

ml_regression

January 10, 2021

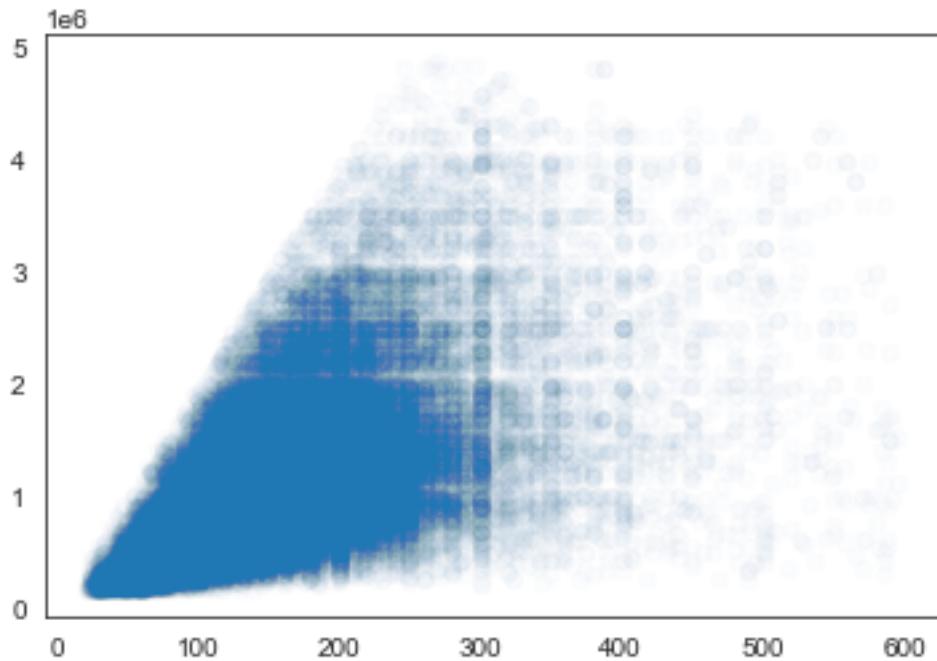
1 Vorraussetzung Kaufpreis von Wohnobjekten

2 Linear Regression

Als erstes machen wir eine Lineare Regression (Ordinary least squares Linear Regression) auf AreaLiving und schauen nach, wie gut der PurchasePrice vorhergesagt ist. Als Metric benutzen wir den MAPE (Mean Absolute Percentage Error):

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

A_t ist der korrekte Preis und F_t unser kalkulierter Preis.



Wir sehen schon anhand des Plots, dass die Prediction nicht gut sein wird. Es hat zuviel Varianz. Wir werden immer, falls möglich, die Daten wie folgt präprozessieren: - den Logarithmus vom PurchasePrice nehmen um diesen besser normalverteilt zu machen. - Onehot-Encoden: Kategorische

Daten oder Strings werden Onehot-encoded. Dies bedeutet, dass ein Attribut in mehrere Attribute binär aufgeteilt wird und diese neuen Attribute sagen, ob dieser Eintrag ein String oder Kategorie enthält. - Label-Encoden: (Gleiche) Strings werden durch (gleiche) Zahlen ersetzt. - Standardisieren auf die Z-Score: $z = \frac{x-\mu}{\sigma}$. (σ ist entweder die Standardabweichung oder das 75% Quantil subtrahiert vom 25% Quantil.) - NaNs auffüllen (entweder mit Mean oder Median oder mit einem bestimmten String). - ggf. auf allen geskewten Werten eine Boxcox-Transformation durchführen, wobei dies bei unseren Test das Ergebniss nicht verbessert hat. - ‘StreetAndNr’, ‘LastUpdate’, ‘Locality’, ‘Id’ werden gedropped; ‘StreetAndNr’ und ‘LastUpdate’ haben zuviele, einzigartige Werte und ‘Locality’ ist bereits im Zipcode vorhanden. - Falls ein Attribut nur Werte zwischen 0 und 1 hat, wird ein neues Attribut kreiert, welches besagt, dass dieses Attribut 0 ist. - Falls ein Attribut NaNs hatte, wird ein neues Attribut erstellt, welches zeigt, wo ein Attribut NaNs hatte.

In den Daten haben wir auch Ungereihmheiten entdeckt: - 2 Einträge haben oft keine Werte. (2) - Es hat Einträge mit Etagen von über 41, dabei ist die Bar im Roche-Tower auf Etage 41 die höchste Etage der Schweiz. (739) - Es hat Wohnobjekten mit einem Baujahr nach dem Renovationsjahr. (23)

Diese tauchen aber auch im Testset auf, welches wir für die Submission auf Kaggle benutzen müssen. Somit lassen wir diese drin, kennmarken diese aber.

Diese Prozessierung wird grösstenteils mit Hilfe einer Klasse (preprocessor) durchgeführt.

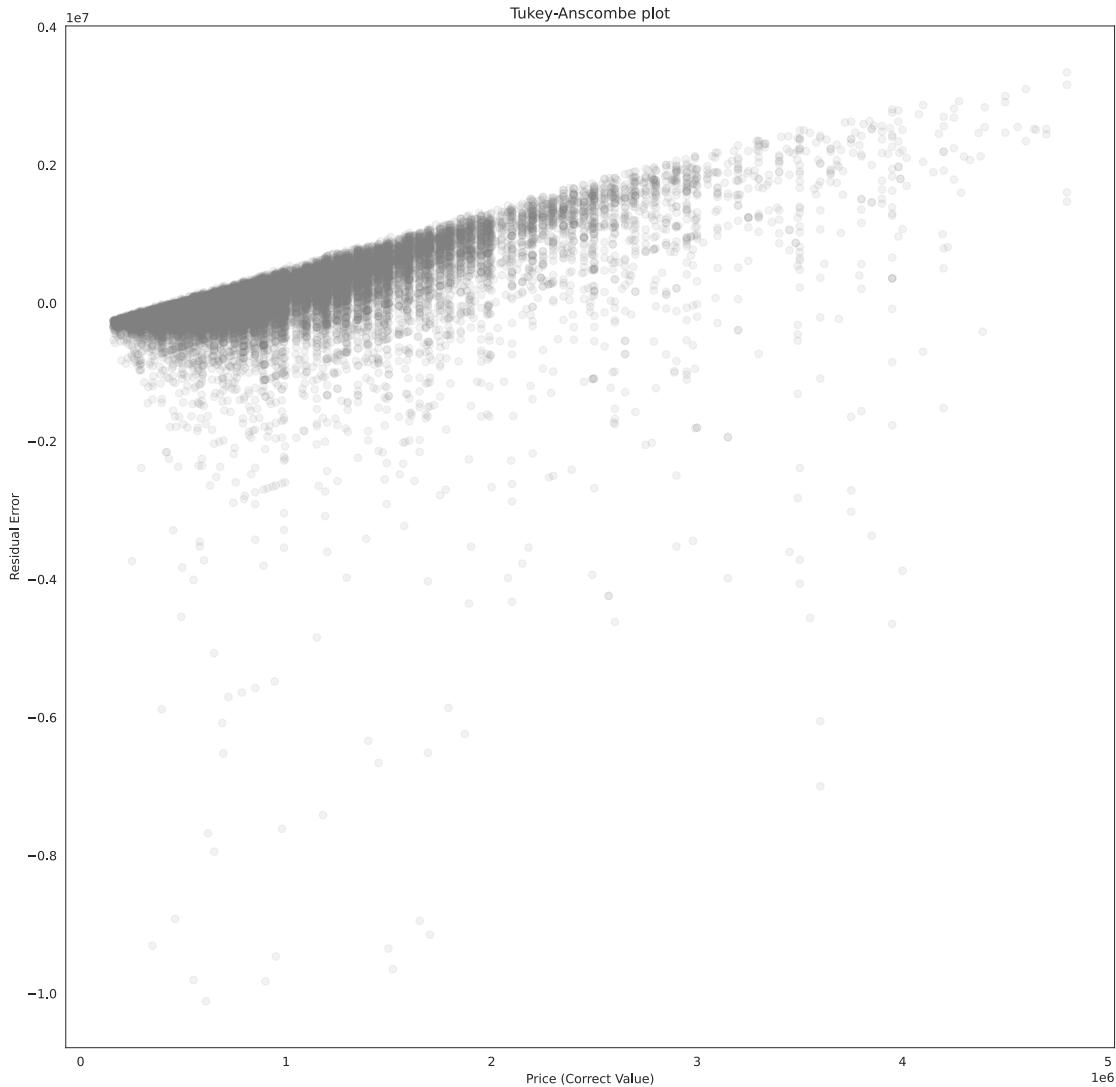
R2-Score: 0.09057998418283497. MAPE: 0.3555584880825077

Eine MAPE von 35% ist extrem schlecht. Wir machen eine Residuenanalyse, um zu sehen, was wir verbessern können.

2.1 Residuen Analyse

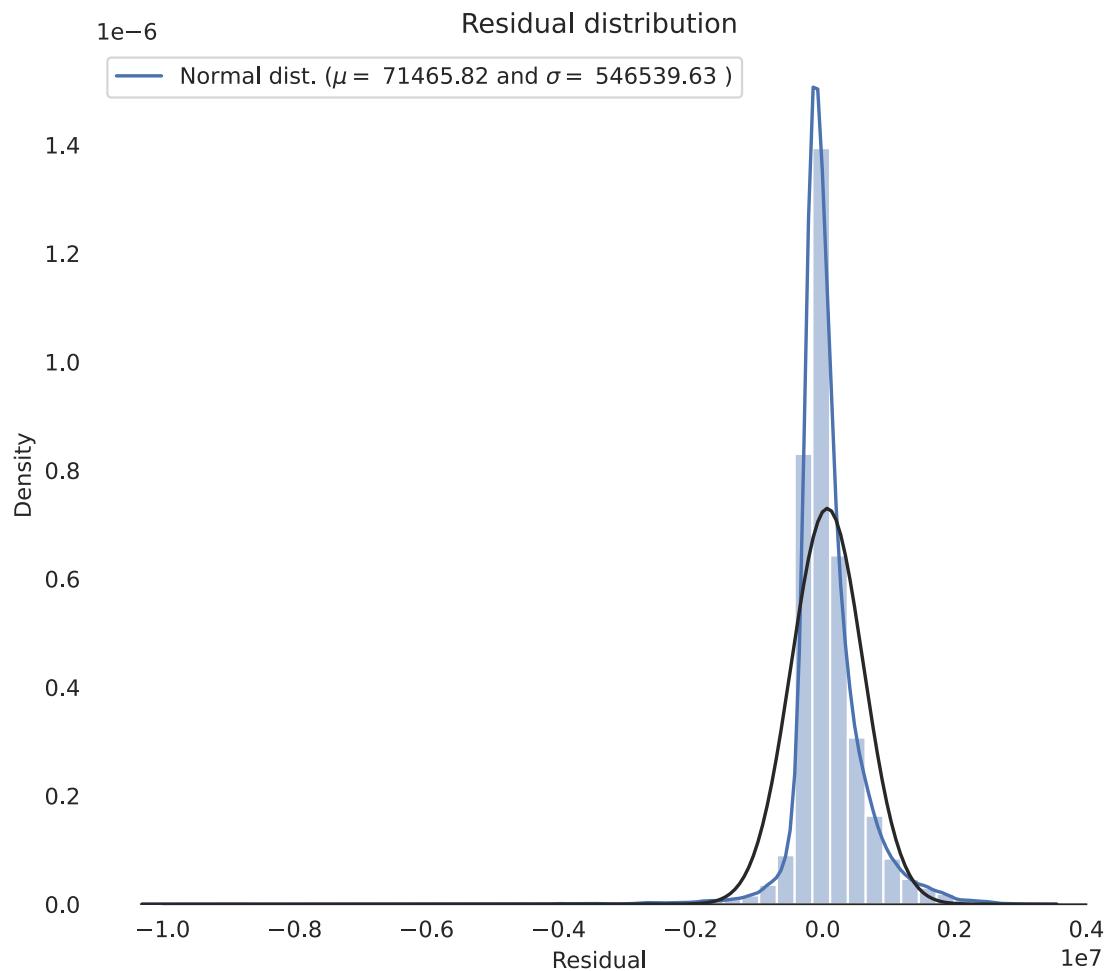
Die Residuen werden dadurch gebildet, dass man die Differenz zwischen dem richtigen Preis und dem vorhergesagten Preis analysiert. Für eine Lineare Regression müssen folgende Punkte erfüllt sein:

- Linearität: Die Beziehung zwischen X und dem Mittelwert von Y ist linear.
- Homoskedastizität: Die Varianz der Residuen ist für jeden Wert von X gleich.
- Unabhängigkeit: Die Beobachtungen sind unabhängig voneinander.
- Normalität: Für jeden Wert von X ist Y normalverteilt.

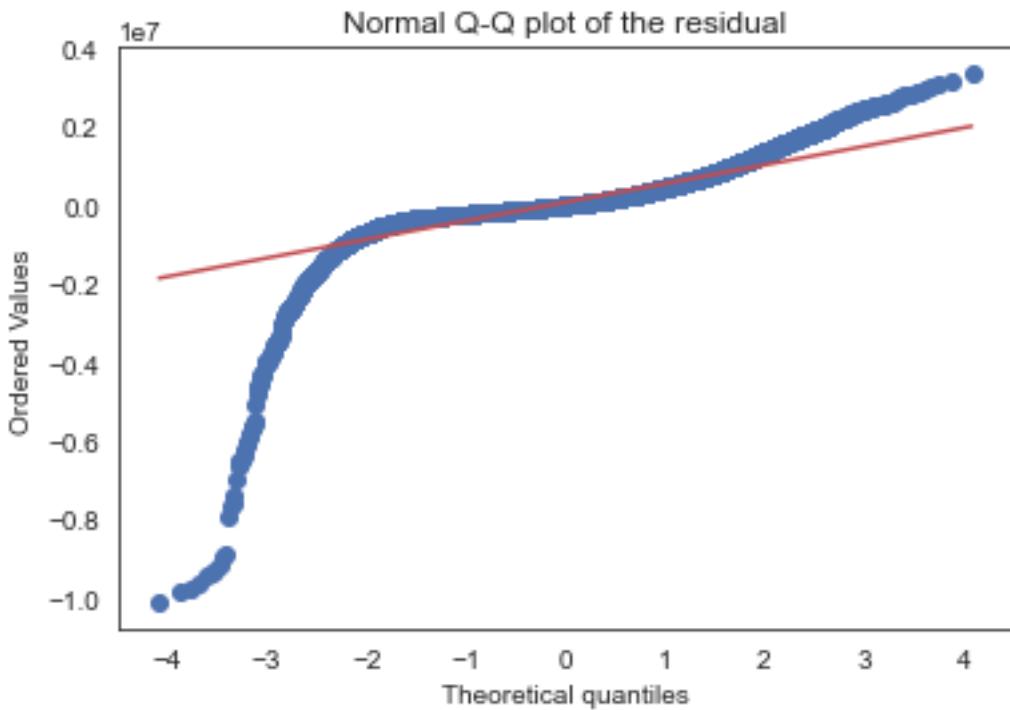


Im oberen Plot sieht man die Residuen (y-Achse) und den richtigen Wert (x-Achse). Man sieht, dass die Residuen nicht um $y=0$ normalverteilt sind und das es unerklärte Varianz hat, da der Fehler mit der Zunahme des Preises grösser wird.

`mu = 71465.82 and sigma = 546539.63`



Im oberen Plot sieht man noch besser, dass die Werte nicht normalverteilt sind. Natürlich lässt sich das auch mit einem QQ-Plot zeigen. Der QQ-Plot vergleicht die Quantile der Normalverteilung (x-Achse) mit den Quantilen der Residuen (y-Achse). Wären die Residuen normalverteilt, wären die Quantile gleich und würden somit eine perfekte gerade bilden.



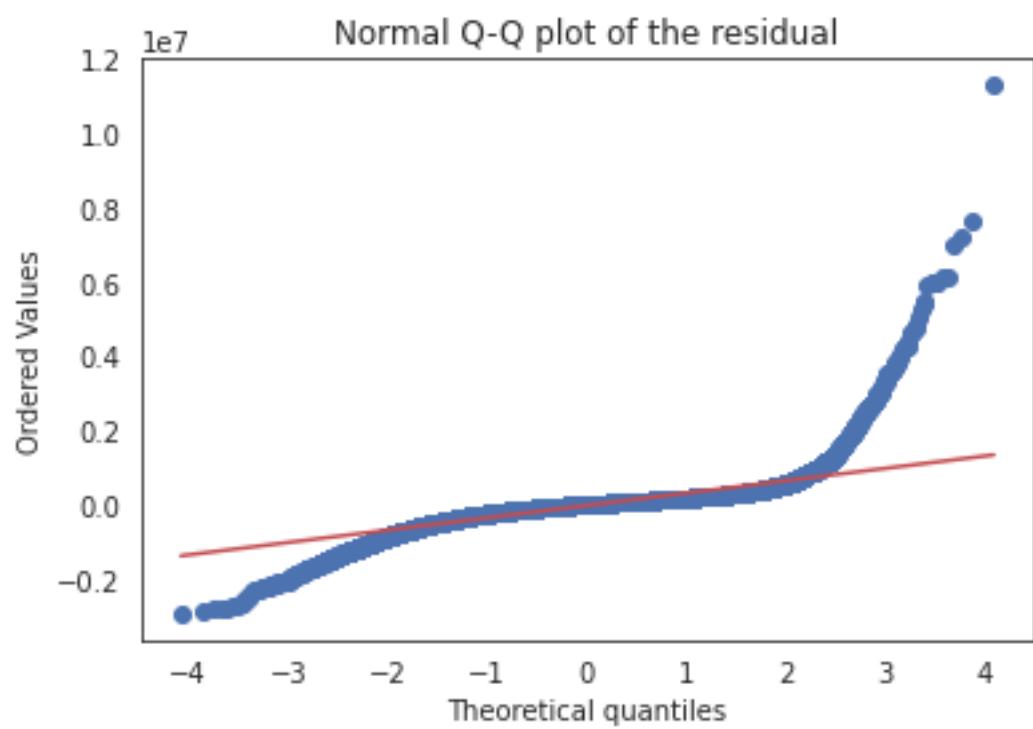
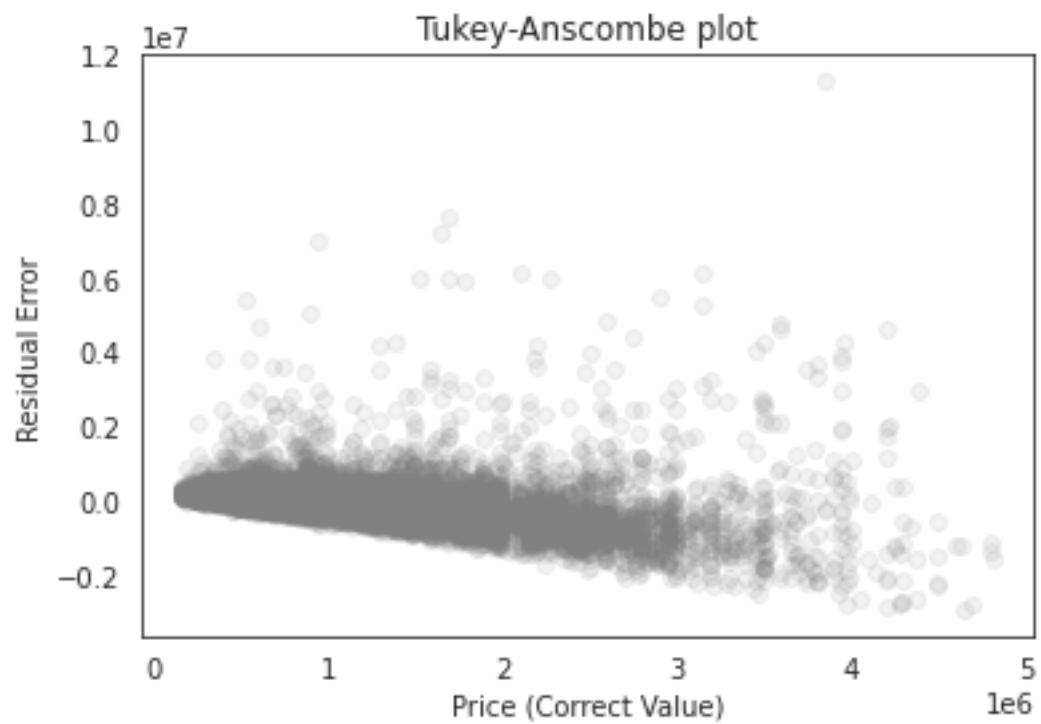
[48]: 1.2383109383385777

Man kann anhand Durbin Watson sehen, dass es eine positive Autokorrelation hat (also die Residuen, sortiert nach dem richtigen Preis, haben eine Autokorrelation), dass die Varianz also zunimmt mit "Area Living". Wäre der Wert um 2, wären die Residuen perfekt unkorreliert.

Wir können mit einem Linearen Modell also nicht die Varianz komplett erklären. Wir versuchen also nun mehr Varianz mit zusätzlichen Attributen zu erklären; Wir fügen Zip-Nummer (durch 100 geteilt und runter-gerundet) Onehot-encoded dazu.

R2-Score: 0.5236758074650314. MAPE: 0.23272362717755254

Ein MAPE von 23% ist eine grosse Verbesserung! Wie sehen die Residuen im Tukey-Anscombe-Plot und im QQ-Plot aus?



Sieht besser aus. Trotzdem sieht man klar, dass unerklärte Varianz vorhanden ist. Ebenfalls scheint es, als würde unser Model den PurchasePrice sehr oft zu hoch schätzen. Es hat also noch Abweichung, was unser Model nicht erklären kann.

2.2 Feature Selection mit Hilfe von EDA

Im EDA-Teil haben wir gesehen, dass nur wenige Features überhaupt positiv oder negativ mit dem Preis korrelieren. Dies bedeutet: wir sollten ein ähnlich gutes Model mit nur wenigen Features herstellen können. Ebenfalls benutzen wir die Attribute, von denen wir die Korrelation nicht berechnen können, also die “Onehot-encoded”-Attribute.

25 Attributes having a Spearman-coefficient above 0.1 with PurchasePrice:

```
[18]: ['AreaLiving',
       'AreaProperty',
       'BuiltYear',
       'HouseObject',
       'PopulationDensityL',
       'RealEstateTypeId',
       'Rooms',
       'TravelTimeMiv',
       'WorkplaceDensityL',
       'WorkplaceDensityM',
       'gde_area_nonproductive_percentage',
       'gde_area_settlement_percentage',
       'gde_empty_apartments',
       'gde_foreigners_percentage',
       'gde_politics_bdp',
       'gde_politics_fdp',
       'gde_politics_gps',
       'gde_politics_svp',
       'gde_pop_per_km2',
       'gde_population',
       'gde_private_apartments',
       'gde_tax',
       'gde_workers_sector3',
       'gde_workers_total',
       'PurchasePrice']
```

R2-Score: 0.4750141939802949. MAPE: 0.2444058045075745

Mit 42 weniger Attributen (vor Onehot-Encoding) erreichen wir fast dasselbe Ergebnis!

3 XGB-Regressor

Man hat bei der Residuenanalyse gesehen, dass die Lineare Regression unerklärbare Varianz hat. Wir müssen also ein anderes Model benutzen. Wir versuchen also einen XGBoost-Regressor. XG-Boost steht für eXtreme Gradient Boosting. Es ist also ein Decision-Tree Algorithmus, welcher seine Genauigkeit mit weiteren Bäumen verbessert. Dies passiert durch das reduzieren einer von

uns gewählten Object-Function. Standardmäßig nimmt der XGBoost-Regressor den MSE zwischen dem kalkulierten Preis und dem richtigen Preis.

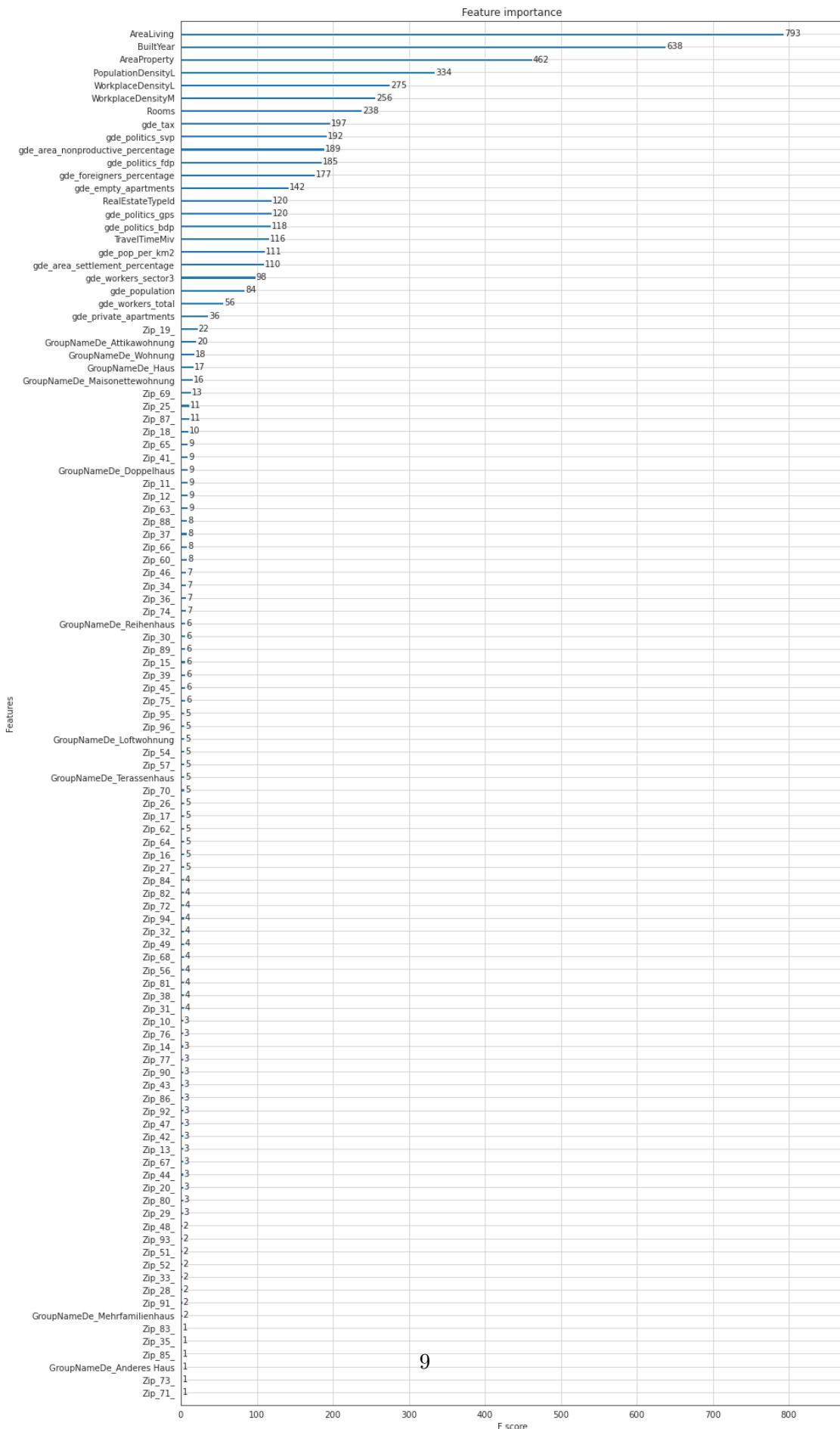
R2-Score: 0.8117852093958533. MAPE: 0.16745824960560005

Eine starke Verbesserung! Ohne jegliche Parameteroptimierungen und mit massiv reduzierten Daten. Im nächsten Schritt sehen wir die Wichtigkeit der Attribute.

Die Wichtigkeitstabelle stellt dar, wie wertvoll ein Feature in der Konstruktion des ‘Boosted Decision Tree’-Modells ist: je häufiger ein Attribut für eine Entscheidung genutzt wird, desto höher ist seine relative Wichtigkeit. So können die Attribute sortiert und miteinander verglichen werden.

Die Wichtigkeit wird für einen einzelnen Entscheidungsbaum durch den Wert berechnet, um den jeder Attribut-Aufteilungspunkt das Leistungsmass verbessert – gewichtet nach der Anzahl der Beobachtungen, für die der Knoten verantwortlich ist.

[24]: <matplotlib.axes._subplots.AxesSubplot at 0x41a6b27910>



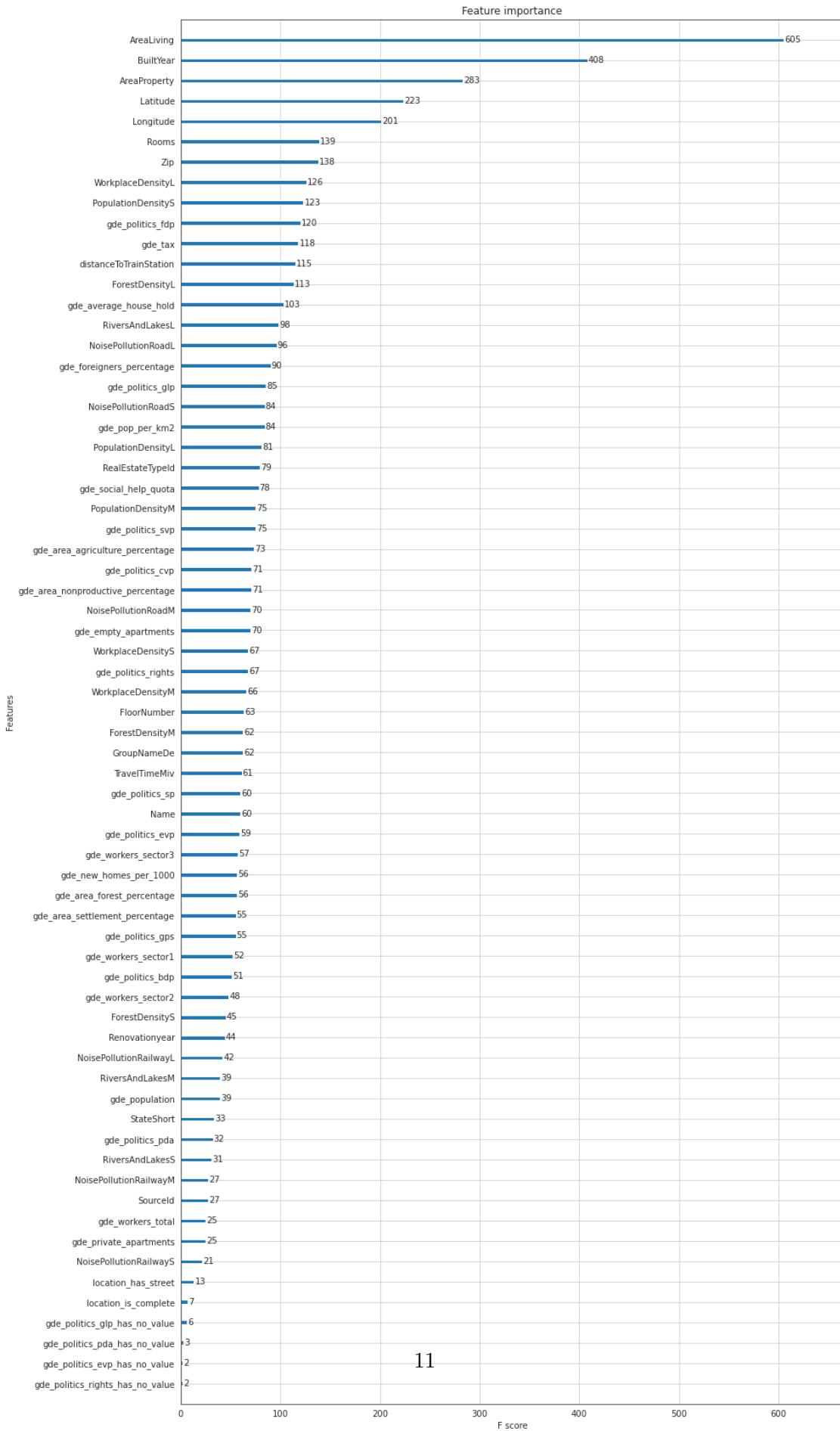
Das Ergebniss ist akzeptabel. Da es ‘Boosted Random Forest Tree’ ist, müssen wir die Features nicht notwendigerweise Onehot-encoden. Wir können Objekte auch Label-encoden und so RAM einsparen.

```
/opt/conda/lib/python3.8/site-packages/sklearn/utils/validation.py:72:  
DataConversionWarning: A column-vector y was passed when a 1d array was  
expected. Please change the shape of y to (n_samples, ), for example using  
ravel().  
    return f(**kwargs)
```

R2-Score: 0.8228875532196948. MAPE: 0.15948104328521964

Hier nochmals die Wichtigkeit der Attribute:

```
[28]: <matplotlib.axes._subplots.AxesSubplot at 0x40c6f80a60>
```



Sieht schon gut aus! Wir können nun noch die Parameter optimieren!

4 Parameteroptimierung von XGBoost mit Hilfe der Bayes'scher Inferenz und Gauß-Prozess

Wie sieht es mit XGB-Regression aus, falls wir Parameter optimieren? Zur Optimierung benutzen wir eine Methode, welche auf Bayes'scher Inferenz und Gauß-Prozess aufbaut und versucht, den Maximalwert einer unbekannten Funktion in so wenigen Iterationen wie möglich zu finden.

Die Bayes'sche Optimierung funktioniert, indem eine Verteilung von Funktionen (Gauß-Prozess) konstruiert wird, die die zu optimierende Funktion am besten beschreibt. Wenn die Anzahl der Beobachtungen wächst, wird die Verteilung genauer und der Algorithmus wird immer sicherer, welche Regionen im Parameterraum (nicht) wertvoll zu erforschen sind.

Ebenfalls definieren wir eine neue Objective-Function für den XGBoost-Regressor. Die Aufgabe war ja, den MAPE zu minimieren. Somit geben wir dem XGBoost Regressor diese Aufgabe und die Bayes'sche Optimierung soll die besten Parameter für einen möglichst niedrigen MAPE-Wert finden.

Um Bayesian-Optimization durchzuführen, müssen wir zuerst Folgendes machen: - die Daten vorbereiten, - eine Funktion beschreiben, welche maximiert werden kann, - sinnvolle Parameterräume definieren, - die Anzahl Iterationen definieren und Anzahl Probepunkte.

Ebenfalls machen wir noch etwas Feature-Engineering: - wir erstellen das Attribut "Sq_per_room", das besagt, wieviel ein Wohnobjekt an Wohnraum pro Raum hat, - wir teilen die Postleitzahl nicht mehr durch 100, sondern nur noch durch 10 und Onehot-encoden dies, - wir subtrahieren vom jetzigen Jahr das Baujahr und das Renovationsjahr, da für die Maschine eher interessant ist, wie lange das Bau- bzw. die Renovationsjahr her ist.

Folgende Parameter werden optimiert: - learning_rate: Wie schnell das Model das Gelernte anwendet, - colsample_bytree: Welche Fraktion der Attribute für die jetzige Iteration benutzt wird. Verhindert/Reduziert Overfitting, - subsample: Welche Fraktion der Reihen für die jetzige Iteration benutzt wird. Verhindert/Reduziert Overfitting, - max_depth: Maximale Tiefe des Baumes. Verhindert/Reduziert Overfitting, - n_estimators: Wieviele Decisiontrees erstellt werden, - gradient_mult_mape: Parameter für die MAPE-objective Funktion.

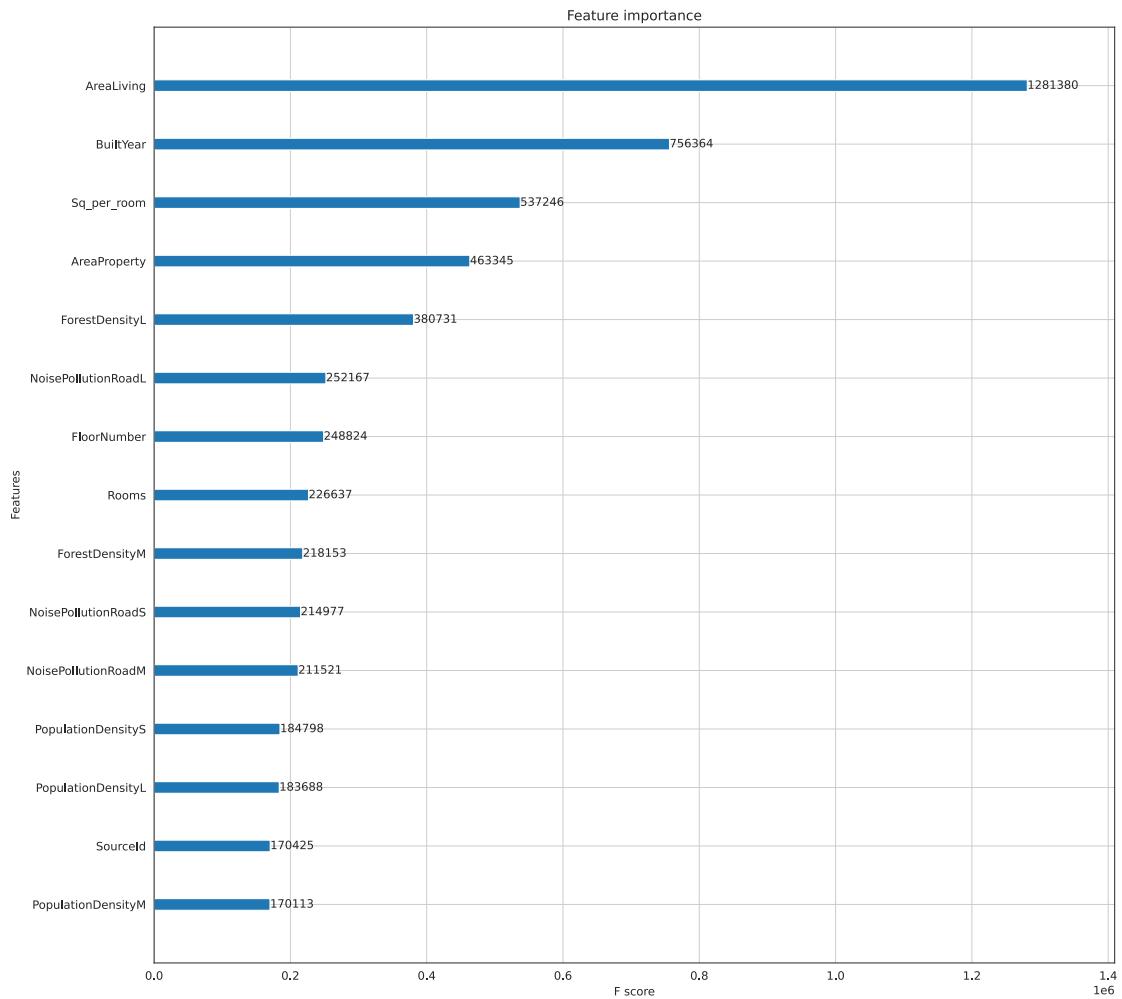
Ebenfalls, da Pandas nicht intelligent mit den Typen umgeht, reduzieren wir die Grösse der Daten im Speicher mit der Klasse Reducer.

```
reduced df from 112.0678 MB to 73.4443 MB in 18.76 seconds
reduced df from 12.5211 MB to 8.1602 MB in 1.83 seconds
```

```
Starting the optimization from scratch.
| iter      | target    | colsam... | gradie... | learni... | max_depth |
n_esti... | subsample |
-----
```

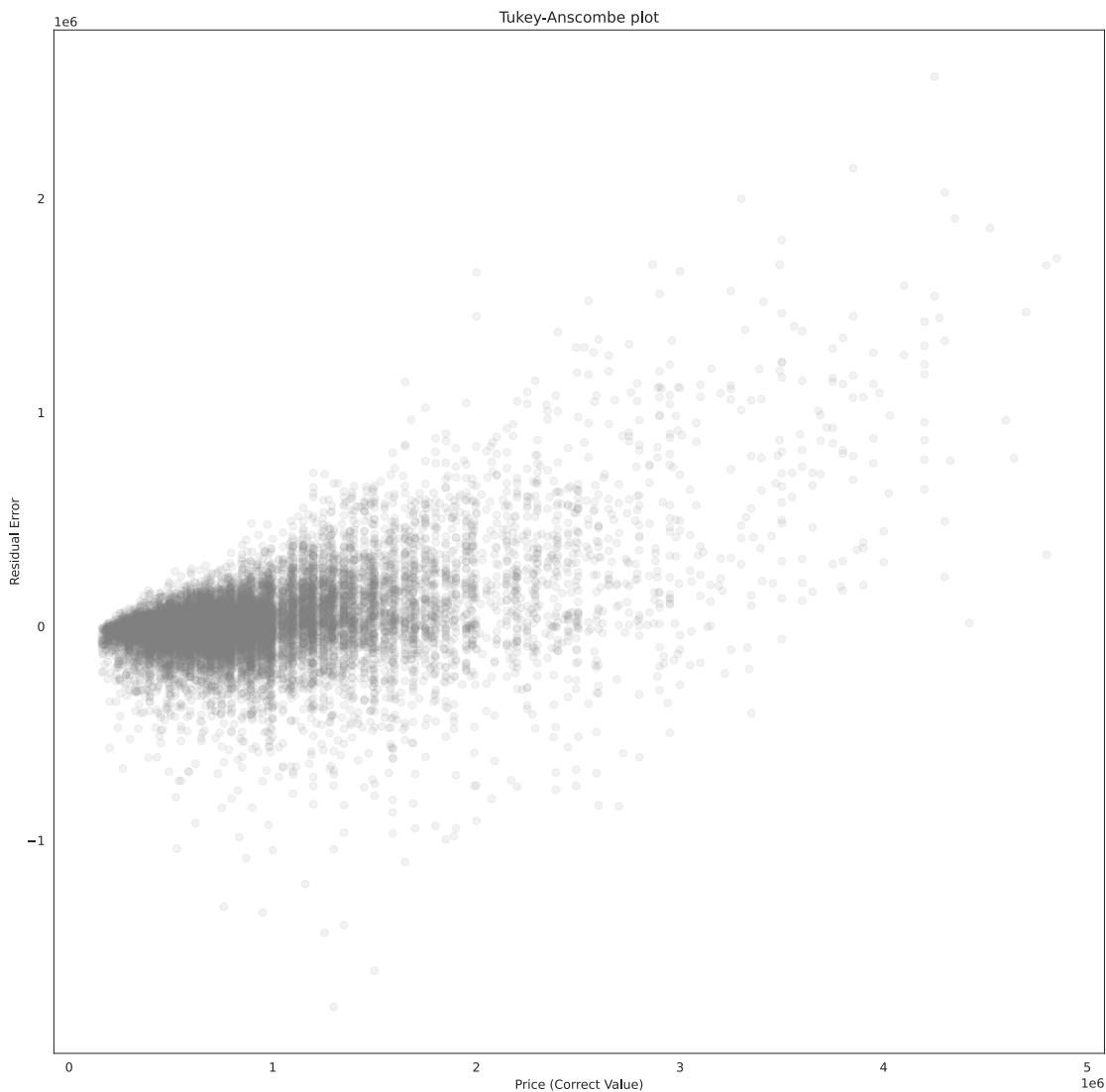
```
reduced df from 112.0516 MB to 73.4331 MB in 5.68 seconds
reduced df from 12.5195 MB to 8.1592 MB in 0.50 seconds
R2-Score: 0.8657933058420197. MAPE: 0.12985230216045093
```

[27]: <AxesSubplot:title={'center':'Feature importance'}, xlabel='F score', ylabel='Features'>

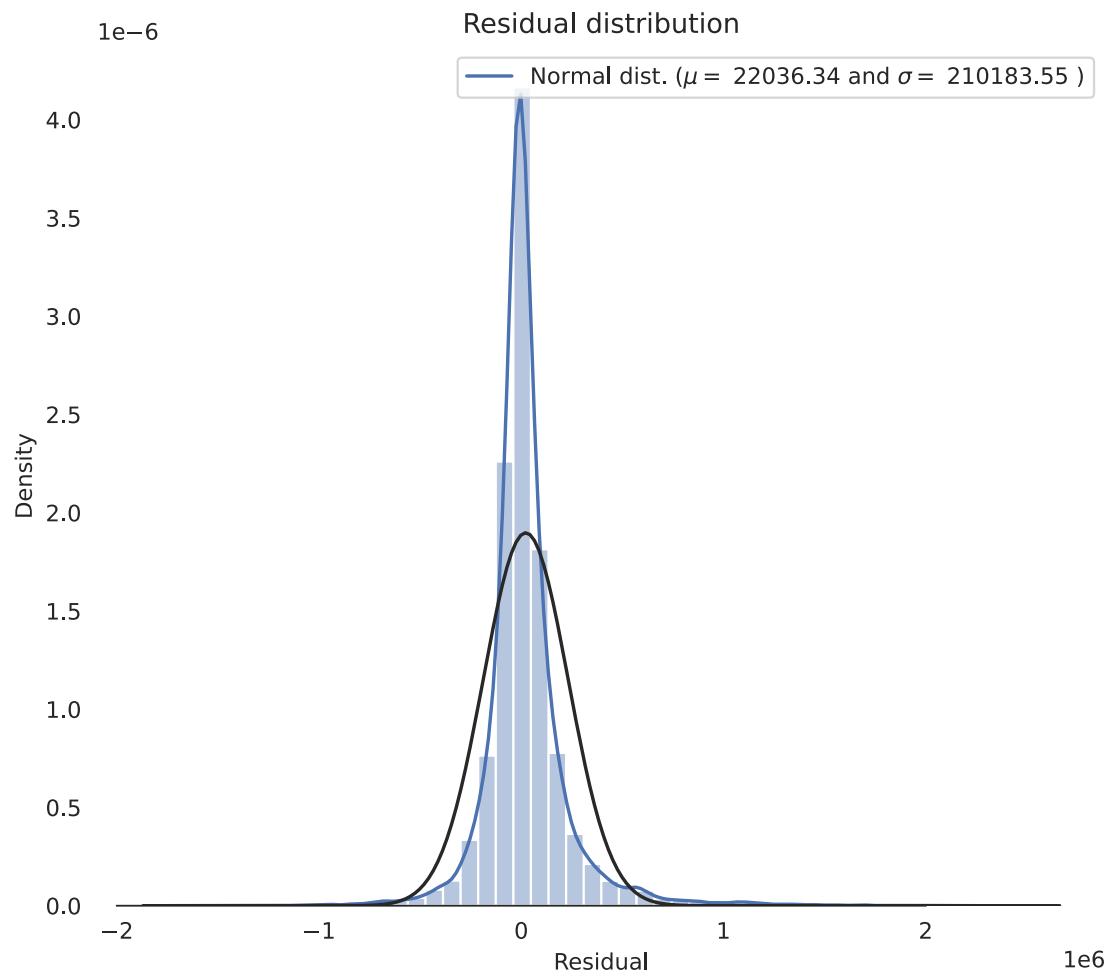


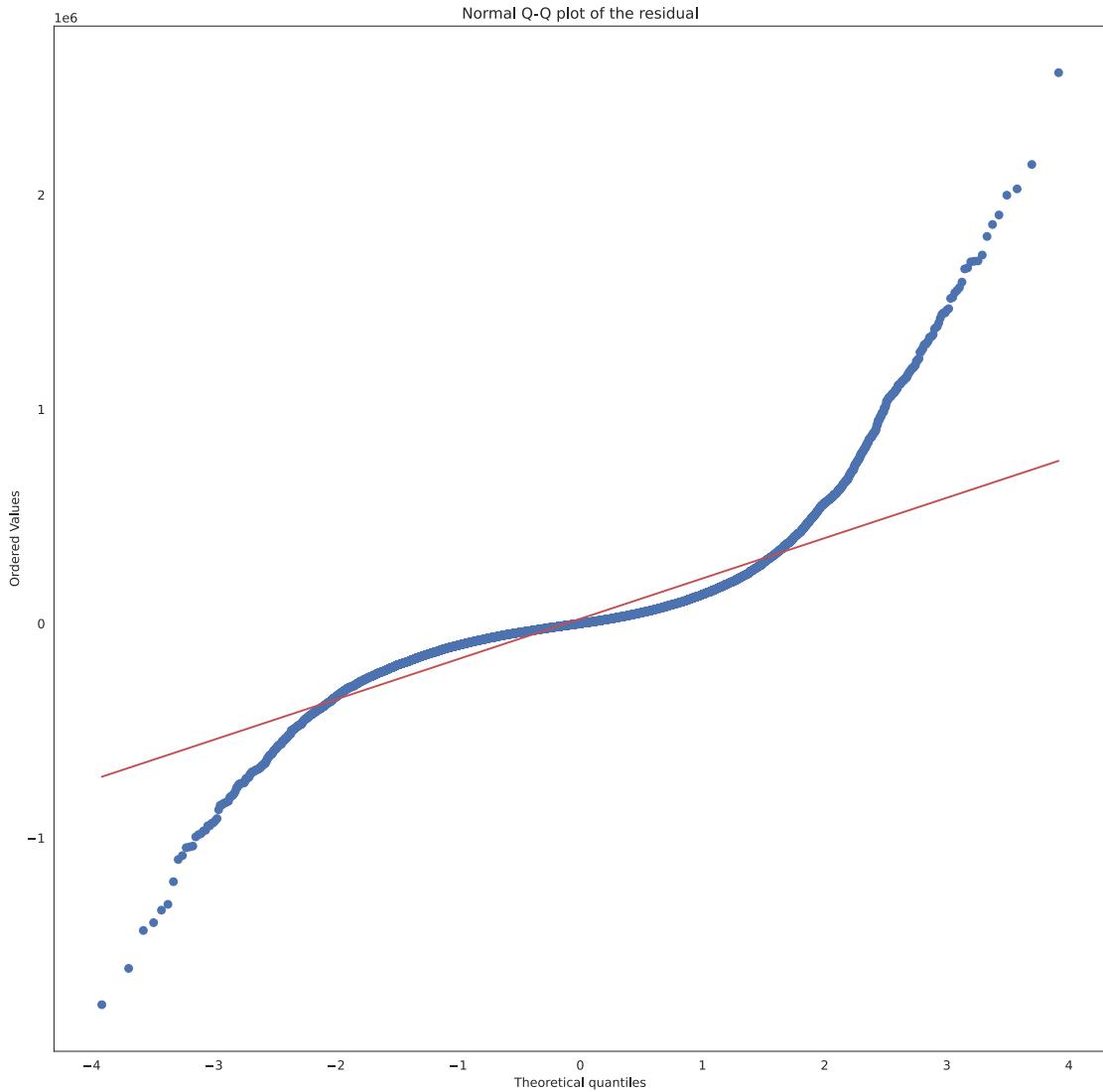
4.1 Residuenanalyse

Wir können uns die Residuen vom besten Model anschauen, um zu sehen, ob wir ein Muster in den Residuen entdecken. Dies würde uns sagen, ob wir noch etwas verbessern könnten.



`mu = 22036.34 and sigma = 210183.55`





[37] : 1.4326169109577225

Wir haben bei den Residuen gesehen, dass es besser ist. Es ist aber noch abzulesen, dass die Varianz mit der grösste des Purchase-Price zunimmt. Es hat also noch unerklärte Varianz. Dies sieht man auch bei durbin_watson; Ein Wert um 2 wäre perfekt.

5 Hochladen des Test-Sets auf Kaggle

Da wir nun das bestmögliche Model gefunden haben, können wir die Preise kalkulieren und diese auf Kaggle hochladen:

6 LGBM-Regressor

Wir sehen, dass das Model von XGBoost sehr gross wird (600 MB +) und die Trainingszeit sehr lange ist (15 Minuten). Da ist der LGBM-Regressor bestimmt interessant, da er in der Leistung ähnlich ist, aber bekannterweise schneller trainiert wird und ein kleineres Modell besitzt. Im Gegensatz dazu, braucht der LGBM-Regressor pro Fit ca. 20 Sekunden und das Model ist mit etwa 60MB rund 10 Mal kleiner. Leider ist auch der MAPE mit ungefähr 13.2% fast ein halbes Prozent schlechter. Die viel schnellere Laufzeit und das viel kleinere Model macht LGBM trotzdem zu einer sehr attraktiven Alternative.