 5: XGBoost (Train mit NA-Werte)

Hier trainieren wir ein XGB-Modell mit dem Train Set, das die NA-Werte enthält.

```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
Best parameters: {'eval_metric': 'mape', 'learning_rate': 0.2, 'max_depth': 6,
'nthread': -1, 'reg_alpha': 0.1}
MAPE: 0.2431
R2: 0.7959
```



Wenn wir das Modell nun mit dem Train Set mit fehlenden Werten trainieren, erhalten wir einen $MAPE$ von 24.31% und einen R^2 von 79.59% Verglichen mit Modell 4 ist die Modellleistung im Sinne von $MAPE$ etwas schlechter geworden, aber der R^2 ist etwas besser geworden.

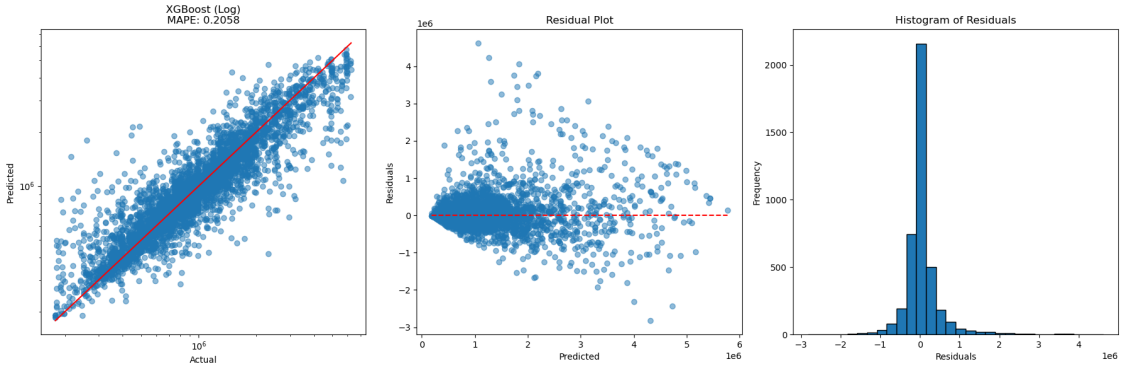
1.6.6 Modell 6: XGBoost (Log-Transformiert)

Hier trainieren wir ein XGB-Modell mit dem imputierten Train Set und der Log-Transformation auf der Zielvariable.

```
Fitting 5 folds for each of 24 candidates, totalling 120 fits

/Users/alexanderschilling/anaconda3/lib/python3.11/site-
packages/joblib/externals/loky/process_executor.py:702: UserWarning: A worker
stopped while some jobs were given to the executor. This can be caused by a too
short worker timeout or by a memory leak.
  warnings.warn(

Best parameters: {'eval_metric': 'mape', 'learning_rate': 0.2, 'max_depth': 9,
'nthread': -1, 'reg_alpha': 1.0}
MAPE: 0.2058
R2: 0.7793
```



Wir sehen bei dieser Variante eine starke Verbesserung im Sinne des $MAPE$, welcher nun bei 20.58% liegt. Der R^2 ist jedoch auf 77.93% gesunken.

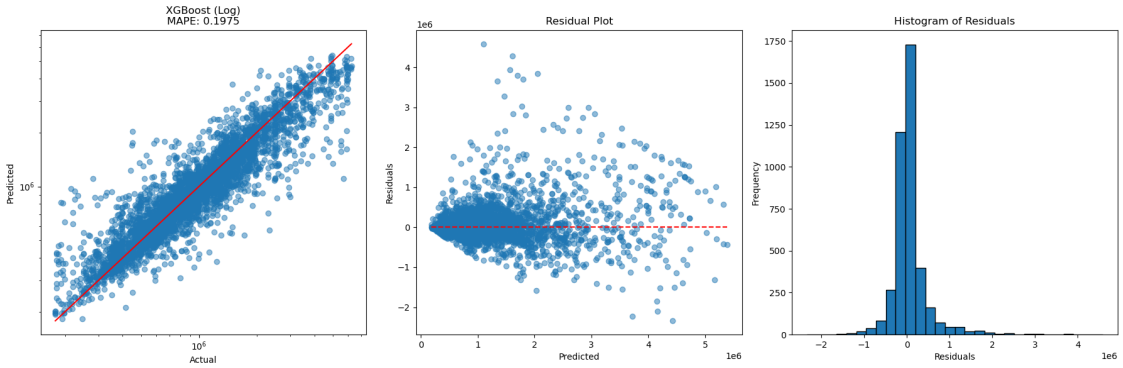
1.6.7 Modell 7: XGBoost (Log-Transformiert, Train mit NA-Werte)

Hier trainieren wir ein XGB-Modell mit em Train Set, das die NA-Werte enthält und der Log-Transformation auf der Zielvariable.

```
Fitting 5 folds for each of 24 candidates, totalling 120 fits

/Users/alexanderschilling/anaconda3/lib/python3.11/site-
packages/joblib/externals/loky/process_executor.py:702: UserWarning: A worker
stopped while some jobs were given to the executor. This can be caused by a too
short worker timeout or by a memory leak.
  warnings.warn(

Best parameters: {'eval_metric': 'mape', 'learning_rate': 0.1, 'max_depth': 9,
'nthread': -1, 'reg_alpha': 1.0}
MAPE: 0.1975
R2: 0.7926
```

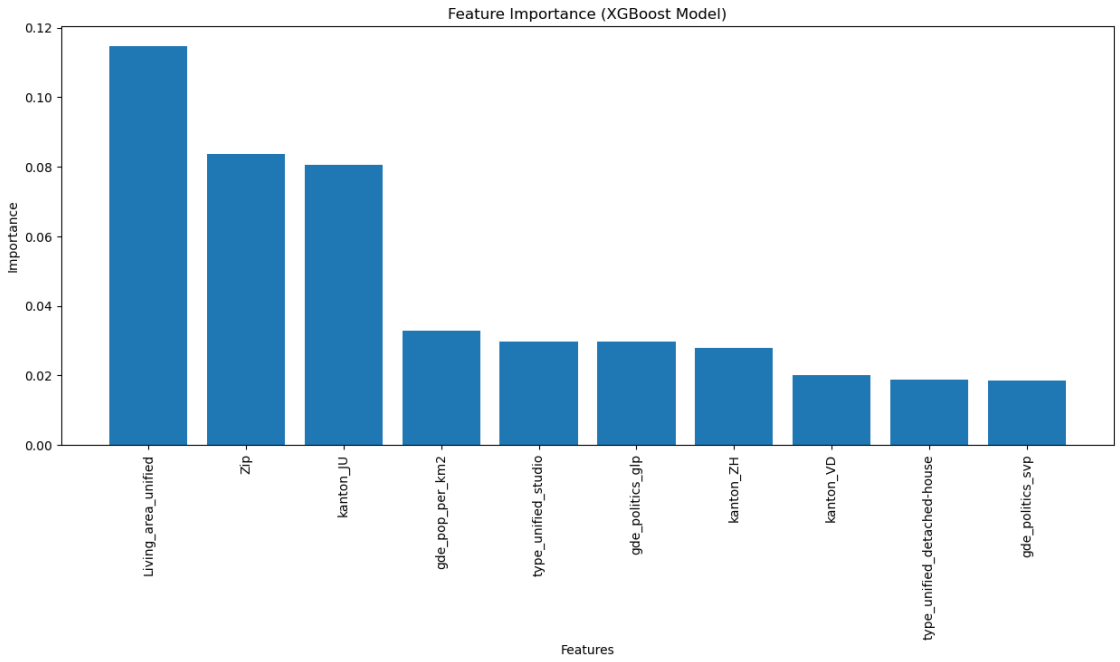


Das bisherige beste XGB-Modell wurde nun erreicht. Der $MAPE$ beträgt 19.75% und der R^2 79.26%.

1.6.8 Feature Importance: XGBoost

In XGBoost wird die Feature Importance standardmässig anhand der "Gewichtung" berechnet, was die Anzahl der Male ist, dass ein Feature in den Split-Punkten aller Bäume verwendet wird. Ein Feature wird als wichtiger angesehen, wenn es häufiger an Entscheidungspunkten in den Bäumen verwendet wird.

machinelearningmastery - Feature Importance with XGBoost

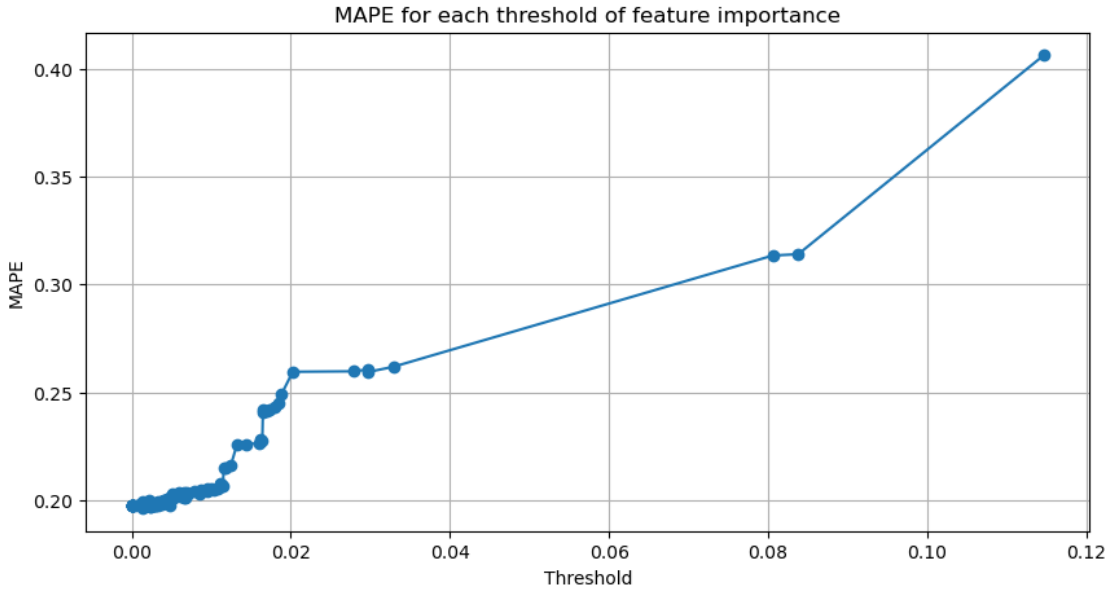


`Living_area_unified` scheint das wichtigste Merkmal für das Modell zu sein. Dies macht auch Sinn, da die Grösse der Wohnfläche in der Realität einen grossen Einfluss auf den Preis hat. `Zip` und überraschenderweise `kanton_JU` weisen ebenfalls relativ hohe Wichtigkeitswerte auf. Das bedeutet, dass der Standort auch ein Schlüsselfaktor für die Vorhersagen des Modells ist.

Andere Merkmale, darunter `gde_pop_per_km2`, `type_unified_studio`, `kanton_ZH`, `kanton_VD` und `gde_politics_glp`, haben geringere Wichtigkeitswerte, was bedeutet, dass sie weniger zu den Vorhersagen beitragen, aber immer noch zu den wichtigsten Merkmalen gehören, die vom Modell berücksichtigt werden.

1.6.9 Modell 8: XGBoost (Log-Transformiert, Train mit NA-Werte, Feature Importance)

Wir wollen die Feature Importance weiter untersuchen, indem das Modell immer wieder einzelne Merkmale hinzugefügt werden. Wir werden die Merkmale in der Reihenfolge ihrer Wichtigkeit hinzufügen und die Auswirkungen auf die Modellleistung beobachten.



Zu Beginn, wenn der Schwellenwert auf Null oder nahe Null gesetzt wird, werden alle Merkmale verwendet, und der $MAPE$-Wert ist am niedrigsten, was darauf hindeutet, dass das Modell am besten funktioniert, wenn alle bzw. die meisten Merkmale einbezogen werden.

Untersuchen wir dies genauer. Wir suchen nun den optimale Schwellenwert, also dort wo $MAPE$ am niedrigsten ist.

```
Best threshold: 0.0013734367676079273
Index of best threshold: 14
MAPE: 0.19618698559589726
```

Wir sehen also, dass der optimale Schwellenwert bei 0.00137 liegt. Die Anzahl der Variablen unter diesem Schwellenwert beträgt 14, was bedeutet, dass wir 14 Variablen entfernen könnten, um den besten MAPE-Wert von 19.61% zu erhalten.

Wir schauen uns nun die Feature Importance der 14 Variablen an, die unter dem Schwellenwert liegen.

```
kanton_AI
kanton_AR
kanton_GE
kanton_GL
kanton_GR
kanton_OW
kanton_SH
kanton_TI
kanton_UR
kanton_VS
type_unified_loft
type_unified_stepped-house
```

`Availability_Categorized_Within 3 months`
`Availability_Categorized_Within 6 months`

Alle 14 Variablen stammen aus den kategorischen Variablen `kanton`, `type_unified` und `Availability_Categorized`, welche wir in Dummy-Variablen umgewandelt haben. Es kann sein, dass diese Variablen nicht so wichtig sind, weil sie nicht so viele Informationen enthalten und so recht sparse sind (vor allem einige Immobilienarten von `type_unified` und Werte von `Availability_Categorized`).

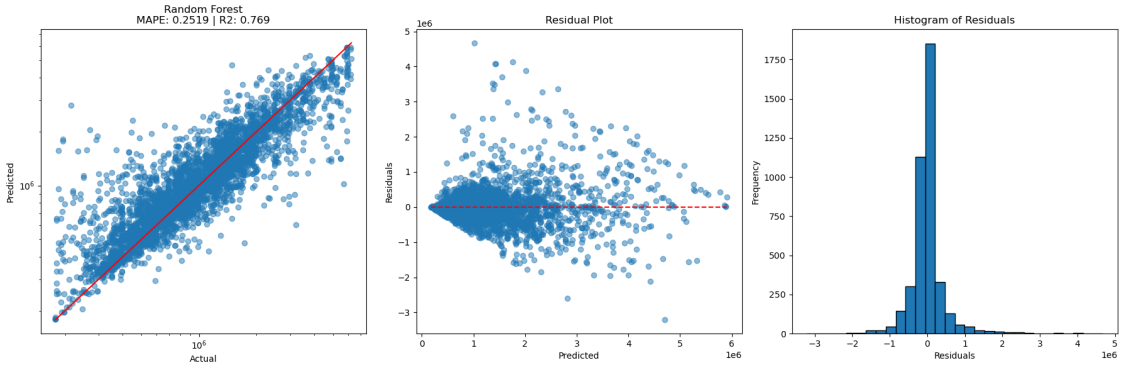
Für den Kaggle-Wettbewerb werden wir Modelle testen mit und ohne diese unwichtigen Variablen, um zu sehen, ob wir bessere Resultate erzielen können und das Modell einfacher wird.

1.6.10 Modell 9: Random Forest

Random Forest Regression ist ein Ensemble-Lernverfahren, das auf der Aggregation mehrerer Entscheidungsbäume basiert, um robustere und genaue Vorhersagen zu liefern. Jeder Baum im Wald wird auf einer zufälligen Stichprobe der Daten mit Ersatz gebaut und verwendet eine zufällige Auswahl von Merkmalen bei jedem Split, was zu einer Reduktion von Überanpassung und Varianz führt. Das Endergebnis ist der Durchschnitt der Vorhersagen aller Bäume im Wald, was in der Regel zu einer Verbesserung der Vorhersageleistung und Stabilität im Vergleich zu einzelnen Entscheidungsbäumen führt.

sklearn - Random Forest

```
Fitting 5 folds for each of 1 candidates, totalling 5 fits
Best parameters: {}
MAPE: 0.2519
R2: 0.769
```



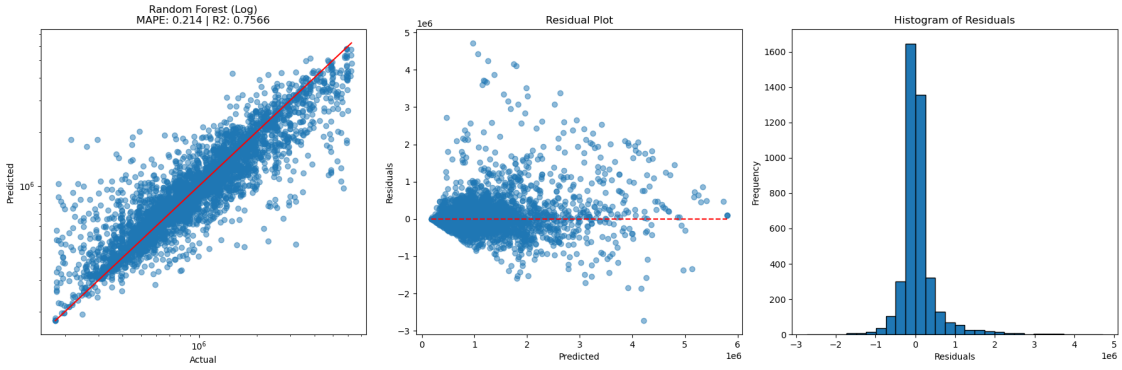
Wir bekommen mit dem Random Forest Modell einen $MAPE$ von 25.19% und einen R^2 von 76.9%.

1.6.11 Modell 10: Random Forest (Log-Transformiert)

Hier trainieren wir ein Random Forest-Modell mit dem imputierten Train Set und der Log-Transformation auf der Zielvariable.

```
Fitting 2 folds for each of 1 candidates, totalling 2 fits
Best parameters: {}
```

```
MAPE: 0.214
R2: 0.7566
```

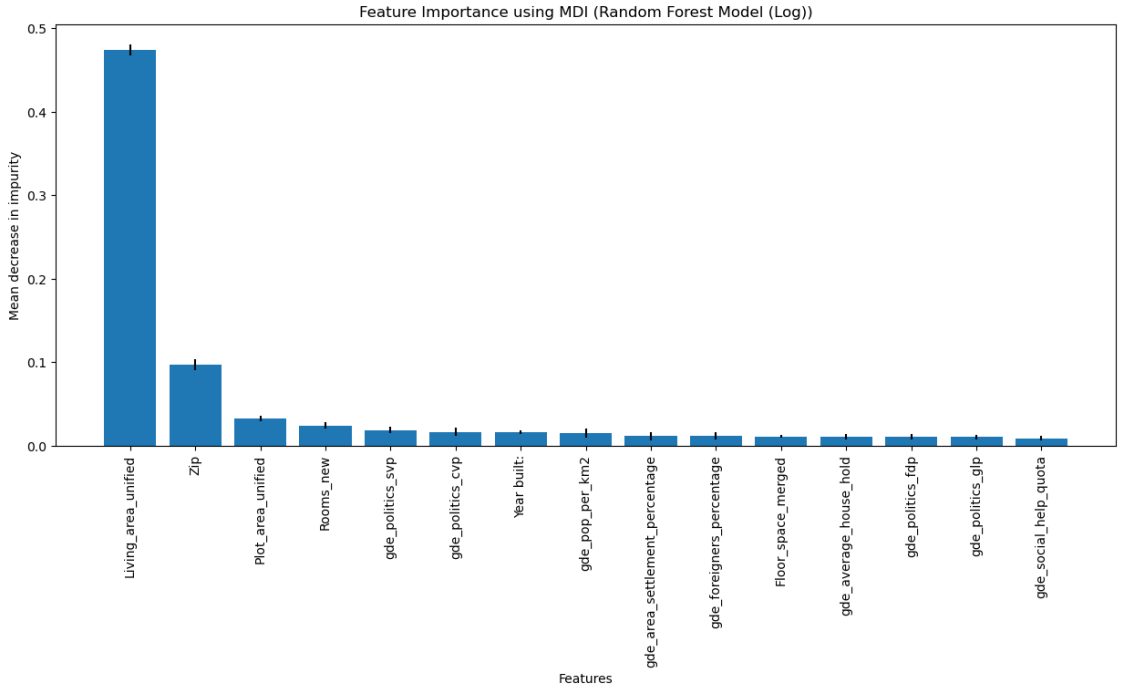


Mit der Log-Transformation der Zielvariable erhalten wir für den Random Forest einen $MAPE$ von 21.14%, was etwa 4% besser ist als ohne Log-Transformation.

1.6.12 Feature Importance: Random Forest

Die Feature Importance in einem Random Forest-Modell gibt an, wie wichtig jedes Merkmal (Feature) für die Vorhersagekraft des gesamten Modells ist. In diesem Code wird die Wichtigkeit eines Merkmals anhand des Mean Decrease in Impurity (MDI) berechnet, was bedeutet, wie viel die Impurity (hier typischerweise gemessen durch die Gini-Impurity) im Durchschnitt abnimmt, wenn das Merkmal in den Bäumen des Random Forest verwendet wird.

sklearn - Feature importances with a forest of trees



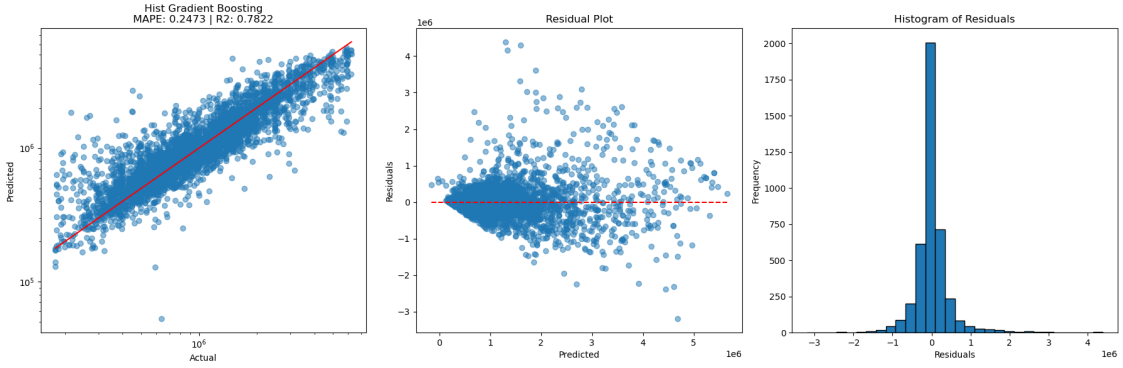
Aus dem Plot geht hervor, dass das Merkmal `Living_area_unified` die höchste Wichtigkeit besitzt und somit den grössten Einfluss auf die Vorhersageleistung des Modells hat. Die zweitwichtigste Variable ist `Zip`. Die schwarzen Linien, die in den Balken zu sehen sind, stellen die Standardabweichung der Wichtigkeit dieses Merkmals über alle Bäume hinweg dar und geben somit einen Anhaltspunkt für die Konsistenz seiner Bedeutung im Modell. Die anderen Merkmale tragen weniger zur Vorhersageleistung bei. Dieses Wissen kann genutzt werden, um die Merkmale zu identifizieren, die für die Modellierung am relevantesten sind, und um möglicherweise die Dimensionalität des Problems zu reduzieren, indem weniger wichtige Merkmale entfernt werden.

1.6.13 Modell 11: Hist Gradient Boosting Regression

Hist Gradient Boosting Regression ist eine effiziente Implementierung des Gradient Boosting-Frameworks, die auf Entscheidungsbäumen basiert und speziell für grosse Datensätze entwickelt wurde. Es verwendet Histogramm-basierte Optimierung, bei der die kontinuierlichen Eingabevariablen in diskrete Bins eingeteilt werden, was die Berechnung schneller und weniger speicherintensiv macht. Diese Technik ermöglicht es dem Modell, schneller zu konvergieren und mit grossen Datenmengen umzugehen, während es immer noch von der Vorhersagekraft der Gradient Boosting-Modelle profitiert.

sklearn - Hist Gradient Boosting Regression

```
Fitting 5 folds for each of 72 candidates, totalling 360 fits
Best parameters: {'l2_regularization': 0.1, 'learning_rate': 0.1, 'max_depth':
9, 'max_iter': 300}
MAPE: 0.2473
R2: 0.7822
```

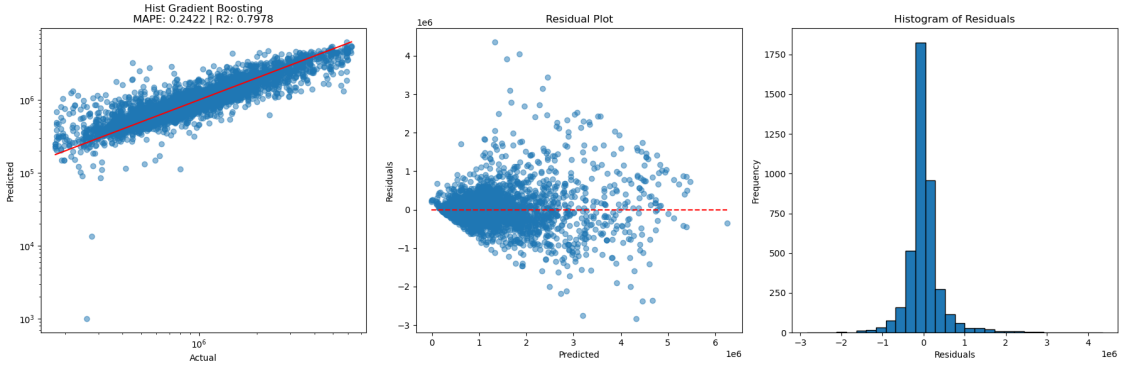


Mittels Hist Gradient Boosting Regression erhalten wir im Vergleich zum Baseline-Modell eine starke Verbesserung des $MAPE$-Werts von 24.73%. XGBoost ist immer noch besser.

1.6.14 Modell 12: Hist Gradient Boosting Regression (Train mit NA-Werte)

Hier trainieren wir ein Hist-Gradient-Boosting-Regression-Modell mit dem Train Set, das die NA-Werte enthält.

```
Fitting 5 folds for each of 72 candidates, totalling 360 fits
Best parameters: {'l2_regularization': 0.0, 'learning_rate': 0.1, 'max_depth':
9, 'max_iter': 300}
MAPE: 0.2422
R2: 0.7978
```

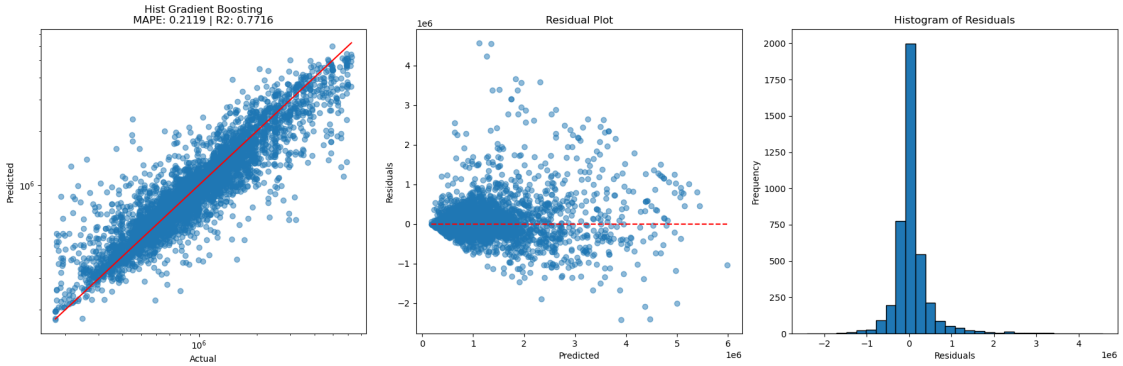


Verglichen mit dem Modell, das mit dem imputierten Train Set trainiert wurde, ist die Vorhersageleistung des Modells mit dem Train Set mit NA-Werten etwas besser. Der $MAPE$-Wert ist von 24.73% auf 24.22% gesunken.

1.6.15 Modell 13: Hist Gradient Boosting Regression (Log-Transformiert)

Hier trainieren wir ein Hist-Gradient-Boosting-Regression-Modell mit dem imputierten Train Set und der Log-Transformation auf der Zielvariable.

```
Fitting 5 folds for each of 72 candidates, totalling 360 fits
Best parameters: {'l2_regularization': 0.1, 'learning_rate': 0.1, 'max_depth':
9, 'max_iter': 300}
MAPE: 0.2119
R2: 0.7716
```

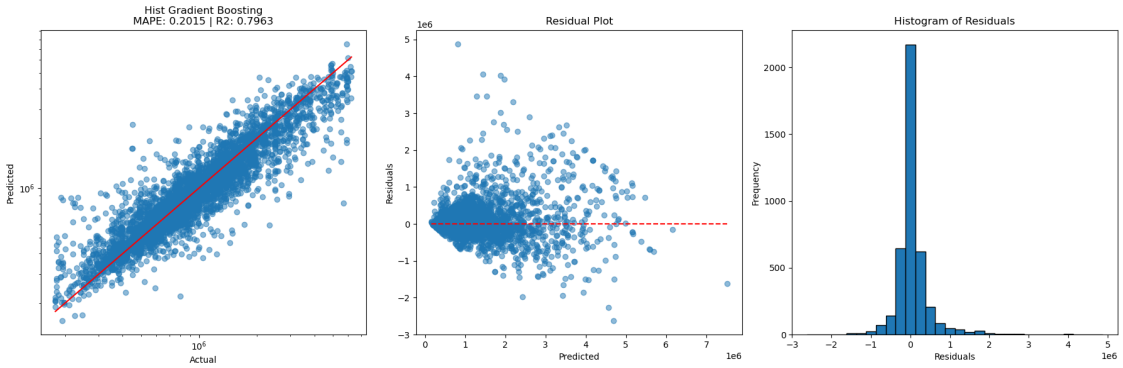


Wir konnten so den $MAPE$-Wert von um 3% senken.

1.6.16 Modell 14: Hist Gradient Boosting Regression (Log-Transformiert, Train mit NA-Werte)

Hier trainieren wir ein Hist-Gradient-Boosting-Regression-Modell mit em Train Set, das die NA-Werte enthält und der Log-Transformation auf der Zielvariable.

```
Fitting 5 folds for each of 72 candidates, totalling 360 fits
Best parameters: {'l2_regularization': 0.0, 'learning_rate': 0.1, 'max_depth':
None, 'max_iter': 300}
MAPE: 0.2015
R2: 0.7963
```



Verglichen mit dem normalen Hist Gradient Boosting Modell (Modell 11) ist die Vorhersageleistung des Modells mit dem Train Set mit NA-Werten und der Log-Transformation um etwa 4.5% besser geworden. Derliegt nun bei 20.15%.

1.6.17 Modell 15: Gradient Boosting Regression

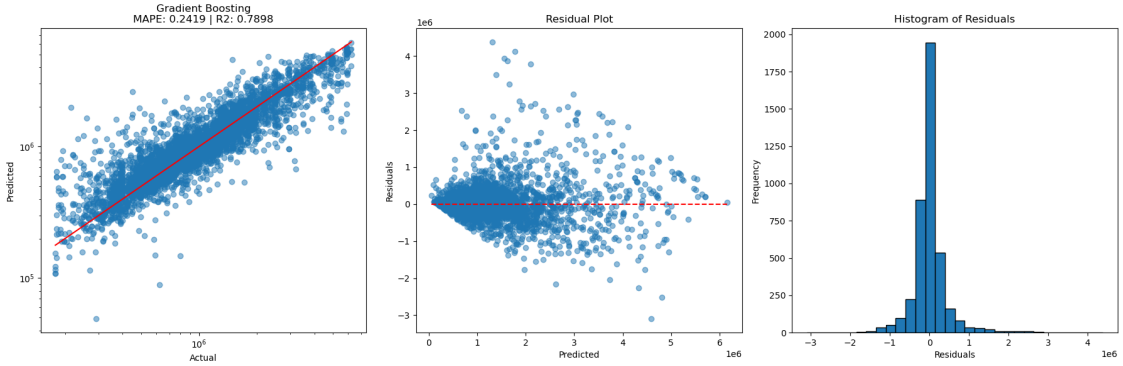
Die Gradient Boosting Regression ist ein Ensemble-Modell, das auf der Idee des schrittweisen Verbesserns von schwachen Vorhersagemodellen basiert. Es passt wiederholt Entscheidungsbäume an die Residuen der vorherigen Modelle an und kombiniert diese, um eine stärkere Vorhersage zu erstellen. Dieses iterative Lernverfahren führt zu einem leistungsstarken Modell, das in der Lage ist, komplexe nicht-lineare Beziehungen zwischen den Merkmalen und der Zielvariable zu modellieren.

sklearn - Gradient Boosting Regression

```
Fitting 5 folds for each of 36 candidates, totalling 180 fits

/Users/alexanderschilling/anaconda3/lib/python3.11/site-
packages/joblib/externals/loky/process_executor.py:702: UserWarning: A worker
stopped while some jobs were given to the executor. This can be caused by a too
short worker timeout or by a memory leak.
  warnings.warn(

Best parameters: {'learning_rate': 0.1, 'max_depth': 6, 'min_samples_split': 4,
'n_estimators': 300}
MAPE: 0.2419
R2: 0.7898
```

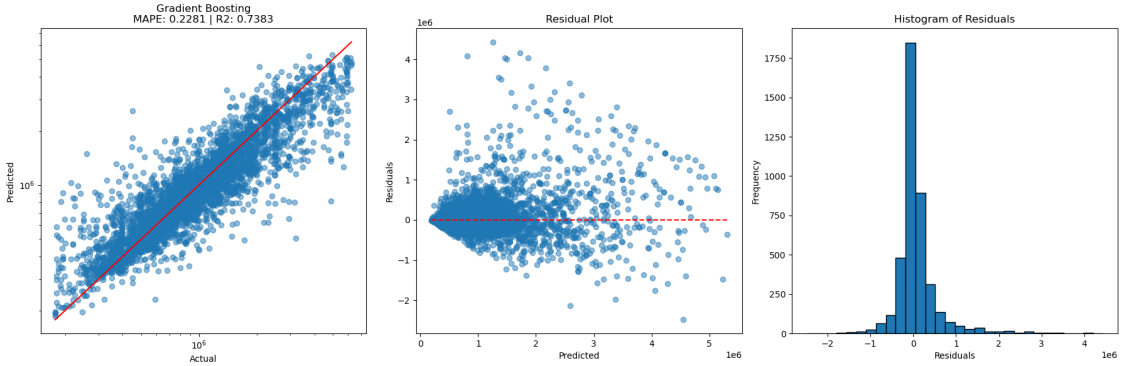


Mit Gradient Boosting erhalten wir im Vergleich zu den anderen Modellen eine mittelmässiges Modell mit einem $MAPE$-Wert von 24.19%.

1.6.18 Modell 16: Gradient Boosting Regression (Log-Transformiert)

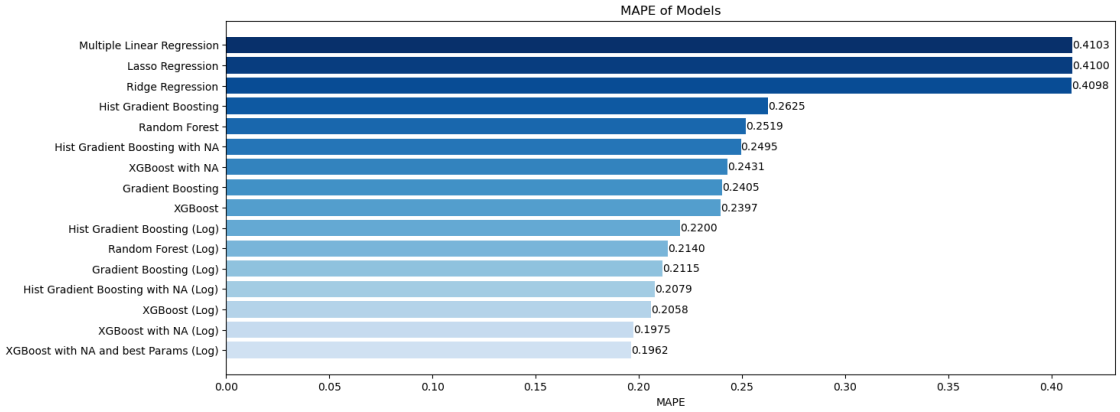
Hier trainieren wir ein Gradient-Boosting-Modell mit dem imputierten Train Set und der Log-Transformation auf der Zielvariable.

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits
Best parameters: {'learning_rate': 0.1, 'max_depth': 6}
MAPE: 0.2281
R2: 0.7383
```



Verglichen mit dem normalen Gradient Boosting Modell (Modell 15) ist die Vorhersageleistung des Modells mit der Log-Transformation um etwa 1.5% besser geworden. Der $MAPE$-Wert liegt nun bei 22.81%.

1.7 Modelle vergleichen



1.7.1 Interpretation der Ergebnisse

Hier haben wir nun alle Modelle verglichen. Aus dem Diagramm geht hervor, dass die einfacheren linearen Modelle wie die Multiple Lineare Regression, Lasso Regression und Ridge Regression die höchsten MAPE-Werte aufweisen, was auf eine geringere Vorhersagegenauigkeit im Vergleich zu den Ensemble-Methoden hinweist.

Insbesondere die Varianten von XGBoost und Gradient Boosting mit Log-Transformation und der Umgang mit fehlenden Daten scheinen die besten Modelle in dieser speziellen Analyse zu sein, mit dem XGBoost mit NA (Log) Modell, das den niedrigsten MAPE-Wert hat und damit die präziseste Vorhersagegenauigkeit aufweist.

Das beste Modell: XGBoost mit Log-Transformation der Zielvariable, ohne Imputation und ohne einflusslose Merkmale

Was bedeutet dies für die Vorhersage von Immobilienpreisen?

`Durchschnittlicher Immobilienpreis: CHF 1164790.61`

Unser Mean Absolute Percentage Error (MAPE) von etwa 20% bedeutet, dass die Vorhersagen des Modells im Durchschnitt um 20% von den tatsächlichen Immobilienpreisen abweichen. Wenn der durchschnittliche Immobilienpreis in unserem Trainings-Datensatz CHF 1'164'790.61 beträgt, dann beträgt die durchschnittliche Abweichung in CHF aufgrund des MAPE:

$0.1962 \times 1'164'790.61 \approx 228'531.92$

Das bedeutet, dass die Vorhersagen des Modells im Durchschnitt um etwa CHF 228'531.92 von den tatsächlichen Preisen abweichen. Diese Abweichung ist der Betrag, um den das Modell durchschnittlich über oder unter den tatsächlichen Preisen liegt.

1.8 Ausblick

1. **Feature Engineering:** Das Erweitern des Feature Engineerings könnte neue Merkmale umfassen, die besser für Vorhersagen geeignet sind. Beispielsweise könnten Interaktionsterme

zwischen Variablen eingeführt werden, um komplexe Beziehungen besser abzubilden. Eine weitere Möglichkeit wäre, Trainingsdaten zu transformieren, um die Verteilung der Daten zu verändern, z.B. durch Log-Transformation oder Wurzeltransformation.

2. **Modellensembles:** Die Kombination der Vorhersagen mehrerer Modelle durch Ensembling-Techniken kann die Gesamtleistung verbessern. Modelle, die sich ergänzen, sind solche, die unterschiedliche Fehlermuster aufweisen und verschiedene Aspekte der Daten erfassen. Beispielsweise könnte ein Ensemble aus einem baumbasierten Modell wie Random Forest und einem linearen Modell wie Lasso Regression genutzt werden, da sie auf unterschiedliche Arten von Datenstrukturen reagieren.

    - **Fehleranalyse:** Um die sich ergänzenden Modelle zu identifizieren, sollten die Residuals oder Fehler der einzelnen Modelle untersucht werden. Modelle, deren Fehler gering korreliert sind, können bei der Kombination ihrer Vorhersagen zu einer insgesamt stärkeren Leistung führen.
    - **Protokollierung der Fehlermuster:** Ein genaues Protokoll darüber, wo jedes Modell Fehler macht (z.B. über- oder unterbewertet bestimmte Beobachtungen), kann bei der Entscheidung helfen, welche Modelle sich in einem Ensemble gut ergänzen. Modelle, die in verschiedenen Bereichen des Datenraums gut abschneiden, können potentiell kombiniert werden, um die Schwächen des anderen auszugleichen.