Stabilizing Geo-Spatial Surfaces in Data-Sparse Regions - An Application to Residential Property Prices

Norbert Pfeifer and **Miriam Steurer**

University of Graz, Austria

What this paper is about

We introduce a new method that combines two popular modelling techniques

1. Penalized regression spline
2. Decision tree approach (e.g. Random Forest or XGBoost)

Why:

- Penalized regression splines are one of the most used methods for **smoothing noisy data.**
- Decision tree based methods are great at simultaneously **clustering and averaging data.**

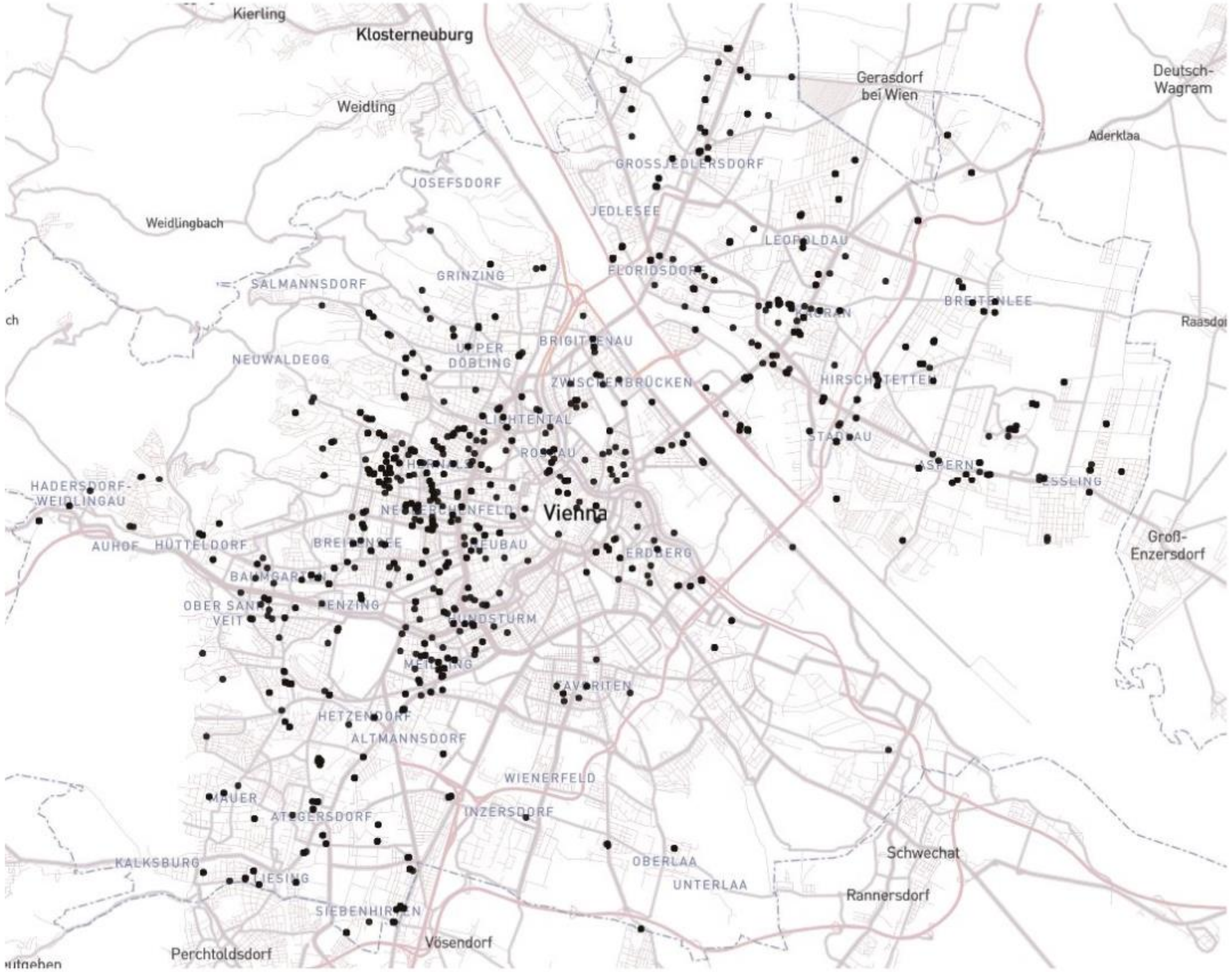
Our combination method improves on a regression spline model

- avoids overshooting
- does not limit accuracy where data is plentiful

We demonstrate our approach by estimating a geo-spatial price surface for newbuild apartments in Viennain 2020.

Vienna:

new built appartments sold in 2020



Introduction

A three-dimensional house price surfaces can be useful to

- identify regional subcenters (McMillen, 2001),
- provide inputs for quantitative spatial models (Allen and Arkolakis, 2014; Ahlfeldt et al., 2015),
- estimate spillover efects of local amenities such as public schools (Gibbons and Machin, 2003), and
- replace regional fixed effects in hedonic house price models (Hill and Scholz, 2018).

Price surfaces can be constructed in different ways: KNN, locally weighted regression, kernel regression, penalized regression splines, etc.

Penalized regression splines have benefits over other methods when estimating geospatial price surfaces (Craig and Ng, 2001; Bao andWan, 2004; Hill and Scholz, 2018; Melser and Hill, 2019).

Introduction cont.

- Starting in the 1950s spline functions popular for modelling complex surfaces (especially in industrial production)
- Broader academic interest since the 1970s (de Boor 1978, Wahba & Wold 1975, Hastie & Tibshirani 1986)
- Increasingly employed in housing economics (Bao and Wan 2004, Hill and Scholz 2018, Melser and Hill 2019, Diewert and Shimizu 2019)

Introduction cont.

However, there is a problem with splines:

- They are bad at extrapolating into areas without data support → they 'overshoot' (i.e., produce highly implausible values)
- Overshooting is worst at boundaries

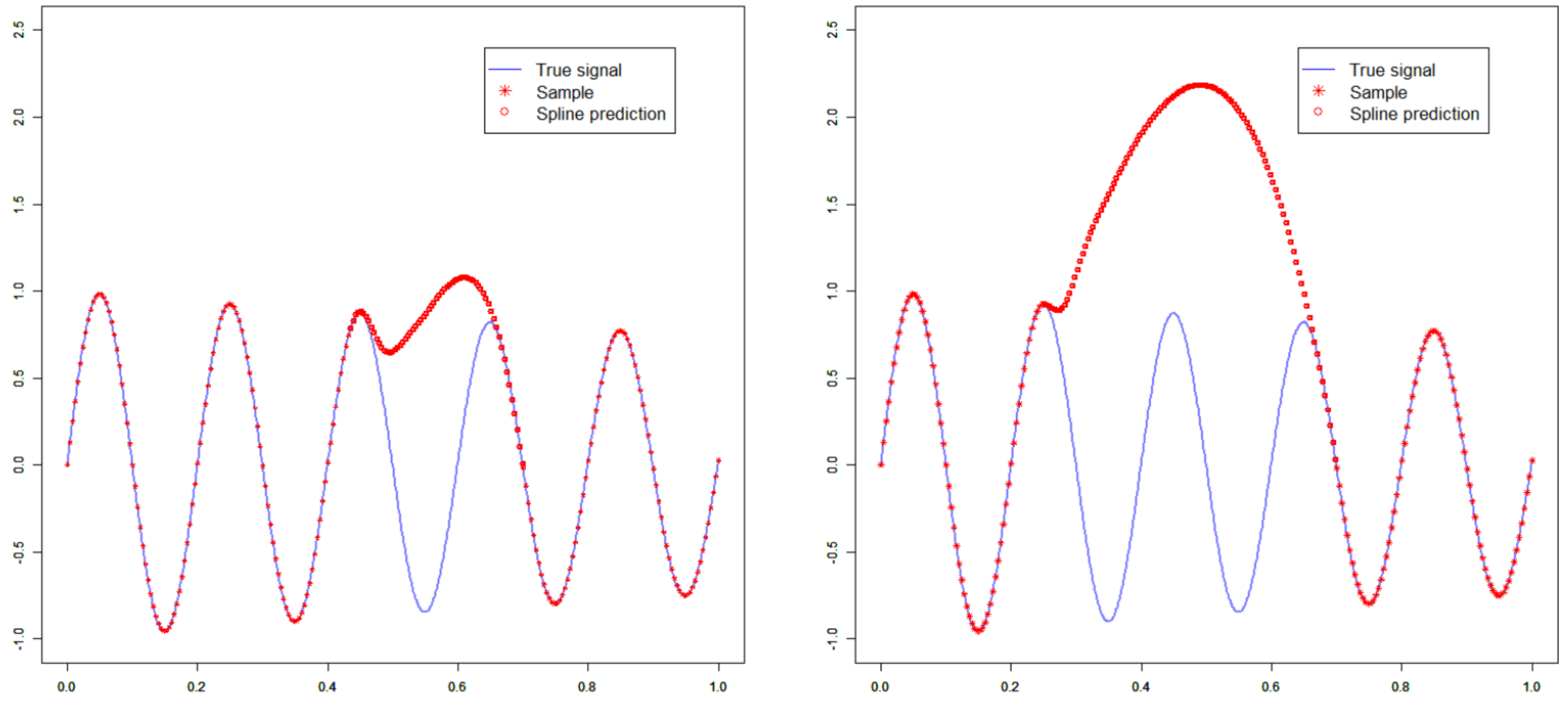


Figure: Spline behaviour by different length of data gaps

Problem and solution

- Housing data is not directly suited for spline estimation:
  - they are not evenly distributed
  - there are many data gaps

→ we introduce **helper points** in data-gap areas prior to estimating the penalized regression spline surface.

- We estimate helper point values with decision-tree-based algorithm (e.g. XGBoost).

- Combination of penalized regression splines with the decision tree-based helper points optimally combines the strengths of each method.

Penalized regression splines

- Piece-wise polynomial functions, defined locally between ”knots”.
- between knots they are formed by adding polynomial basis functions.
- At knot values they are combined so that overall spline function is smooth and twice-differentiable.
- A penalization parameter alpha puts a cost on overall wiggliness of the overall spline function.

The objective of a penalized regression spline is to find the function that solves the following minimization problem:

$$\min_{f(.)}\{\sum_{i=1}^{n}(y_i - f(x_i))^2 + \lambda \int (f''(x))^2 dx\}.$$

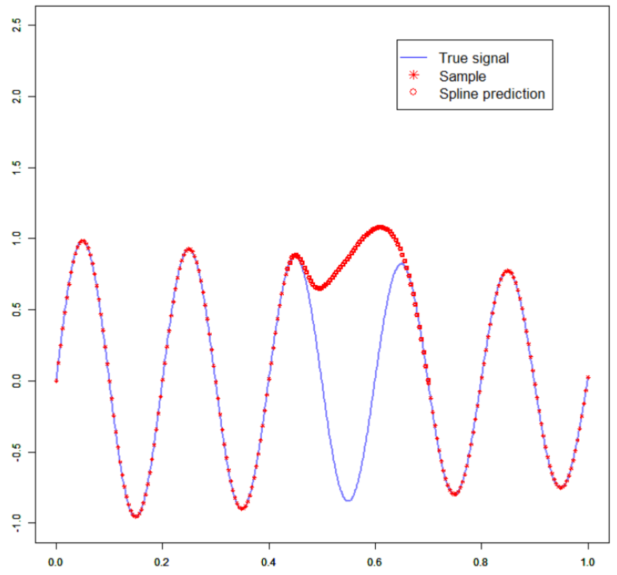
- When optimizing the spline function, coefficients for each of the basis functions are determined so that together they approximate the underlying data generating function. The value of the spline function at point x is then simply the sum of these estimated basis functions at x:

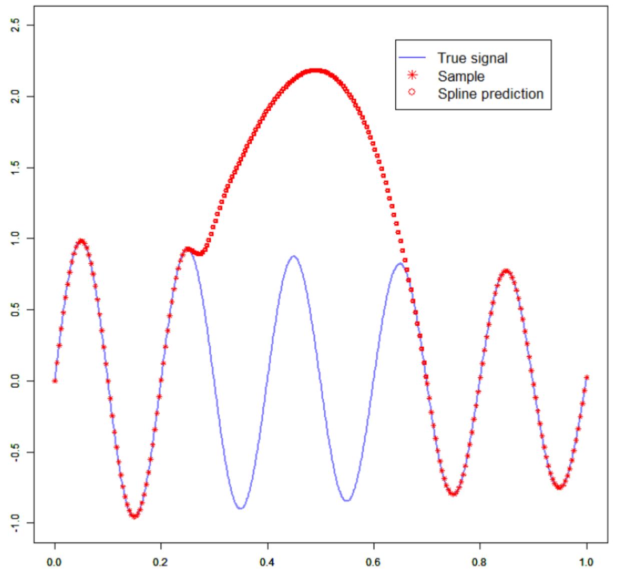
$$f(x) = \sum_{k=1}^{K} \beta_k b_k(x),$$

How splines operate in regions with missing data?

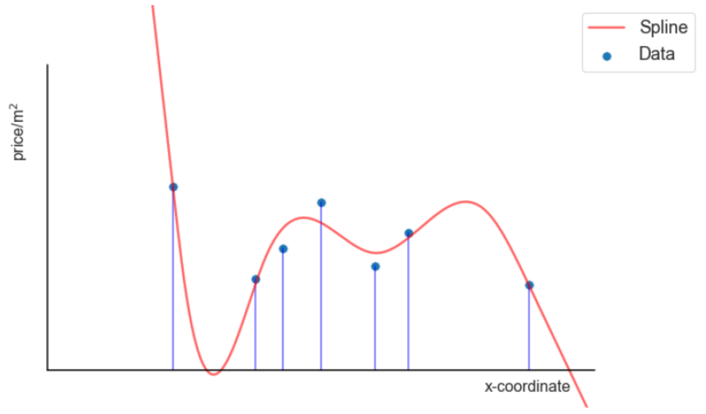
No data points → deviations are without cost.

The only binding constraint in such areas is the global constraint to keep the overall ”wiggliness” as low as possible, which is achieved by minimizing the spline’s second derivative. This is accomplished by establishing a ”peak” or ”trough” in the estimated values; in other words, the algorithm overshoots.

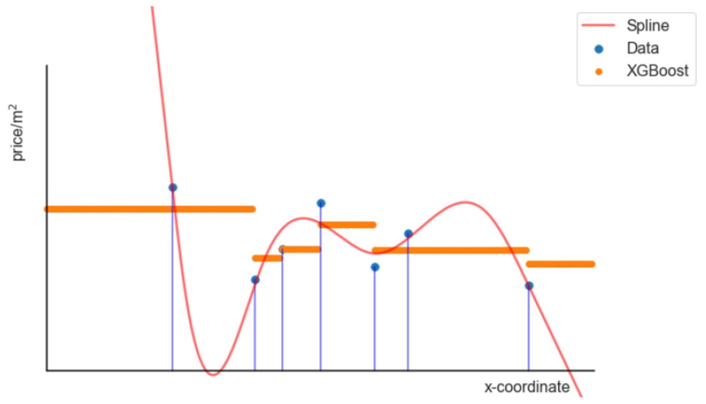




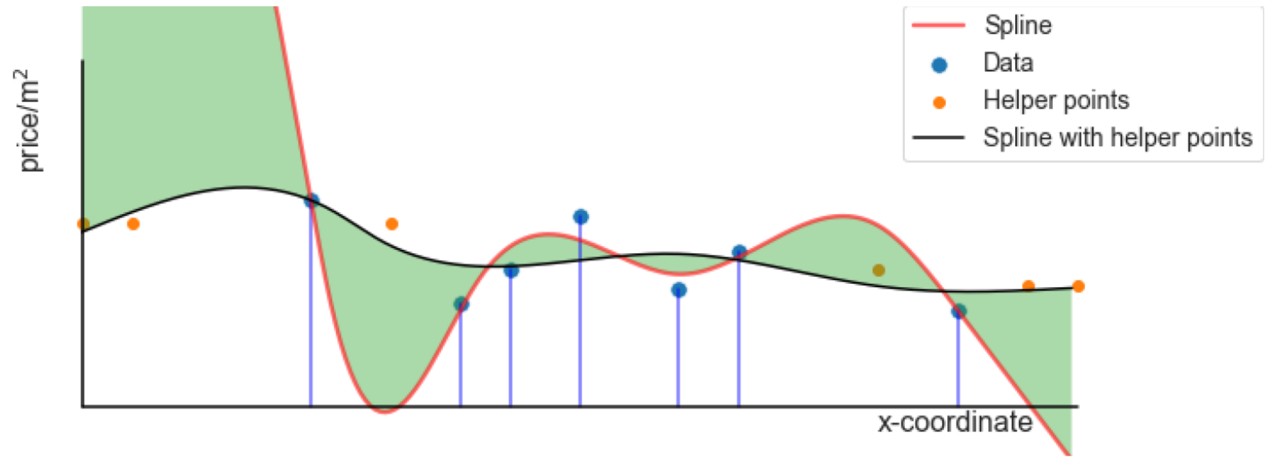
Our solution concept



(a) Spline fitting on original data

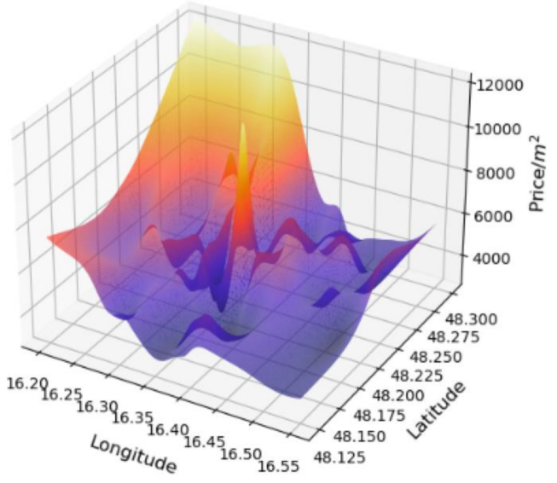


(b) Fit a tree-based model to estimate helper values

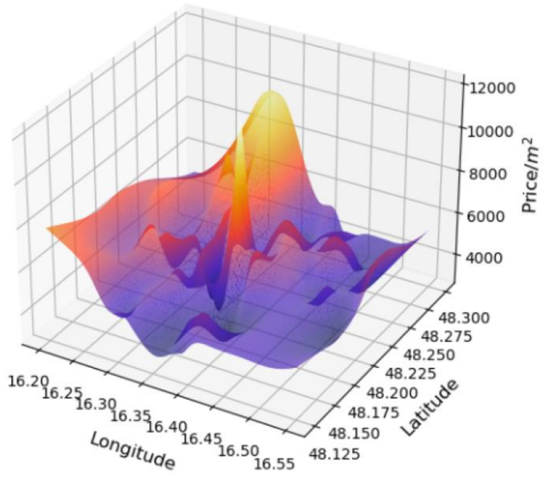


(c) Spline with and without helper points)

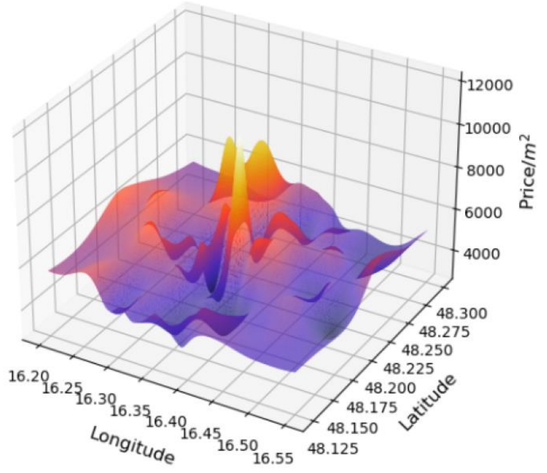
Figure 2: Illustration of Guided Spline Function Approach



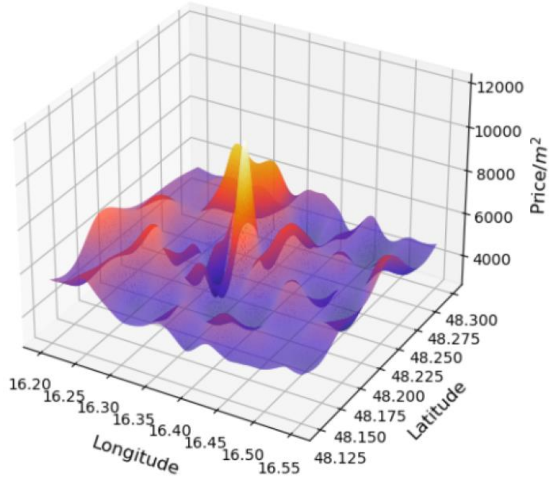
(a) Spline surface without helper points



(b) Spline surface with 2 helper points



(c) Spline surface with 19 helper points



(d) Spline surface with 177 helper points

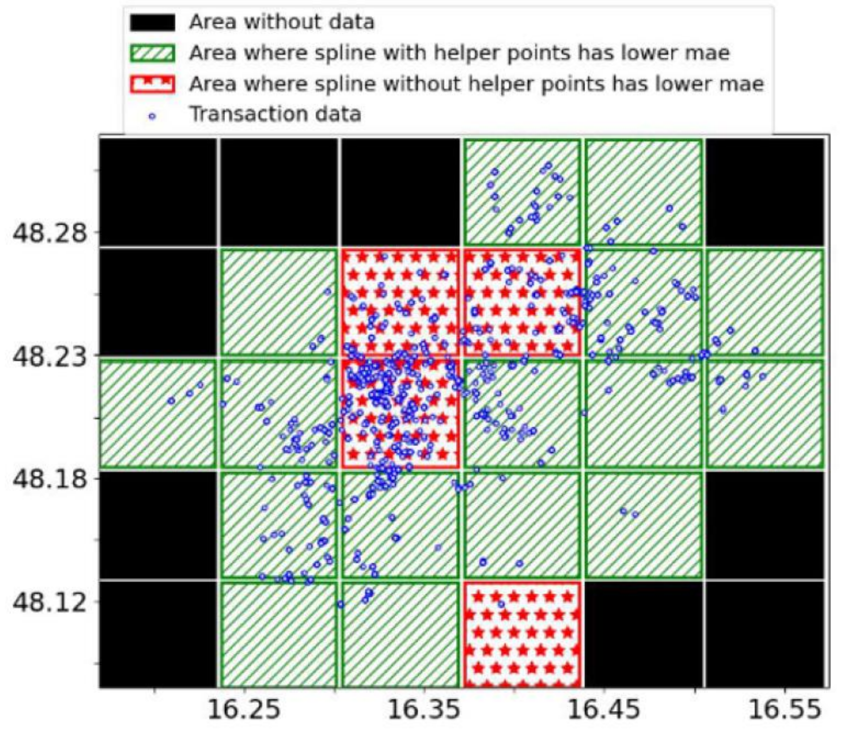
Guided spline estimation:
Estimated square-meter price for new-built apartments in Vienna in 2020

→ Boundary values become a lot better when helper points included.
→ We do not need many helper points!

Estimating spline performance in data gap areas

To produce (additional) artificial data gap areas, we define equally sized areas by placing a grid over the map of Vienna. Then, we take one cell as our cut-out sample and fit spline surfaces without and with the inclusion of helper points on the remaining data. Repeat with next cell….

Placement of helper points	Cut-out-sample MAE	
min. dist. (# HP)	without HPs	with HPs
5 km (2 HPs)	1419.21	1406.23
2.5 km (19 HPs)	1419.21	1228.91
1 km (177 HPs)	1419.21	1070.67
0.5 km (911 HPs)	1419.21	1021.15
1168 HPs (no transactions data)	1419.21	970.72


Area without data
Area where spline with helper points has lower mae
Area where spline without helper points has lower mae
Transaction data
48.28
48.23
48.18
48.12
16.25
16.35
16.45
16.55

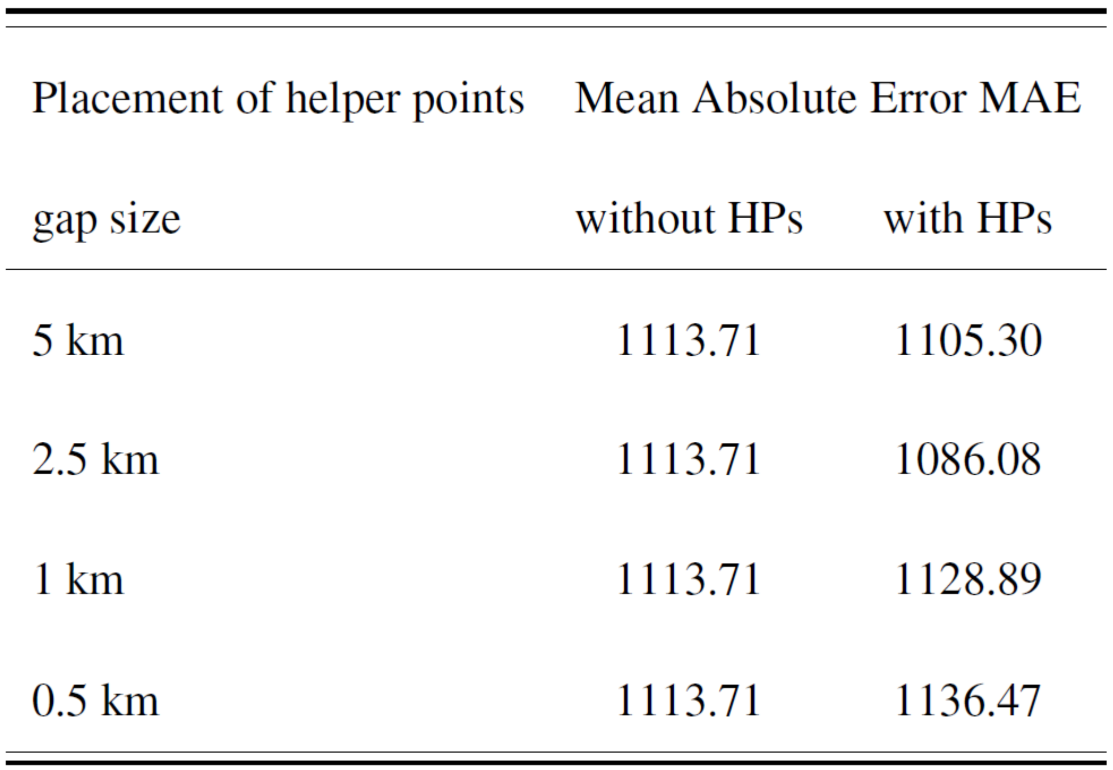
Testing out-of-sample accuracy

How well can spline surfaces predict the price of unseen properties from the dataset?

→ split data by randomly placing half of the property locations in each subset.

→ estimate spline surfaces with and without helper points

→Finding: out-of-sample accuracy best when a low number of helper points is included.



Placement of helper points	Mean Absolute Error MAE	
gap size	without HPs	with HPs
5 km	1113.71	1105.30
2.5 km	1113.71	1086.08
1 km	1113.71	1128.89
0.5 km	1113.71	1136.47

Conclusion

- we introduced the concept of guided spline construction with helper points.
- our method incorporates a decision-tree-based algorithm along with the penalized regression spline estimation.
- both of these approaches can represent local price levels, but neither is ideal on its own.

- Findings:
- guided spline method can prevent overshooting of spline values in regions with low data density (especially at boundary regions) and simultaneously maintain the spline function's flexibility in data-rich regions.
- Guided spline method can also improve overall out-of-sample accuracy as long as helper points are not placed too closely to existing values.

Thank you for your attention!

For comments please contact me at:

miriamsteurer@hotmail.com

miriam.steurer@uni-graz.at

C Appendix: Pseudo algorithm for the implementation of the spline estimation with helper points

Algorithm 1 Geo-spatial cross validation with test grid size 5km-by-5km as illustrated in **??**

```
1:  $data \leftarrow$ all transactions
2:  $min_{dist} \leftarrow 2.5$ // min distance to nearest observation for helper points
3:  $latitude_{step} \leftarrow 0.044966$ // haversine distance of 5 km along latitude
4:  $longitude_{step} \leftarrow 0.067416$ // haversine distance of 5 km along longitude
5:  $latitude_{list} \leftarrow sequence(min(data[''Latitude''], max(data[''Latitude''], latitude_{step})$
6:  $longitude_{list} \leftarrow sequence(min(data[''Longitude''], max(data[''Longitude''], longitude_{step})$
7:  $helperpoints \leftarrow list()$
8:  for $i = 1$ to $Length(latitude_{list}) - 1$ do
9:      for $j = 1$ to $Length(longitude_{list}) - 1$ do
10:         $trainData \leftarrow data[''Latitude''] > latitude_{list}[i]$ AND
                $data[''Latitude''] < latitude_{list}[i+1]$ AND
                $data[''Longitude''] > longitude_{list}[j]$ AND
                $data[''Longitude''] < longitude_{list}[j+1]$
11:         $testData \leftarrow$ data not in $trainData$
12:         $rounds \leftarrow 0$
13:         while $rounds < 100000$ do
14:             $rounds = 0$
15:             $rand_{latitude} \leftarrow random(latitude_{min}, latitude_{max})$
16:             $rand_{longitude} \leftarrow random(longitude_{min}, longitude_{max})$
17:             $rand_p \leftarrow (rand_{longitude}, rand_{latitude})$
18:             $dist_{vector} \leftarrow$ calculate distance from $rand_p$ to all latitude, longitude pairs in $trainData$
19:             if $min(dist_{vector}) <= min_{dist}$ then
20:                 append random point to $helperpoints$
21:             else
22:                 $rounds+ = 1$ //increase rounds by 1
23:             end if
24:         end while
25:     end for
26: end for
27: Train $XGBoost$ model on $trainData$
28: $Helper \leftarrow$ Predict price from $XGBoost$ model for all $helperpoints$
29: $trainDataHelper \leftarrow trainData + Helper$
30: splineHelperPoints $\leftarrow$ train spline on latitude, longitude from $trainDataHelper$ to predict square meter price on $testData$
```

XGBoost output

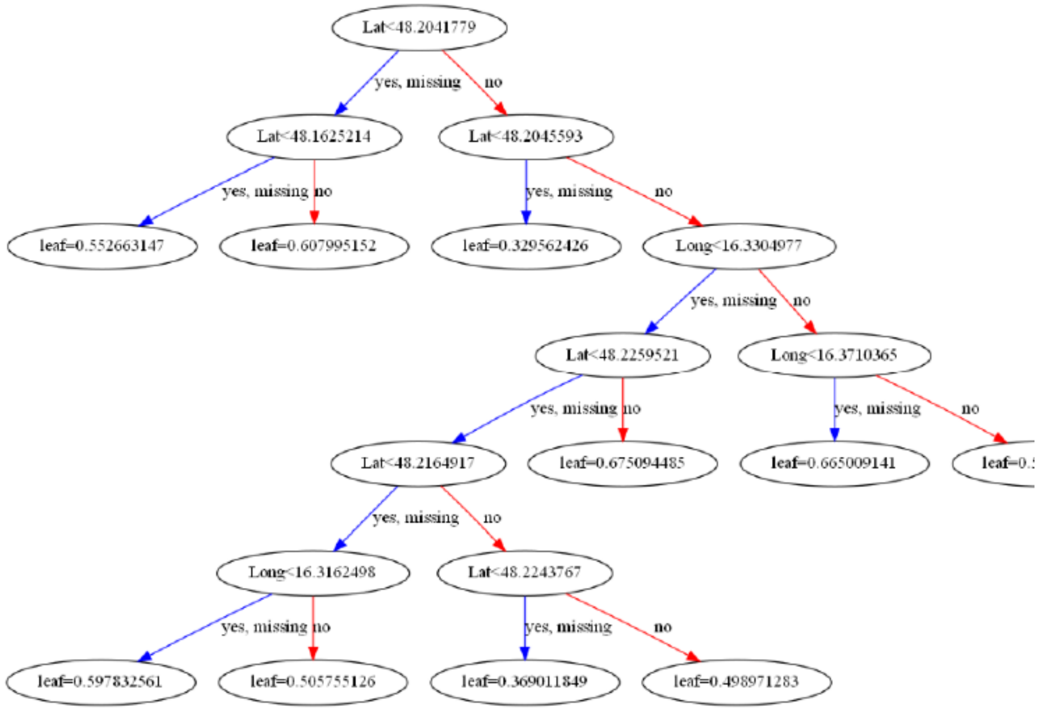


Figure B1: Sample Tree of used XGBoost model

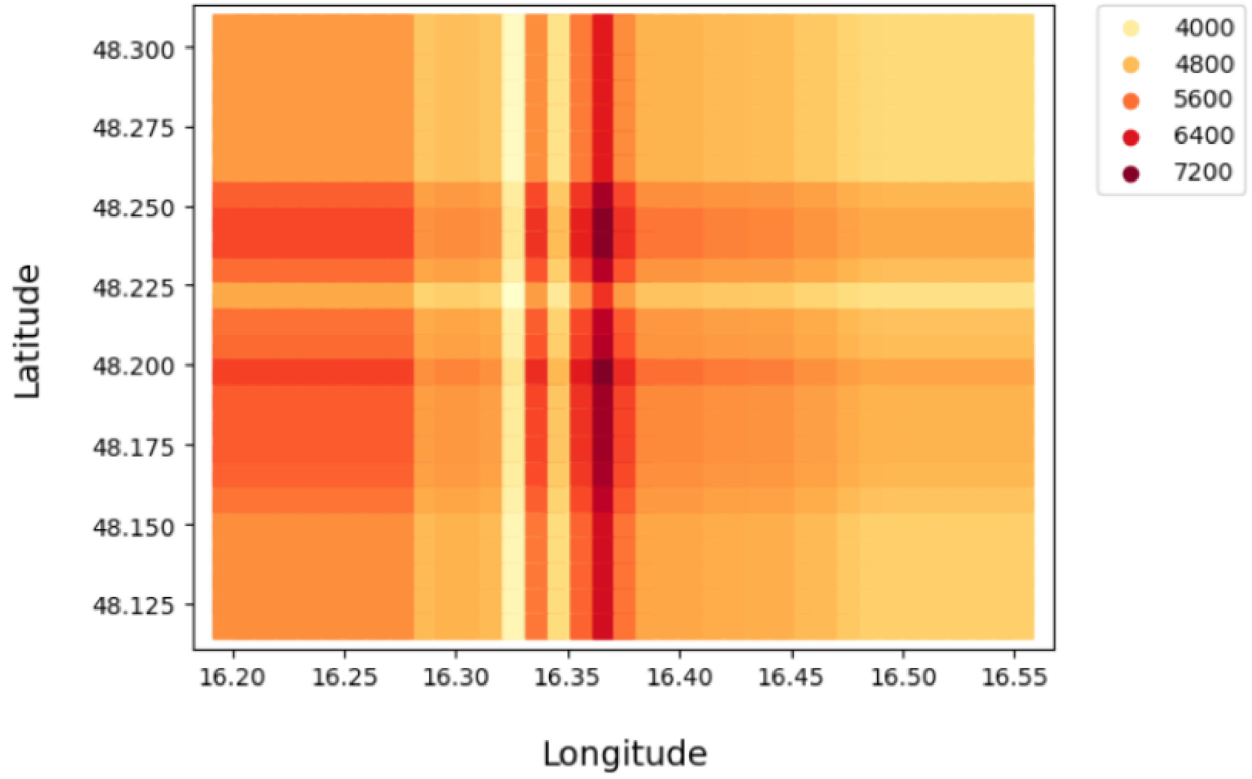


Figure B2: XGBoost price level estimate