AirBnB
Berlin

Price Optimisation of listings for Hosts

AirBnB Terms

Hosts

Owner of the property

AirBnB Terms

Listing

Eg. Apartments, single rooms, houseboats, castle etc listings_summary.csv

listings.csv

reviews_summary.csv

~~reviews.csv~~

~~calendar_summary.csv~~

~~neighbourhoods.csv~~

Business Case

Price optimisation of listings for Hosts

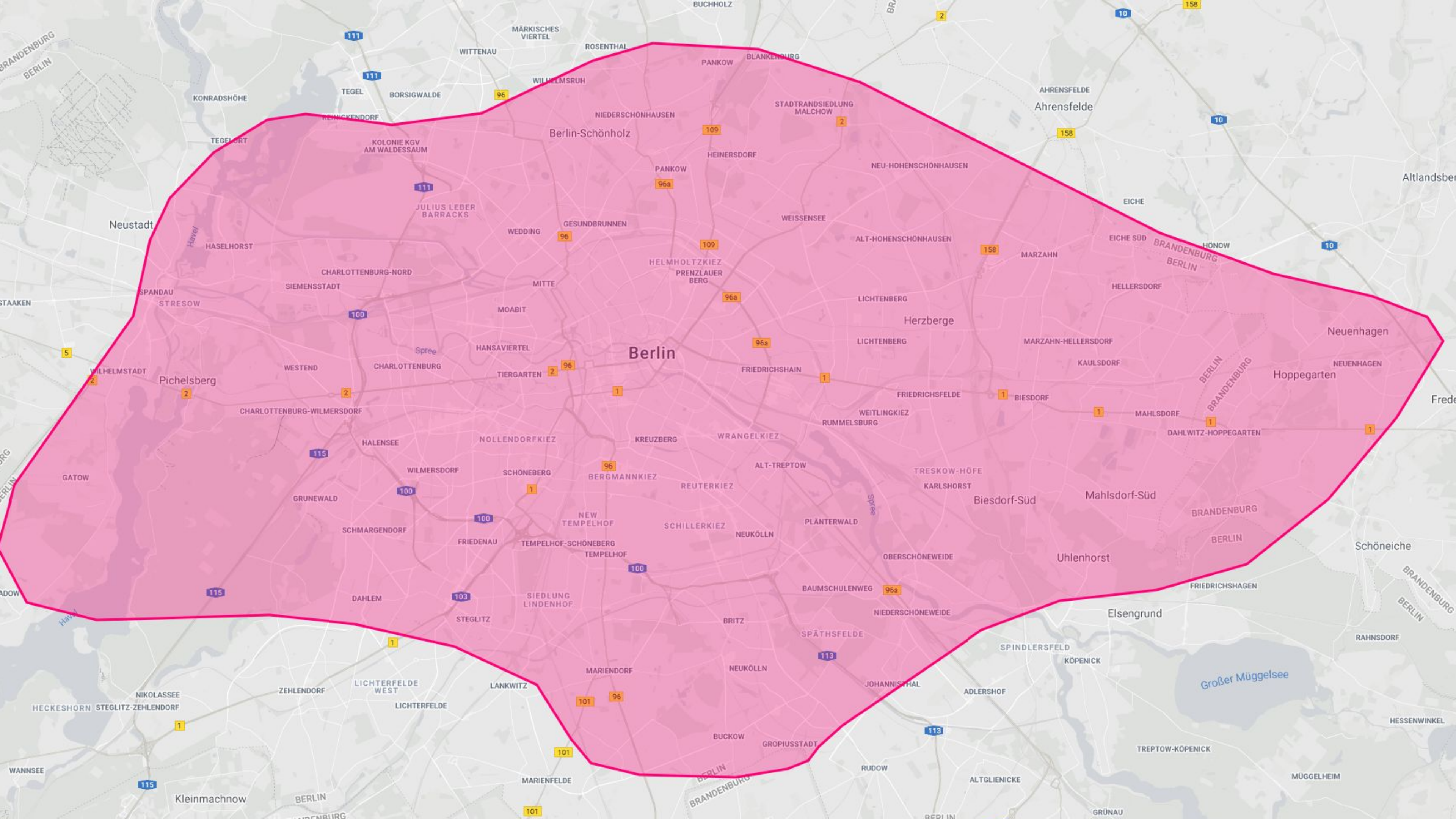
Machine Learning Model
Random Forest Regressor

Listing Summary CSV
EDA
Principal Component Analysis
MissForest

Review Summary CSV
Language Identification and Cleaning using NLTK
Sentiment analysis of reviews using Vader
WordCloud using wordcloud

Neighbourhood Data (From Listing CSV)
Geographical Analysis and Visualisations using GeoPandas

Why optimise?



Berlin
Berlin-Schönholz
Herzberge
Pichelsberg
Neustadt
Ahrensfelde
Neuenhagen
Hoppegarten
Biesdorf-Süd
Mahlsdorf-Süd
Uhlenhorst
Elsengrund
Schöneiche
Kleinmachnow
Altlandsberg
Großer Müggelsee
Spree
Havel
BUCHHOLZ
MÄRKISCHES VIERTEL
WITTENAU
ROSENTHAL
PANKOW
BLANKENBURG
WILHELMSRUH
TEGEL
BORSIGWALDE
KONRADSHÖHE
REINICKENDORF
TEGEL ORT
NIEDERSCHÖNHAUSEN
STADTRANDSIEDLUNG MALCHOW
KOLONIE KGV AM WALDESSAUM
HEINERSDORF
NEU-HOHENSCHÖNHAUSEN
AHRENSFELDE
EICHE
JULIUS LEBER BARRACKS
GESUNDBRUNNEN
WEISSENSEE
WEDDING
ALT-HOHENSCHÖNHAUSEN
MARZAHN
EICHE SÜD
HÖNOW
HASELHORST
HELMHOLTZKIEZ
PRENZLAUER BERG
CHARLOTTENBURG-NORD
SIEMENSSTADT
MITTE
HELLERSDORF
SPANDAU
STRESOW
STAAKEN
MOABIT
LICHTENBERG
HANSAVIERTEL
MARZAHN-HELLERSDORF
KAULSDORF
NEUENHAGEN
WILHELMSTADT
WESTEND
CHARLOTTENBURG
TIERGARTEN
FRIEDRICHSHAIN
FRIEDRICHSFELDE
BIESDORF
MAHLSDORF
CHARLOTTENBURG-WILMERSDORF
WEITLINGKIEZ
RUMMELSBURG
DAHLWITZ-HOPPEGARTEN
HALENSEE
NOLLENDORFKIEZ
KREUZBERG
WRANGELKIEZ
GATOW
WILMERSDORF
SCHÖNEBERG
BERGMANNKIEZ
ALT-TREPTOW
TRESKOW-HÖFE
KARLSHORST
GRUNEWALD
REUTERKIEZ
NEW TEMPELHOF
SCHILLERKIEZ
PLÄNTERWALD
SCHMARGENDORF
NEUKÖLLN
FRIEDENAU
TEMPELHOF-SCHÖNEBERG
TEMPELHOF
OBERSCHÖNEWEIDE
BRANDENBURG
FRIEDRICHSHAGEN
SIEDLUNG LINDENHOF
DAHLEM
BAUMSCHULENWEG
NIEDERSCHÖNEWEIDE
STEGLITZ
BRITZ
SPÄTHSFELDE
SPINDLERSFELD
KÖPENICK
RAHNSDORF
MARIENDORF
NEUKÖLLN
JOHANNISTHAL
ADLERSHOF
NIKOLASSEE
ZEHLENDORF
LICHTERFELDE WEST
LANKWITZ
HECKESHORN
STEGLITZ-ZEHLENDORF
LICHTERFELDE
HESSENWINKEL
BUCKOW
GROPIUSSTADT
TREPTOW-KÖPENICK
WANNSEE
RUDOW
ALTGLIENICKE
MÜGGELHEIM
MARIENFELDE
GRÜNAU

22,252

Listings in Berlin

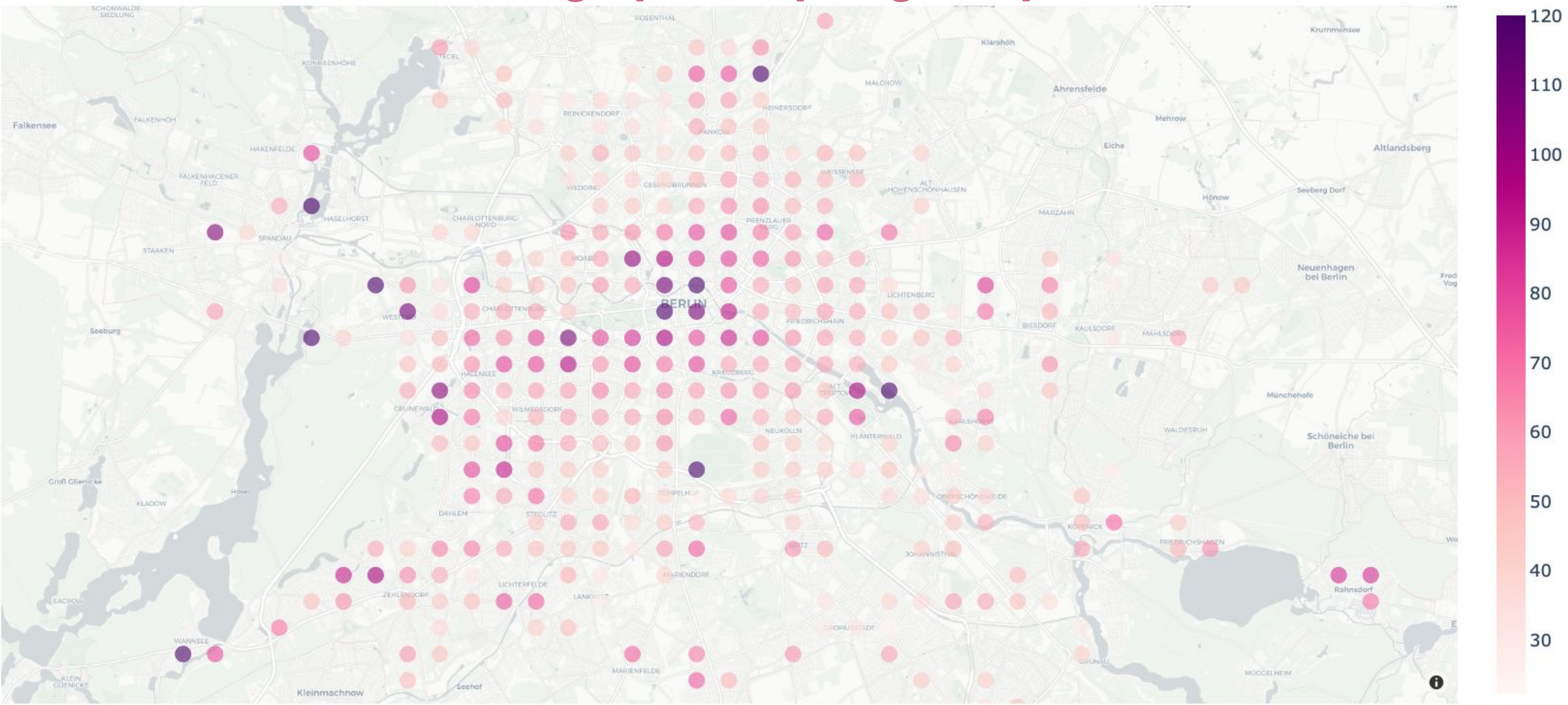
10,700

HDB Blocks

GeoData

With Geopandas

Average prices per grid space



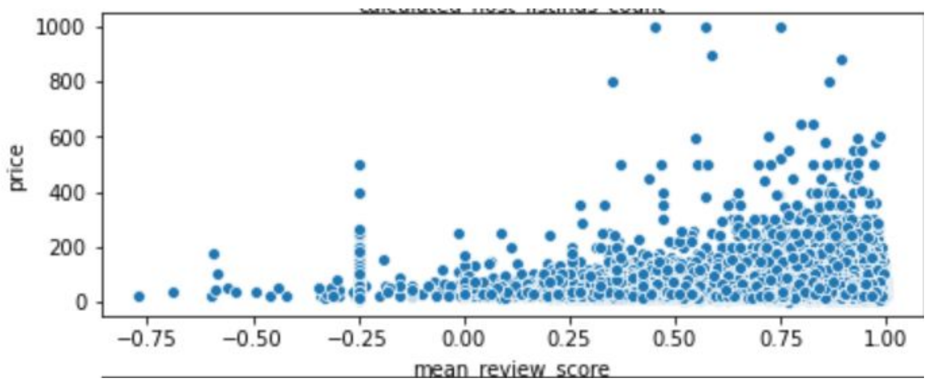
1. Made Berlin into a grid
2. Grouped listings in those grid spaces
3. Took the mean of their prices

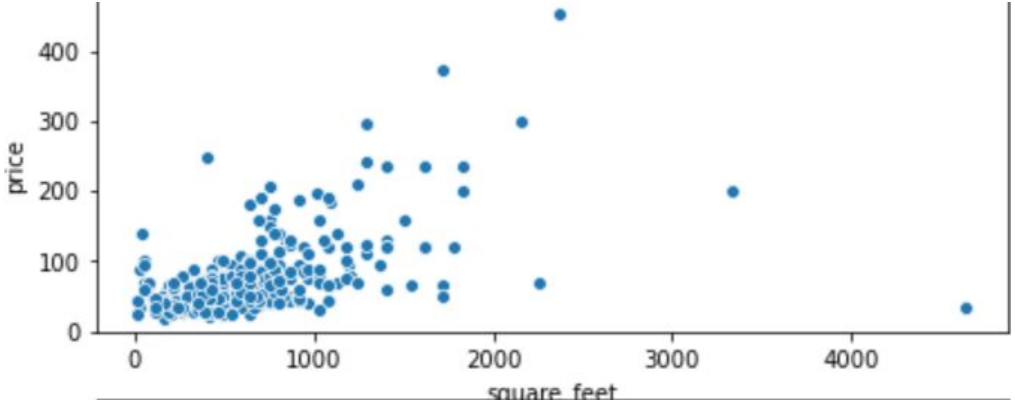
Plot!

EDA

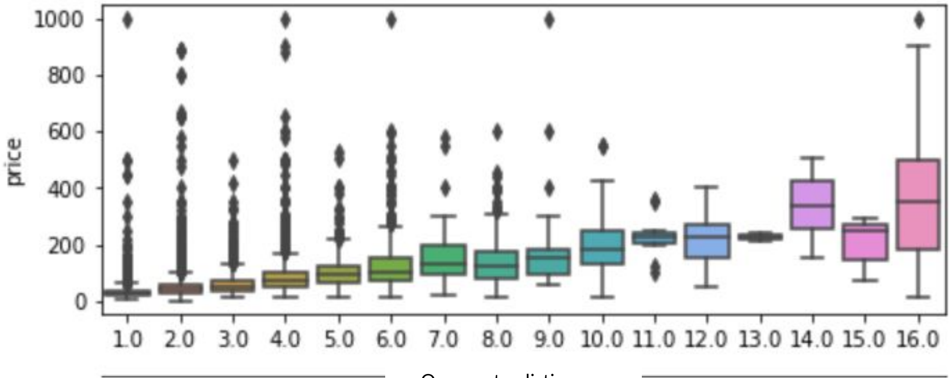
With pandas

EDA

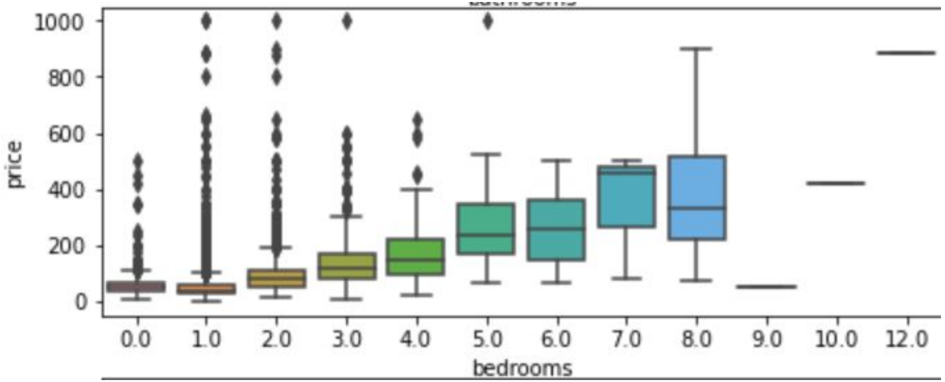

price
1000
800
600
400
200
0
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
mean_review_score

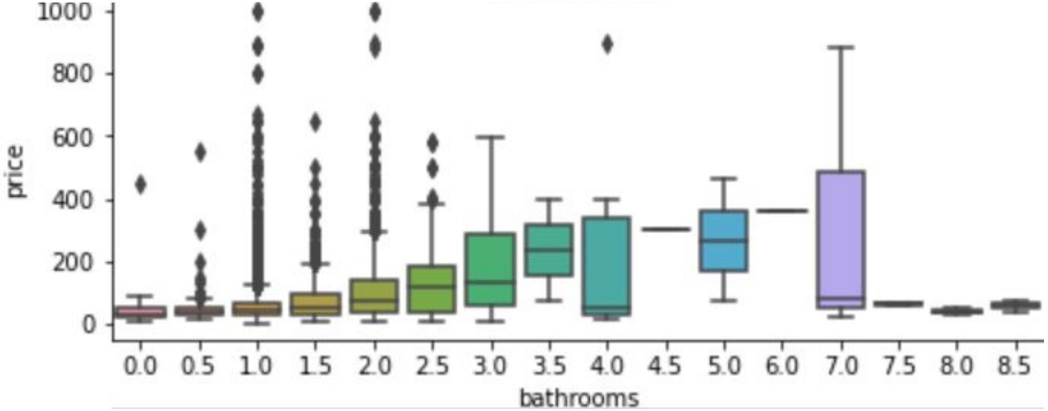

price
400
300
200
100
0
0
1000
2000
3000
4000
square_feet

EDA



Occupants a listing can host





Sentiment Analysis

LangDetect
Natural Language Toolkit (NLTK)
Vader Sentiment Analysis

Natural Language ToolKit (NLTK)

Tokenize Words

Breaks the whole review up into **individual words.**

Words are analysed individually, without context of other words.

Remove STOP words

STOP Words are unimportant, commonly used words with no meaning

Eg. The, A, Might, Various

Reduces noise during analysis, allows us to focus on impactful words

POS Tagging (Part of Speech)

Words are tagged by grammatical group.

Simply put, they are tagged as nouns, verbs, adjectives etc.

Lemmatize Words

Reduces words to their base form.

Running = Run
Feet = Foot
Hotel = trivago

Vader Sentiment Analysis

Vader

Returns sentiment scores as ratios

- Positive
- Negative
- Neutral

Eg. Pos: 0.48, Neu: 0.52

	comments_clean	pos
76596	good good good clean clean clean thx	1.000
8390	great location perfectly clean great value	0.946
83132	warm friendly welcome would definitely recomme...	0.943
41263	truly excellent apartment great welcoming helpful	0.943
148657	great cozy wonderful relax definitely recommen...	0.941

	comments_clean	neg
191318	bad organize rude aggressive host	0.802
134922	staff eve unfriendly rude bad attitude	0.750
142487	every terrible de people rude bad service	0.706
38379	apartment smell bad host behave strangely fina...	0.693
249851	living room nothing special clear nice	0.693

Data Cleaning

120

columns

35

columns

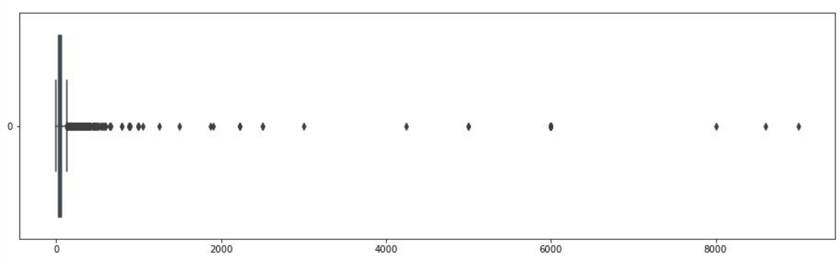
Response Variable(Price)

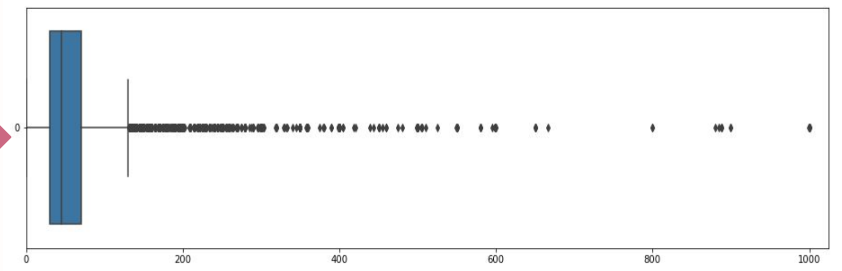
Convert it from string to numeric

- Remove $ sign
- Remove comma(eg: 5,000)

Remove illogical data and some outliers

- Price = 0(Free stay XD??)
- Price > 1000





Predictor Variables

Numeric and Object

Numeric Predictors

Modify Illogical Data

- Square Feet = 0
- Replaced with NaN

Missing Data

accommodates	22507 non-null	float64
bathrooms	22475 non-null	float64
bedrooms	22489 non-null	float64
beds	22467 non-null	float64
square_feet	323 non-null	float64
availability_30	22507 non-null	float64
availability_60	22507 non-null	float64
availability_90	22507 non-null	float64
availability_365	22507 non-null	float64
number_of_reviews	22507 non-null	float64
review_scores_rating	18141 non-null	float64
review_scores_accuracy	18116 non-null	float64
review_scores_cleanliness	18119 non-null	float64
review_scores_checkin	18098 non-null	float64
review_scores_communication	18112 non-null	float64
review_scores_location	18099 non-null	float64
review_scores_value	18095 non-null	float64
minimum_nights	22507 non-null	float64
reviews_per_month	18614 non-null	float64
calculated_host_listings_count	22507 non-null	float64
mean_review_score	17582 non-null	float64
distance_from_central	22462 non-null	float64

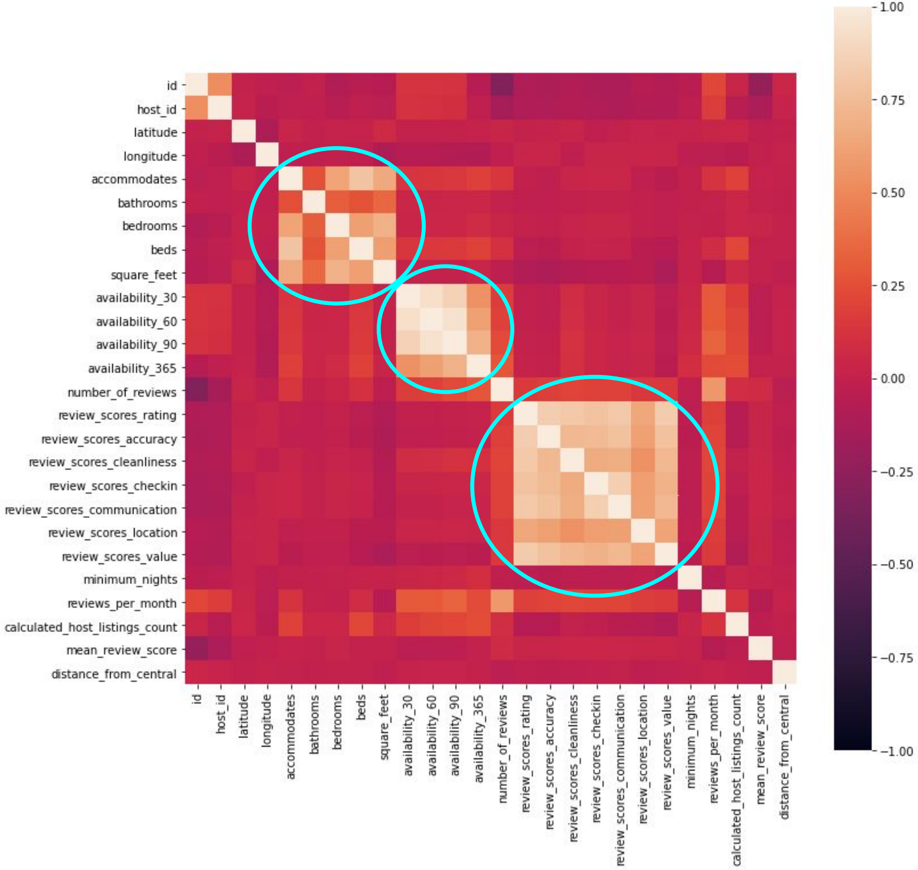
Impute missing ... rests iteratively

accommodates	22507 non-null	float64
bathrooms	22507 non-null	float64
bedrooms	22507 non-null	float64
beds	22507 non-null	float64
square_feet	22507 non-null	float64
availability_30	22507 non-null	float64
availability_60	22507 non-null	float64
availability_90	22507 non-null	float64
availability_365	22507 non-null	float64
number_of_reviews	22507 non-null	float64
review_scores_rating	22507 non-null	float64
review_scores_accuracy	22507 non-null	float64
review_scores_cleanliness	22507 non-null	float64
review_scores_checkin	22507 non-null	float64
review_scores_communication	22507 non-null	float64
review_scores_location	22507 non-null	float64
review_scores_value	22507 non-null	float64
minimum_nights	22507 non-null	float64
reviews_per_month	22507 non-null	float64
calculated_host_listings_count	22507 non-null	float64
mean_review_score	22507 non-null	float64
distance_from_central	22507 non-null	float64

Curse of Dimensionality

Principal Component Analysis (PCA)

linear dimensionality reduction technique



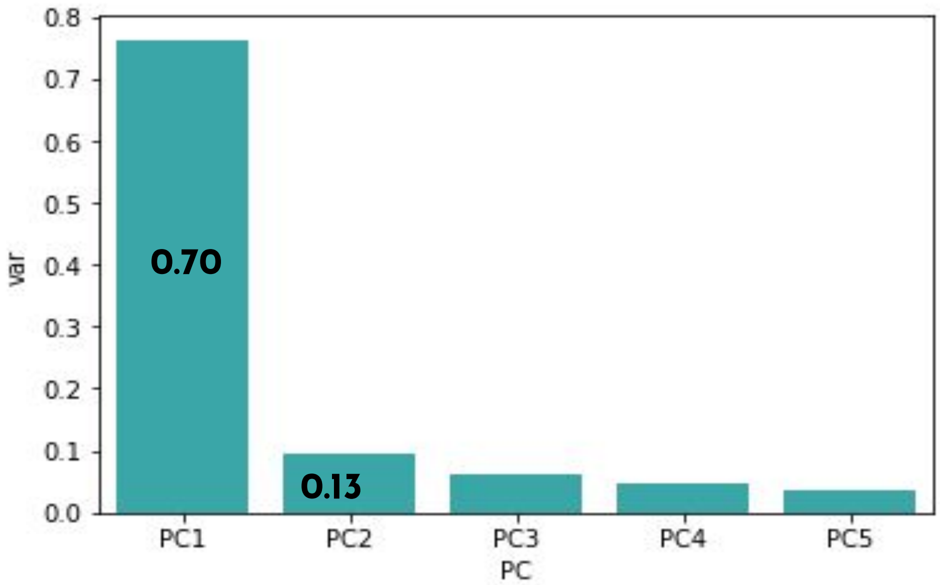
accommodates
bathrooms
bedrooms
beds
square feet

availability 30

Review scores value

NORMALISE

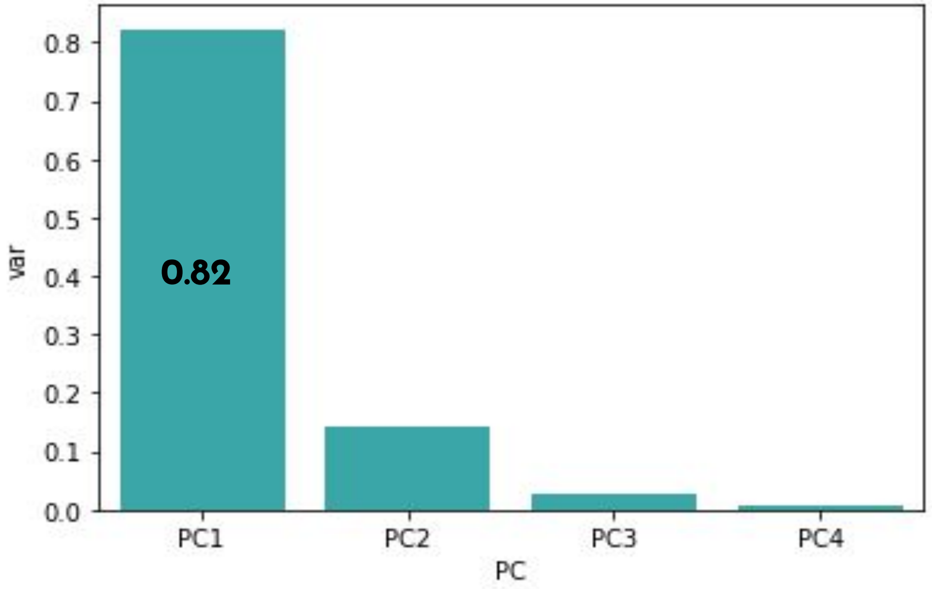
First Cluster(House Attributes)



>0.8 Explained Variance

attribute1, attribute2

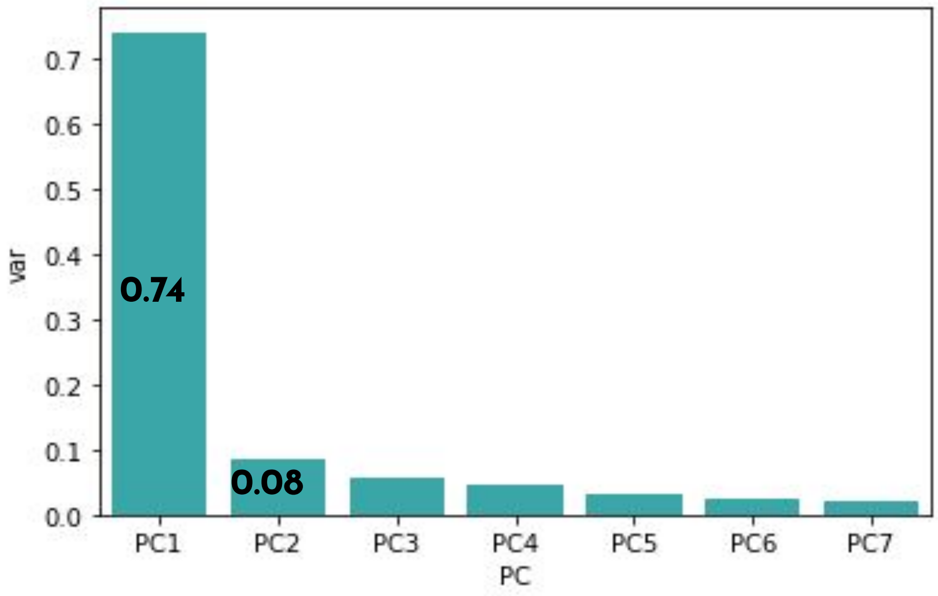
Second Cluster(Availability)



>0.8 Explained Variance

availibility1

Third Cluster(Review Score)



>0.8 Explained Variance

review1, review2

22 → 10

columns

Object Predictors

Convert to Numeric

Host Since and Last Review

- Time difference between reference period and given dates
- Reference period: 2021-01-01

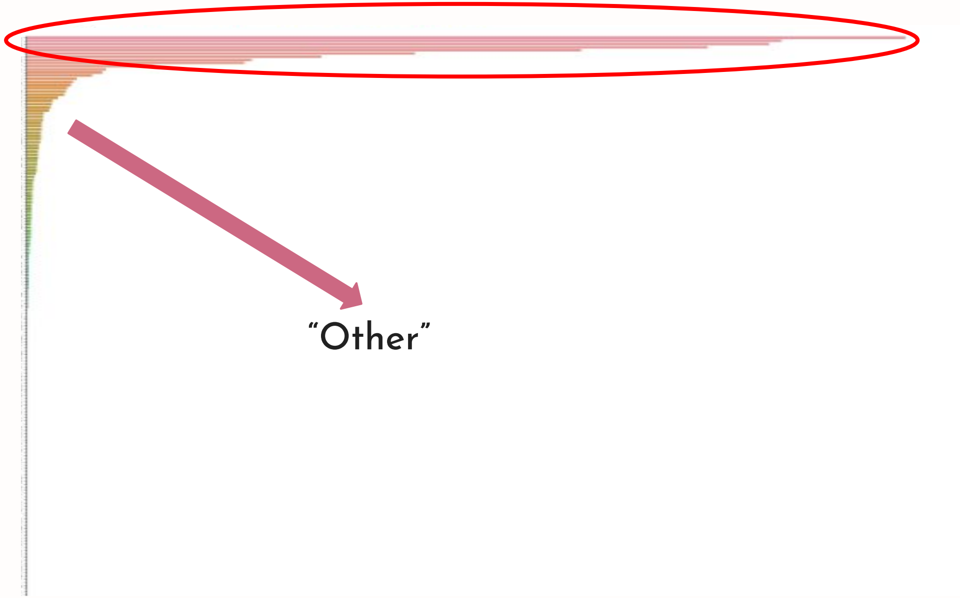
host_since	last_review	reference_period	host_since_period	last_review_period
2008-08-18	2018-10-28	2021-01-01	4519	796
2008-09-16	2018-10-01	2021-01-01	4490	823
2008-10-19	2017-03-20	2021-01-01	4457	1383
2008-11-07	2018-08-16	2021-01-01	4438	869
2009-05-16	2018-11-04	2021-01-01	4248	789

Convert to Numeric

Remaining object Data

- Unordered data
- One Hot Encode
- Categorise lower-count-unique -data as 'Other'

host_neighbourhood



```
0   minimum_nights                                   22507 non-null   float64
1   reviews_per_month                                22507 non-null   float64
2   calculated_host_listings_count                   22507 non-null   float64
3   mean_review_score                                22507 non-null   float64
4   distance_from_central                            22507 non-null   float64
5   host_since_period                                22507 non-null   float64
6   last_review_period                               22507 non-null   float64
7   attribute_1                                      22507 non-null   float64
8   attribute_2                                      22507 non-null   float64
9   availability_1                                   22507 non-null   float64
10  review_1                                         22507 non-null   float64
11  review_2                                         22507 non-null   float64
12  host_neighbourhood_Charlottenburg                22507 non-null   float64
13  host_neighbourhood_Friedrichshain                22507 non-null   float64
14  host_neighbourhood_Kreuzberg                     22507 non-null   float64
15  host_neighbourhood_Mitte                         22507 non-null   float64
16  host_neighbourhood_Moabit                        22507 non-null
17  host_neighbourhood_Neukölln                      22507 non-
18  host_neighbourhood_Other                         2250
19  host_neighbourhood_Pankow
20  host_neighbourhood_Prenzlauer Berg
21  host_neighbourhood_Schöneberg
22  host_neighbourhood_Tempelhof
23  host_neighbourhood_Wedding
24  host_neighbourhood_Wilmersdorf
25  neighbourhood_Alexanderplatz
26  neighbourhood_Brunnenstr. Süd                                     float64
27  neighbourhood_Frankfurter Allee Nord                         null   float64
28  neighbourhood_Frankfurter Allee Süd FK                  non-null   float64
29  neighbourhood_Neuköllner Mitte/Zentrum           22507 non-null   float64
30  neighbourhood_Other                              22507 non-null   float64
31  neighbourhood_Prenzlauer Berg Nordwest           22507 non-null   float64
32  neighbourhood_Prenzlauer Berg Südwest            22507 non-null   float64
33  neighbourhood_Reuterstraße                       22507 non-null   float64
34  neighbourhood_Rixdorf                            22507 non-null   float64
```

```
35  neighbourhood_Schillerpromenade                  22507 non-null   float64
36  neighbourhood_Tempelhofer Vorstadt               22507 non-null   float64
37  neighbourhood_südliche Luisenstadt               22507 non-null   float64
38  neighbourhood_group_Charlottenburg-Wilm.         22507 non-null   float64
39  neighbourhood_group_F          hain-Kreuzberg    22507 non-null   float64
40  neighbourhood_                                   22507 non-null   float64
41  neighbou               ellersdorf                22507 non-null   float64
42  nei                                              22507 non-null   float64
                                                     22507 non-null   float64
                                                     22507 non-null   float64
                                                     22507 non-null   float64
                                                     22507 non-null   float64
                    endorf                           22507 non-null   float64
                  - Schöneberg                       22507 non-null   float64
              eptow - Köpenick                       22507 non-null   float64
              rtment                                 22507 non-null   float64
          pe_Condominium                             22507 non-null   float64
       rty_type_Hostel                               22507 non-null   float64
    property_type_House                              22507 non-null   float64
54  property_type_Loft                               22507 non-null   float64
55  property_type_Other                              22507 non-null   float64
56  property_type_Serviced apartment                 22507 non-null   float64
57  room_type_Entire home/apt                        22507 non-null   float64
58  room_type_Private room                           22507 non-null   float64
59  room_type_Shared room                            22507 non-null   float64
60  cancellation_policy_flexible                     22507 non-null   float64
61  cancellation_policy_moderate                     22507 non-null   float64
62  cancellation_policy_strict_14_with_grace_period  22507 non-null   float64
63  cancellation_policy_super_strict_30              22507 non-null   float64
64  cancellation_policy_super_strict_60              22507 non-null   float64
```

NORMALISE

Random Forest

Modelling

With sklearn

Choosing the model

Model used: Random Forest Regressor

GridSearchCV

Finding the best combination of numbers for each parameter

```
param_grid = {
    'bootstrap': [True],
    'max_depth': [30,40,50],
    'max_features': [20,30,40],
    'min_samples_leaf': [1,2,3],
    'min_samples_split': [2,3,4],
    'n_estimators': [100,200,300,400]}
# Create a based model
rf = RandomForestRegressor()
# Instantiate the grid search model
grid_search = GridSearchCV(estimator = rf, param_grid = param_grid,
                          cv = 3, n_jobs = -1, verbose = 2)
```

GridSearch Result

```
{'bootstrap': True,
 'max_depth': 30,
 'max_features': 20,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'n_estimators': 400}
```

Random Forest Result

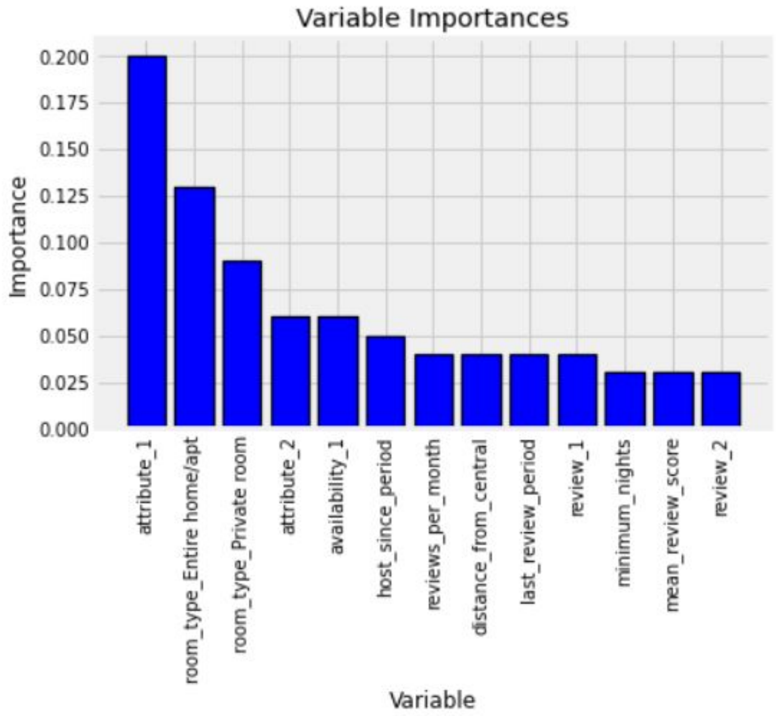
```
# Get train score
best_grid = grid_search.best_estimator_
best_grid.fit(X_train,y_train.values.ravel())
best_grid.score(X_train,y_train.values.ravel())
```

```
RandomForestRegressor(max_depth=30, max_features=20, n_estimators=400)

0.9370627222376413
```

```
# Get test score
best_grid.fit(X_test,y_test.values.ravel())
best_grid.score(X_test,y_test.values.ravel())
```

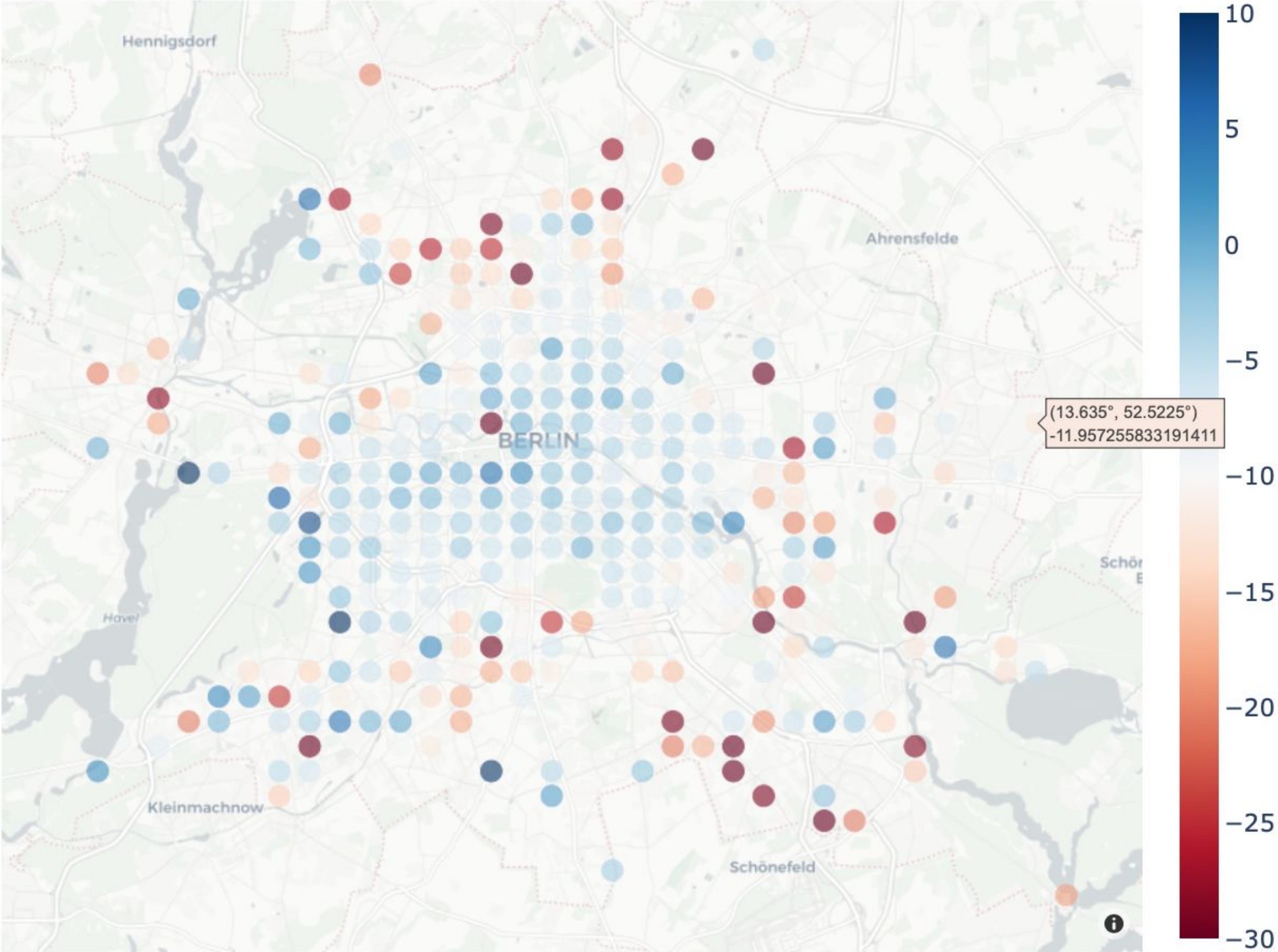
```
RandomForestRegressor(max_depth=30, max_features=20, n_estimators=400)

0.9341594748568888
```

Combining Results



Predictions vs Actual

price	predicted	difference	percentage
60.0	69.722500	-9.722500	-16.204167
17.0	32.148333	-15.148333	-89.107843
90.0	85.312500	4.687500	5.208333
26.0	33.627170	-7.627170	-29.335268
42.0	42.037154	-0.037154	-0.088461


Hennigsdorf
Ahrensfelde
BERLIN
Havel
Kleinmachnow
Schönefeld
(13.635°, 52.5225°)
-11.957255833191411
10
5
0
−5
−10
−15
−20
−25
−30

Danger of Overfitting

Overfitting

Overfitting occurs when we have fit our model's parameters too closely to the training data.

Hence, when we overfit, we tend to assume that everything we see in the training data exactly how it'll appear in the real world.

Ways to prevent Overfitting

What we've done to prevent overfitting

1. Inspect the training data
2. Collect Data thoughtfully
3. Augment training data
4. Restricting the feature set
5. Reflection

Inspect Training Data

We inspect training data to look for trends. Trends that we believe are strong within the dataset. If our machine model misses some of these trends, we know we might have gone wrong somewhere.

Some general trends that we visually identified

The larger the listing, the higher the listing price
(EDA + Data Cleaning)

The closer you are to central Berlin, the higher the listing price
(Geo Analysis)

The higher the review score, the higher the listing price
(Sentiment Analysis)

The more occupants a house can host, the higher the listing price
(EDA)

RF on Training Data

Our machine model recognised all of these general, more obvious trends. By the feature importance plot, it showed that **Attribute 1** (which consists of Square Feet, No of Bedrooms, No of Bathrooms) was the most important factor. Also, **distance from centre** and **mean review score** were among the top 10 most important features.

Some general trends that we visually identified

The larger the listing, the higher the listing price (EDA + Data Cleaning)

The closer you are to central Berlin, the higher the listing price (Geo Analysis)

The higher the review score, the higher the listing price (Sentiment Analysis)

The more occupants a house can host, the higher the listing price (EDA)

Collect Data Thoughtfully

We need to collect data from all groups to ensure that we have an accurate perspective of the population. Some neighbourhoods/areas might be undersampled, or have significantly less data points, so the model will fit to the oversampled population

What we have done:

All of the original data represented all the listings in Berlin. We took the whole population.

However, to reduce the amount of undersampled neighbourhoods, we have reduced our population, to listings within 10km of the city centre.

22132

rows

21170

rows

962 rows

Difference of 4%

Reflection

Machine learning algorithms always introduce a bias as a function of programs that are trying to make assumptions and rules by looking at data.

What we have done:

We have acknowledged that our model has a bias towards the centre of Berlin, with majority of the listings within 6km of the city centre. We have not compensated for this by reducing the radius to 6km because our business model is to optimise the prices for existing listings, which means we would want to cater to the entirety of Berlin.

Regardless, removing the outskirts was important, because as we see, it affects our model slightly.

GridSearch CV Parameters

Original Data

```
{'bootstrap': True,
 'max_depth': 30,
 'max_features': 20,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'n_estimators': 400}
```

Outskirts Removed

```
{'bootstrap': True,
 'max_depth': 40,
 'max_features': 20,
 'min_samples_leaf': 2,
 'min_samples_split': 3,
 'n_estimators': 400}
```

Train & Test Score

Original Data

```
# Get train score
best_grid = grid_search.best_estimator_
best_grid.fit(X_train,y_train.values.ravel())
best_grid.score(X_train,y_train.values.ravel())
```

```
RandomForestRegressor(max_depth=30, max_features=20, n_estimators=400)

0.9370627222376413
```

```
# Get test score
best_grid.fit(X_test,y_test.values.ravel())
best_grid.score(X_test,y_test.values.ravel())
```

```
RandomForestRegressor(max_depth=30, max_features=20, n_estimators=400)

0.9341594748568888
```

Outskirts Removed

```
# Get train score
best_grid = grid_search.best_estimator_
best_grid.fit(X_train,y_train.values.ravel())
best_grid.score(X_train,y_train.values.ravel())
```

```
RandomForestRegressor(max_depth=40, max_features=20, min_samples_leaf=2,
                      min_samples_split=3, n_estimators=400)

0.8860685420290422
```

```
# Get test score
best_grid.fit(X_test,y_test.values.ravel())
best_grid.score(X_test,y_test.values.ravel())
```

```
RandomForestRegressor(max_depth=40, max_features=20, min_samples_leaf=2,
                      min_samples_split=3, n_estimators=400)

0.878624029526128
```

Train/Test Score Analysis

Original Data

```
# Get train score
best_grid = grid_search.best_estimator_
best_grid.fit(X_train,y_train.values.ravel())
best_grid.score(X_train,y_train.values.ravel())
```

```
RandomForestRegressor(max_depth=30, max_features=20, n_estimators=400)

0.9370627222376413
```

```
# Get test score
best_grid.fit(X_test,y_test.values.ravel())
best_grid.score(X_test,y_test.values.ravel())
```

```
RandomForestRegressor(max_depth=30, max_features=20, n_estimators=400)

0.9341594748568888
```

Outskirts Removed

```
# Get train score
best_grid = grid_search.best_estimator_
best_grid.fit(X_train,y_train.values.ravel())
best_grid.score(X_train,y_train.values.ravel())
```

```
RandomForestRegressor(max_depth=40, max_features=20, min_samples_leaf=2,
                      min_samples_split=3, n_estimators=400)

0.8860685420290422
```

```
# Get test score
best_grid.fit(X_test,y_test.values.ravel())
best_grid.score(X_test,y_test.values.ravel())
```

```
RandomForestRegressor(max_depth=40, max_features=20, min_samples_leaf=2,
                      min_samples_split=3, n_estimators=400)

0.878624029526128
```

For both **Original Data** and **Outskirts Removed**, we can see that our train and test scores are relatively high. We can also see that for both, our train and test scores are very close to each other, which indicates that we have avoided overfitting.

While this is a good indication that we have not overfit the model, it is not sufficient to guarantee it. Hence, given more time, we would run it though cross-validation libraries and use other models to confirm our model is appropriate and our results are reasonable and justifiable.

Variable Importance

Original Data

```
attribute_1            Importance: 0.2
room_type_Entire home/apt Importance: 0.13
room_type_Private room Importance: 0.09
attribute_2            Importance: 0.06
availability_1         Importance: 0.06
host_since_period      Importance: 0.05
reviews_per_month      Importance: 0.04
distance_from_central Importance: 0.04
last_review_period     Importance: 0.04
review_1               Importance: 0.04
```

Outskirts Removed

```
attribute_1            Importance: 0.22
room_type_Entire home/apt Importance: 0.15
room_type_Private room Importance: 0.08
availability_1         Importance: 0.06
host_since_period      Importance: 0.05
attribute_2            Importance: 0.05
reviews_per_month      Importance: 0.04
last_review_period     Importance: 0.04
review_1               Importance: 0.04
minimum_nights         Importance: 0.03
```

Variable Importance Analysis

Original Data

```
attribute_1            Importance: 0.2
room_type_Entire home/apt Importance: 0.13
room_type_Private room Importance: 0.09
attribute_2            Importance: 0.06
availability_1         Importance: 0.06
host_since_period      Importance: 0.05
reviews_per_month      Importance: 0.04
distance_from_central Importance: 0.04
last_review_period     Importance: 0.04
review_1               Importance: 0.04
```

Outskirts Removed

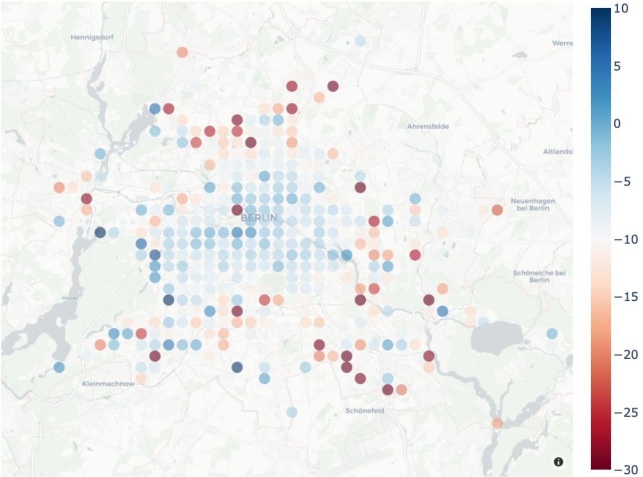
```
attribute_1            Importance: 0.22
room_type_Entire home/apt Importance: 0.15
room_type_Private room Importance: 0.08
availability_1         Importance: 0.06
host_since_period      Importance: 0.05
attribute_2            Importance: 0.05
reviews_per_month      Importance: 0.04
last_review_period     Importance: 0.04
review_1               Importance: 0.04
minimum_nights         Importance: 0.03
```

As we see, even though the order shifts around, it does not affect the values significantly.

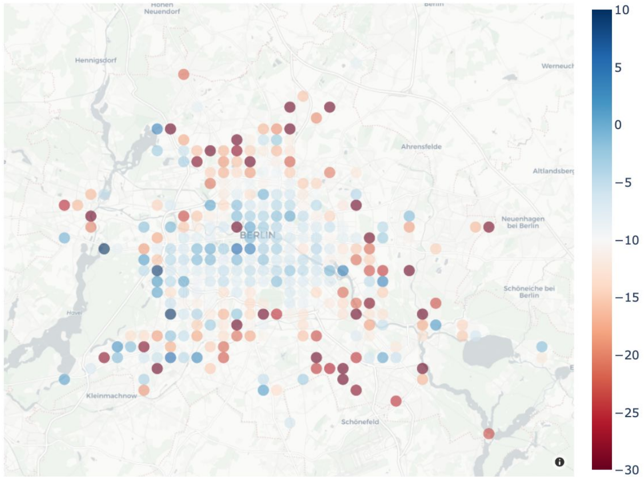
The biggest difference is the lack of distance from central variable. In the Outskirts Removed dataframe, it is in 12th place, with a numerical importance of 0.03. Not a significant change, but one to take note of.

Visualisations

Original Data

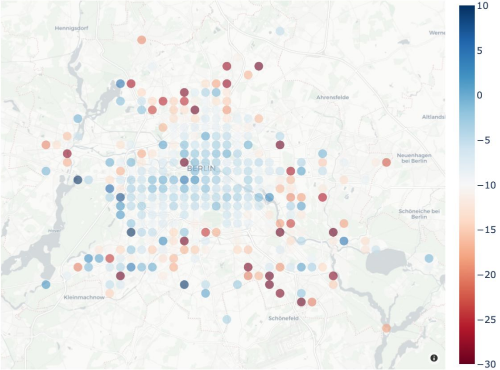


Outskirts Removed

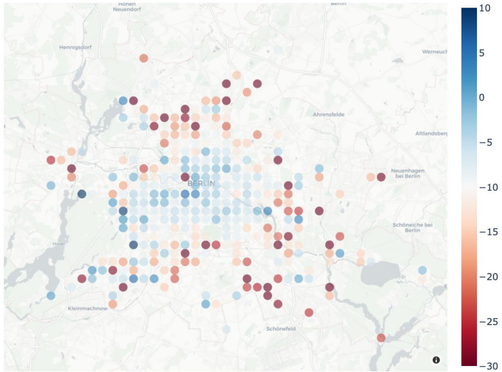


Visualisation Analysis

Original Data



Outskirts Removed



In the original map, we had 342 grid points. In the outskirts removed map, we have 330 grid points. And while they look the same visually, we notice a slight change in the hue of the grid points.

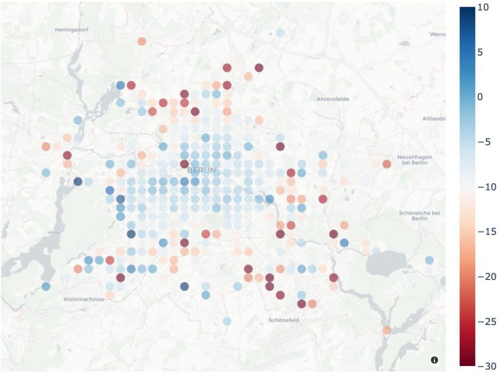
We notice that the listings on the edge of outskirts removed are slightly more undervalued (compared to the original map) and hence, accounts for a smaller centre as well.

Hence, by removing outskirts, we see that it shows us a greater distinction between central Berlin and the outer neighbourhoods.

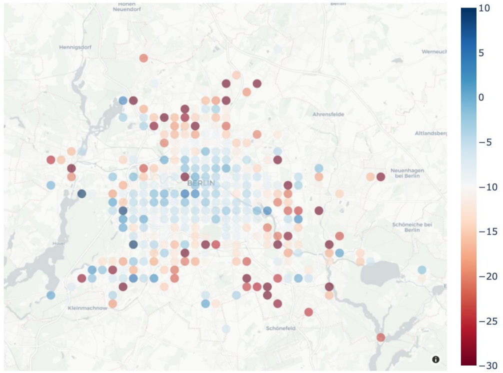
This is not necessarily a good thing because our predicted prices are not as accurate in the Outskirts removed model, compared to the Original Data model. While the visualisation is helpful, it represents a weaker representation of what listing prices should be.

What does this mean for our business case?

Original Data



Outskirts Removed



For our business case, we find that the Outskirts Removed model gave us a weaker train/test score, but it also showed us a greater distinction between listings in central Berlin and the greater neighbourhoods.

The weaker test score gave rise to higher percentages across our dataframe and while it shows the distinction, it is not as helpful as the original data, which better reflects the predicted price.

Given more time and data from other German cities, and other megacities in Europe, we believe that we can build a stronger and more cohesive model that better predicts predicted price, which is our ultimate goal at hand.

With an accurate and justifiable predicted price, we are confident that our feature will help existing hosts adjust their price to better reflect the value of their property, and thus increase their profit in the long run.

Business Case

Price optimisation of listings for Hosts

Thank You