Approximation methods for TSP path length on road networks

Koen Stevens

May 23, 2025

Bachelor's Thesis Econometrics

Supervisor: dr. N.D. van Foreest

Second assessor: dr. W. Zhu

Approximation methods for TSP path length on road networks

Koen Stevens

Abstract

1 Introduction

The Traveling Salesman Problem (TSP) is an important problem in operations research. It is particularly relevant for last-mile carriers and other logistics companies where efficient routing directly impacts cost, time and service quality. Since the number of parcels worldwide has increased between 2013 and 2022 and is expected to keep increasing (Statista, 2025), the need for fast, scalable route planning methods becomes ever more pressing.

The TSP is an NP-hard problem, it is computationally intensive to find the exact solution for large instances. In many real-world scenarios, the exact optimal routes may not be needed, but instead a rough, reliable estimate of the optimal route length. For instance, consider a postal delivery company. This firm may need to assign a certain amount of deliveries or a certain area to each postman. Reliable estimates for the route length can provide valuable information for making such decisions.

Efficient approximation methods provide a solution for such practical applications where exact solutions are too computationally intensive to conduct or not feasible due to insufficient data. These methods aim at approximating the expected optimal total travel time or distance, while using minimal data and computational effort.

There is extensive research on such approximation methods and how they perform in the Euclidean plane. Consider n uniformly drawn locations from some area in $\mathbb{R}^2$ with area A. Beardwood, Halton, and Hammersley (1959) find the relation:

$$\frac{L}{\sqrt{n}} \to \beta\sqrt{A}, \quad \text{as } n \to \infty \tag{1}$$

as an estimation for the length of the shortest TSP path measured by Euclidean distance through these random locations, where β is some proportionality constant.

The BHH formula is a neat result, the formula makes intuitive sense. However, in contrast to the Euclidean plane, based on different features of the area, TSP path lengths in this area might behave differently. For example, when there are clusters of locations with empty space in between them, or natural boundaries such as canals could have impact on TSP path length in an area. The BHH formula might not be able to capture such complex relations, and fail to provide accurate results. The value for β might differ vastly across different types of areas, as a result of varying geographic features. This would result in having to use a different value for β for every different area, making it less viable to use this formula for delivery services, especially they do not know the true value of β for the areas they serve. A potential solution to this might be to use supervised learning methods to predict the length of the shortest TSP path, aimed at capturing more complex relations between area features and the TSP path length in this area. This would result in one central formula to predict TSP path length accurately, removing the need to calculate separate parameters for each area.

This research investigates how well Equation 1 performs, predicting TSP path length on real road networks. Using OpenStreetMap (OpenStreetMap contributors, 2025) data, TSP instances are simulated in a wide variety of different urban areas in the Netherlands, then solve these for the actual shortest paths using the Lin–Kernighan heuristic (Lin and Kernighan, 1973). Then, the β from equation 1 is estimated and the performance of this formula is analyzed. Additionally, the results for β and the performance across the selected areas is compared. In an attempt to capture more complex relations between area features and TSP path length, a supervised learning method is used as an alternative model to predict TSP paths.

The core contributions of this research are:

1. An analysis of the BHH formula under the following conditions:
   (a) The locations are drawn from the set of postcodes in the area in question, a relaxation of the uniform distribution of visited locations. In reality, delivery locations for companies are not uniformly distributed across the delivery area, but they are clustered in certain parts of the area, for example in high rise buildings;
   (b) Applied to realistically sized (in the sense that a delivery person could serve this area in a single workday) real-world parts of cities and villages in the Netherlands;
2. A supervised learning analysis to investigate how well the optimal TSP path length can be predicted, based on features of the area the path is in, including road network density, address density and other natural and man-made features of the area.

The analysis can easily be extended to any type of area in any part of the world, one would only have to download the OpenStreetMap (OpenStreetMap contributors, 2025) for another part of the world and add the names of the areas to apply it to. The source code of this project is available on GitHub.

In section 2 the context and previous research in this field is provided. In section 3 a formal problem definition is stated. Section 4 contains a description of the data that is used, and explains how it is processed. In section 5 the design of the experiment is documented, section 6 displays the results. In section 7 the research is concluded.

2 Literature Review

In this section the existing literature on the BHH formula and some applications, and on the Lin-Kernighan heuristic and its implementations is reviewed.

2.1 Applications of the BHH formula

This research concerns the performance of formula 1 for reasonable amounts of locations j a delivery person can visit in a workday, say $20 \leq j \leq 90$. Lei, Laporte, Liu, and Zhang (2015) estimates the values of β for a selection of values for j. In their research, the points were generated uniformly and the L_2 distance metric was used. Table 1 lists the results.

Table 1: Empirical estimates of β as a function of j, $20 \leq n \leq 90$ (Lei et al., 2015)

j	$\beta(j)$
20	0.8584265
30	0.8269698
40	0.8129900
50	0.7994125
60	0.7908632
70	0.7817751
80	0.7775367
90	0.7773827

Figliozzi (2008) is the first research to apply approximation formulas to real-world instances of TSPs (and VRPs (Vehicle Routing Problems)), in the sense that distances between locations are measured over the actual road network. An extension of formula 1 that works for VRPs is assessed in this setting. It is found that this model has an R^2 of 0.99 (which means 0.99 of the variation in TSP path length is explained by the formula) and MAPE (Mean Absolute Prediction Error) of 4.2%. This prediction error is slightly higher than when it is applied to a setting where Euclidean distances are considered (3.0%), but the formula still performs well (Figliozzi, 2008).

Merchán and Winkenbach (2019) use circuity factors to measure the relative detour incurred for traveling in a road network, compared to the Euclidean distance. This circuity factor for locations p and q is defined as:

$$c = \frac{d_c(p,q)}{d_{L_2}(p,q)} \tag{2}$$

By definition, $c \geq 1$, as the Euclidean distance is the shortest possible path between two points in a plane. Then, β_c is estimated by $\beta_c = c\beta$. This value c, is estimated for three different areas in São Paulo, for which the results are listed in Table 2. These values indicate real travel distances are on average 2.76 times longer in area 1 compared to the L_2 metric. These values were obtained by uniformly generating n locations (for n ranging from 3 to 250), computing near-optimal tour lengths under the Euclidean metric, and solving for β, then scaling by the empirical circuity factor.

Table 2: Estimates of the circuity factor c and its corresponding β_c (Merchán and Winkenbach, 2019)

	Area 1	Area 2	Area 3
c	2.76	2.34	1.82
β_c	2.48	2.10	1.64

It is important to note, however, that the assumptions in this study may limit the generality of the findings. In particular, the use of uniformly distributed locations does not accurately reflect the spatial distribution of delivery points in real urban environments, where locations tend to cluster in residential, commercial, or industrial zones. Additionally, within small urban areas, high-rise buildings and single-family homes may coexist in the same neighborhoods, further challenging the assumption of uniformly distributed

delivery points. Furthermore, The circuity factor c may vary significantly within a single city, depending on local street patterns, infrastructure, and topography. This seems plausible given observed variability in urban design, although further empirical investigation would be required to confirm this. These variations suggest that a fixed circuity factor may oversimplify the complexity of real-world delivery contexts, especially when applied to smaller sub-regions or neighborhoods.

2.2 Lin-Kernighan Heuristic

To be able to efficiently solve many TSPs, to find a good estimate for β, a fast and reliable solution algorithm is needed. The Lin-Kernighan (Lin and Kernighan, 1973) heuristic provides outcome, it is generally considered to be one of the most effective methods of generating (near) optimal solutions for the TSP. In this research a modified implementation of the heuristic is used (Helsgaun, 2000). The run times of both heuristics increase by approximately $n^{2.2}$, but the modified heuristic is much more effective. It is able to find optimal solutions to large instances in reasonable times (Helsgaun, 2000).

3 Problem definition

This section provides a formal definition of the problem this research aims to solve. Define N to be a set of neighborhoods, and define X_n the set of all postal codes in a neighborhood $n \in N$. Let $TSP_{j,n} = \{x_i\}_{i=1}^{j}$, $x_i \in X_n$ be a subset of size j of the postal codes in neighborhood n. Then, $L_{j,n}$ can be defined to be the length of the shortest path through the set $TSP_{j,n}$. This is the length of the solution of the traveling salesman problem consisting of the points in $TSP_{j,n}$. This research consists of two parts.

1. Analyze the behavior of $L_{j,n}$ for various sample sizes j and neighborhoods n, in the context of evaluating the accuracy of predictions made by Equation 1.

2. Investigate how well TSP path length can be predicted using j and a number of features of the neighborhood the TSP is in, using supervised learning.

4 Data

In this section, a detailed explanation of the data is provided. This includes the characteristics of the data used and how this data is processed.

4.1 Source

In order to model the complex nature of real road networks, data from OpenStreetMap (OpenStreetMap contributors, 2025) is used. OpenStreetMap is an open-source project that provides geographic data, including accurate and detailed information about roads, buildings and natural features around the world. The data is continuously maintained and updated by a large community of users, making it a valuable resource for this research.

This data can be downloaded from Geofabrik, and then exported to a `PostgreSQL` database using `osm2pgsql`, in order to be able to efficiently use the data with `Python`.

4.2 Description

For this analysis, the database has three interesting tables: `planet_osm_polygon`, `planet_osm_nodes` and `planet_osm_ways`. Many neighborhoods in the Netherlands have a polygon defined in the data. In OpenStreetMap a polygon is a closed shape formed by a set of geographic coordinates (`nodes`) that are connected by lines (`ways`). These objects can be used to define boundaries of geographic areas, such as lakes, parks, nature reserves and parts of cities and villages. In this research the polygons are used to filter the buildings and roads only in a certain area efficiently. These polygons are stored in `planet_osm_polygon`.

In the database, the roads are defined as `ways`. These ways have three attributes: `id`, `nodes` and `tags`. The attribute `nodes` contains an ordered list of the nodes that this road consists of. In the `tags`, a large amount of information about the way is stored, for instance whether it is a one-way road, or the type of road that it is, i.e. primary, or trunk. The information about the roads that are needed for this analysis is the road ID, the IDs of the nodes the road consists of, the coordinates (Latitude, Longitude) of these nodes, and whether the road is a one-way road.

The buildings are stored as `nodes`. In this table (`planet_osm_nodes`), a large amount of other objects are stored as well. For this analysis, the potential delivery locations need to be extracted. Some of these nodes have a postcode defined, which can be used to extract all buildings that a potential delivery could take place. This way, for example a little shed in someone's backyard is also filtered out, since this does not have its own postcode. For this research only the node ID and the coordinates are needed.

In order to perform the supervised learning analysis, the second contribution of this research, features of the neighborhoods need to be extracted. In the `planet_osm_ways` table, not only roads are stored, there is also a large amount of other objects stored in here, such as parks, bodies of water, what a certain area is used for (i.e. residential, commercial) and other natural or man-made objects. Features like this might provide predictive value for TSP path length.

4.3 Graph construction

Using the geographic information of the roads and buildings, a graph is constructed for each neighborhood, using the `igraph` module in `Python`. This graph connects all buildings to each other over the road network. First, using the information about the roads and buildings the sets of nodes and edges need to be defined. An edge is a line that connects two edges to each other. Extracting the edges that connect the road network is straightforward, since all roads already have an ordered list of nodes defined. The subsequent nodes simply need to be stored as pairs, and all edges are defined.

When the road network is defined as a set of nodes and edges, the next step is to connect the buildings to the road network. This needs to be implemented manually, since no data is stored in OpenStreetMap about to which road the buildings belong. This algorithm needs to be very efficient, since many buildings are added in each area. An R-tree can be used to accomplish this efficiency. An R-tree is a dynamic index structure that is able to retrieve data items quickly according to their spatial locations (Guttman, 1984). Listing 1 displays the algorithm used to make the edges that connect the buildings to the road network. For each building node, the closest point on the closest road is found, using the shapely implementation of the R-tree, **`STRtree`**. Then, if this point is not an already existing node, a new virtual node is added, which splits this existing edge in two

parts. The road is reconnected with the new node in between. Finally, the building is connected to this new node.

In order to find the shortest path in terms of real distance, and to calculate the length of the TSP path, a 'weight' needs to be added. This weight represents the length of this edge in meters. The equirectangular approximation is used to calculate these weights. This approximation is very efficient, but only works when the points the distance is calculated between is small enough such that the rounding of the earth does not have a significant effect on the true distance. The nodes are all close enough together, so the effect of the rounding of earth's surface is negligible. Let A and B be two nodes, and (φ_A, θ_A), (φ_B, θ_B) be their coordinates, in `radians`. Let $R = 6,371,000$ meters, Earth's radius. Then the distance between these nodes (the weight of the edge connecting them), using the equirectangular approximation is:

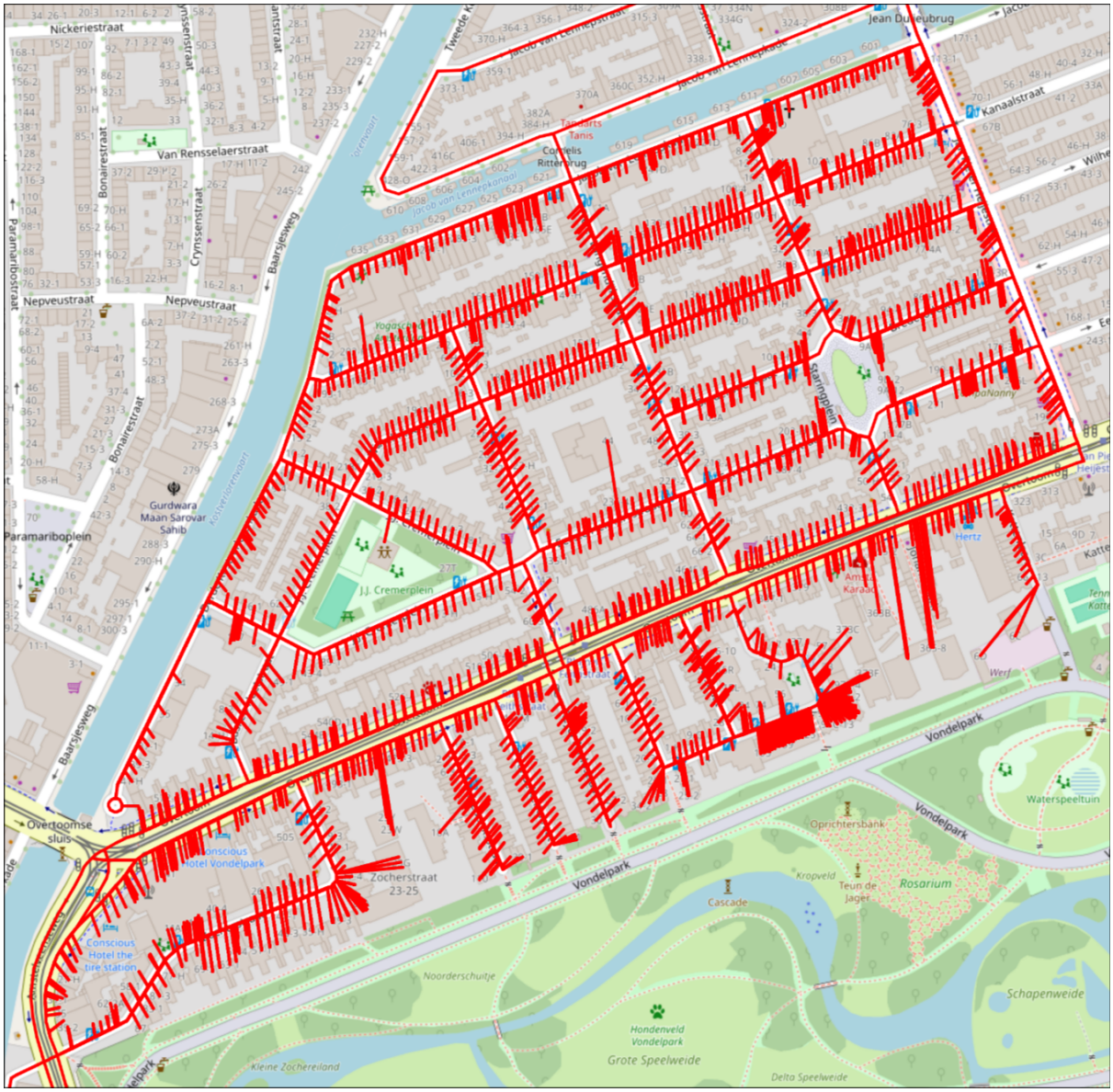
$$d(A, B) = R\sqrt{\varphi^2 + \theta^2}, \tag{3}$$

$$\text{where } \varphi = (\varphi_B - \varphi_A)\cos\left[\frac{\theta_A + \theta_B}{2}\right], \tag{4}$$

$$\text{and } \theta = \theta_B - \theta_A. \tag{5}$$

Using the `folium` module in `Python`, such a graph can be visualized on a geographic map. One of such visualizations is provided in figure 1. All maps like this one, for the areas in this analysis can be viewed and interacted with via this `webpage`.

Figure 1: Visualization of the graph, for the Overtoomse Sluis quarter in Amsterdam.



4.4 Extracting features

One of the predictors of the TSP path length, is the area of the neighborhood the locations are drawn from. The OSM data provides the area of the polygons, but this value can not be used to predict TSP path length effectively. This value significantly overestimates the correct value for A. When there are things like parks, lakes, or large buildings like a stadium inside the quarter, then in that part of the area, there are no delivery locations. Consequently, this area should not be included in the area calculation. To account for this problem, alpha shapes are used to calculate A. Alpha shapes are generalizations of the convex hull, they capture the shape of a point set (Akkiraju, Edelsbrunner, Facello, Fu, Mücke, and Varela, 1995). Figure 2 displays the map of the Beijum quarter in Groningen with the alpha shape for its addresses, as it was used for calculating the area. It can be seen that using this method, large areas without buildings, such as the greenspace in the middle is being left out, as desired.

(a) Beijum on the map

(b) The alpha shape for the addresses

Figure 2: The map of the Beijum quarter in Groningen, with the alpha shape around its addresses.

While the area of a neighborhood is an important predictor, it is only one of many features considered in this analysis. To train a supervised learning model that predicts the TSP path length based on neighborhood characteristics, a comprehensive set of features needs to be extracted.

In total, approximately 80 features are gathered for each neighborhood. These features describe various aspects of the built environment, infrastructure, and spatial distribution of delivery locations. They fall broadly into the following categories:

- **Geographic features:** Including the area of the convex hull around all postcodes, characteristics such as how much water or how many parks, and land use (i.e. residential or commercial).
- **Building characteristics:** number of buildings divided by the area of the convex hull.
- **Road network metrics:** Percentage of roads that are one-way, total length of road network divided by the area of the convex hull.

5 Experimental design

This section provides an overview of the methods and algorithms used to solve these problems.

5.1 Evaluation of BHH formula

The following algorithm is used for the first part of the research. Let $L = \{L_i\}$ and $\hat{L} = \{\hat{L}_i\}$ be the lists of observed and predicted TSP tour lengths, respectively, aggregated over all sample sizes j and repetitions k for neighborhood n. Let $|L|$ denote the total number of evaluated instances per neighborhood.

Algorithm 1 Procedure for evaluating the predictive accuracy of Equation 1

for each neighborhood $n \in N$ **do**
  Extract features of this neighborhood and save to a file for later use
  Construct a graph of the road network and visualize on a map
  Initialize empty lists for L and $\hat{L}$
  for each sample size $j \in \{20, 22, \ldots, 86, 88\}$ **do**
    for each repetition $k \in \{0, 1, \ldots, 10\}$ **do**
      Randomly sample a subset $TSP_{j,n} \subset X_n$ of size j uniformly
      Solve the TSP over $TSP_{j,n}$ to compute $L_{j,n}^{(k)}$
      Solve Equation 1 for β using these solutions $L_{j,n}^{(k)}$
      Compute the predicted length $\hat{L}_{j,n}^{(k)}$ using Equation 1 with the estimated β
      Append $L_{j,n}^{(k)}$ to list L, and $\hat{L}_{j,n}^{(k)}$ to list $\hat{L}$
  Compute the Mean Absolute Error (MAE) for neighborhood n:

$$\text{MAE}_n = \frac{1}{|L|} \sum_{i=1}^{|L|} \left| L_i - \hat{L}_i \right|$$

  Compute the Mean Absolute Percentage Error (MAPE) for neighborhood n:

$$\text{MAPE}_n = \frac{100\%}{|L|} \sum_{i=1}^{|L|} \left| \frac{L_i - \hat{L}_i}{L_i} \right|$$

  Visualize the results in graph form, a scatter plot of TSP length against number of locations, with a fitted equation 1 and a histogram of the errors $\epsilon_i = L_i - \hat{L}_i$.
  Visualize a random TSP solution on a map.
  Save the results for L to a file for later use.

5.2 Supervised learning method

In the second part of this research, a supervised learning model is trained to predict the TSP path length from neighborhood-level features and the number of points j. Specifically, a Random Forest regression model is employed.

The data set is constructed using the results from Algorithm 1. Each entry of L (the list of TSP path lengths) is a separate observation in this data set. For every observation, the corresponding area features and the number of locations j are stored in the data set. Then, the following steps are performed:

1. Split the data into a training and test set, 80/20 split.
2. Normalize the features such that they all have mean 0 and variance 1.
3. Train a Random Forest regressor to predict TSP path length L using the number of points j and neighborhood-level features as input variables.
4. Evaluate model performance on the test set using the metrics (for the test data):
   - Mean Absolute Error (MAE)
   - Mean Squared Error (MSE)
   - Coefficient of determination R^2

The goal of this part is to determine how accurately TSP path length can be predicted based on readily available spatial and demographic characteristics of neighborhoods.

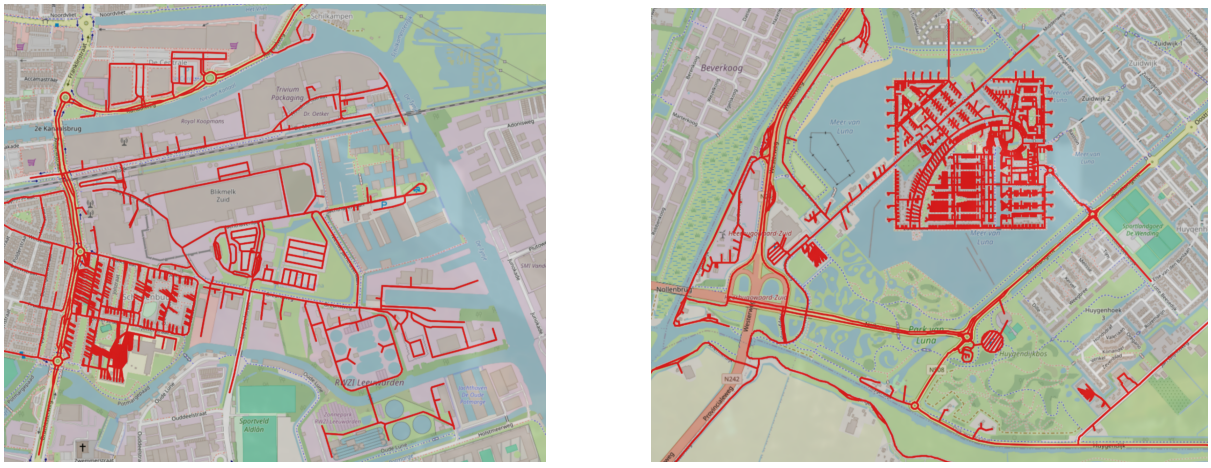
6 Results

In this section the results of the analysis that is described above is laid out. First the performance of Equation 1 is discussed, then the performance of the supervised learning approach is analyzed.

6.1 Evaluation of BHH formula

The value for β differs strongly from area to area, as expected: some areas have a estimated β as low as 1.5, while others have β close to 6. These results are tabulated in Table 3. It is found that fitting Equation 1 yields a Mean Absolute Percentage error of about 6.6%, over all observations together, when letting β differ per neighborhood ("Average" row in Table 3). The prediction accuracy differs vastly per neighborhood: the Mean Absolute Percentage Error is below 4% for some areas, and above 15% for others (using their own β). Potential reasons can best be analyzed using some of the visualizations that were created.

For example, the prediction error for the Schepenbuurt quarter in Friesland, and the Stad van de Zon quarter in Noord-Holland is especially high. These areas, among more examples, have an important feature in common. In these quarters, there is a cluster of addresses somewhere in the area, and another cluster of addresses somewhere else, with empty space between them, as can be seen in Figure 3. Both quarters have a large cluster, with some addresses spread out over the rest of the area.



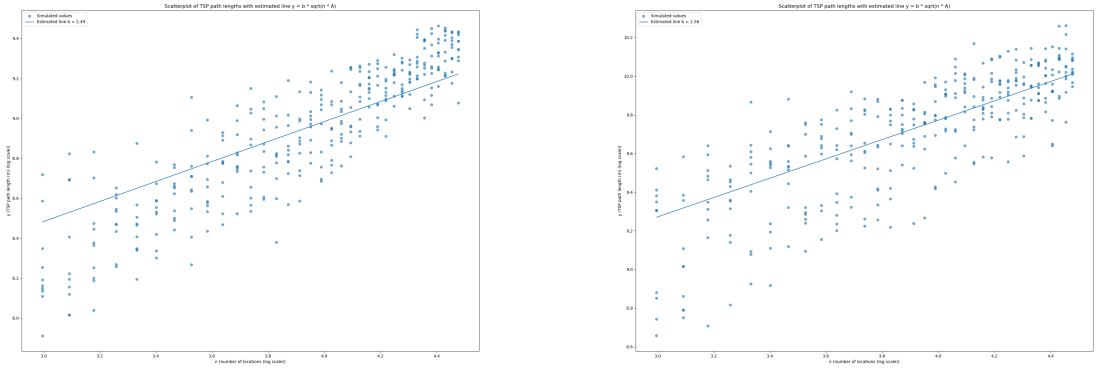
(a) Schepenbuurt (b) Stad van de Zon

Figure 3: Maps of Schepenbuurt and Stad van de Zon quarters.

An explanation for the higher prediction errors is that when the number of locations to visit j is small, and (as in these examples) one of the clusters is much smaller than the other, in the sense of number of addresses in it, that there is a significant probability that no addresses in the smaller cluster are in the TSP. This would result in a significantly lower path length, as the distance to the other cluster and back does not need to be traveled. But, there is also significant probability that there are addresses of the smaller

cluster in the TSP, which would in turn lead to a significantly higher path length. Based on this, a higher variation in the TSP path length is to be expected, which explains the higher prediction error. This problem would likely disappear when j is large enough, since then always locations from the other clusters would be chosen as well.

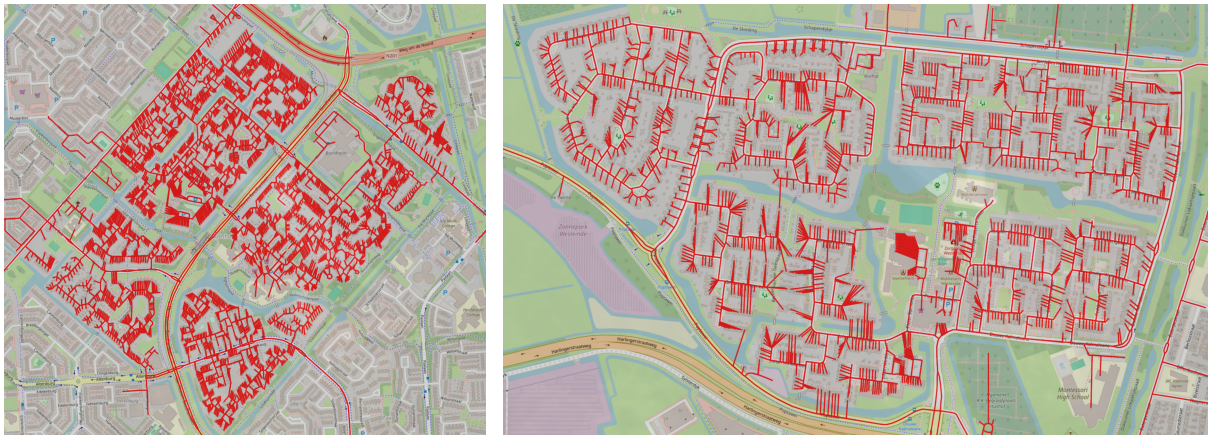
This pattern is again implied by the visualizations in Figure 4. It can be seen especially well for the Stad van de Zon scatter plot, that there are two separate clusters in the data points. There is a cluster of points far below the fitted line for the BHH formula, and a cluster of points far above it. As j increases, the amount of data points in the lower cluster decreases. Both of these observations imply these data points are for a TSP that contains locations only in the large cluster, since the probability of this happening decreases with j.



(a) Schepenbuurt (b) Stad van de Zon

Figure 4: Scatter plots of TSP path length vs number of locations for the quarters Schepenbuurt and Stad van de Zon (log-log scale).

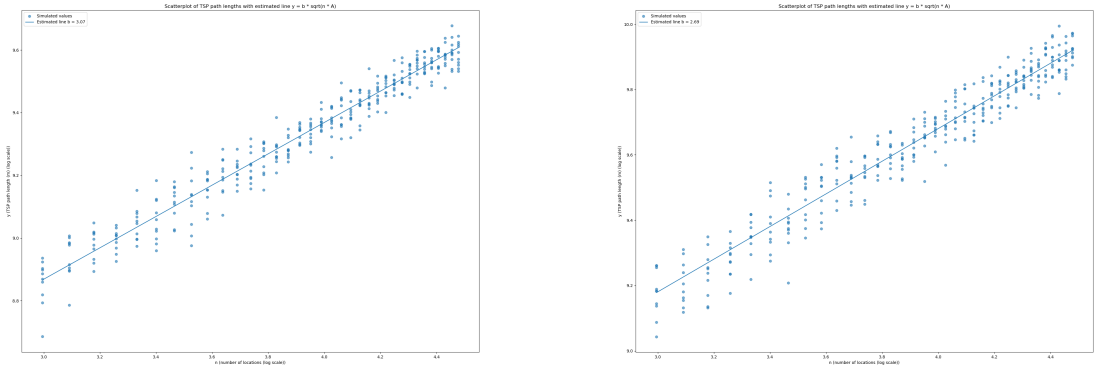
It is also interesting to consider neighborhoods where the prediction errors are especially low. Figure 5 displays maps for two quarters, Westeinde in Friesland and Bornholm in Noord-Holland. These two neighborhoods also share some features. The most important one is that the addresses seem distributed close to uniformly across the area. There are some 'clusters', for example Westeinde can visually be split into four clusters that are only connected by one or two main road connections. However, they all contain a large amount of addresses, so each TSP will likely visit all of them. Additionally, these clusters are all close to each other, there is not much empty space between them, apart from some small greenspace or water. This could explain why the formula works so much better for these areas, compared to Schepenbuurt and Stad van de Zon.



(a) Westeinde (b) Bornholm

Figure 5: Maps of the quarters Westeinde and Bornholm.

For these neighborhoods, there is still some variation around the fitted BHH formula, but Figure 6 shows vastly different levels of variation, compared to Figure 4.



(a) Schepenbuurt (b) Stad van de Zon

Figure 6: Scatter plots of TSP path length vs number of locations for the quarters Westeinde and Bornholm (log-log scale).

When β is restricted to be equal for all neighborhoods ("Total row in Table 3"), the MAPE increases to almost 22%. This is a problem for entities that want to estimate TSP path length for neighborhoods they do not know the β of, since then they would get prediction errors almost four times as high, or have to run simulations to estimate the correct value of β.

6.2 Supervised learning method

The supervised learning model results in an R^2 of 0.95, Mean Absolute Error of 943 meters, and Mean Absolute Percentage error of 6.37%, for the test sample. This is a strong improvement on the BHH formula when restricting β to be equal across areas. This method also outperforms the BHH formula when considering the estimated true values for β for each neighborhood. When the correct value for β is unknown, this model

yields far superior estimates compared to the BHH formula. However, when the true value for β is known, the difference in prediction errors becomes small.

7 Conclusion

8 References

Akkiraju, Nataraj, Herbert Edelsbrunner, Michael Facello, Ping Fu, Ernst P Mücke, and Carlos Varela (1995). Alpha shapes: definition and software. In *Proceedings of the 1st international computational geometry software workshop*, Volume 63, pp. 63–66.

Beardwood, Jillian, John H Halton, and John Michael Hammersley (1959). The shortest path through many points. In *Mathematical proceedings of the Cambridge philosophical society*, Volume 55, pp. 299–327. Cambridge University Press.

Figliozzi, Miguel Andres (2008). Planning approximations to the average length of vehicle routing problems with varying customer demands and routing constraints. *Transportation Research Record 2089*(1), 1–8.

Guttman, Antonin (1984). R-trees: A dynamic index structure for spatial searching. pp. 47–57.

Helsgaun, Keld (2000). An effective implementation of the lin–kernighan traveling salesman heuristic. *European journal of operational research 126*(1), 106–130.

Lei, H., G. Laporte, Y. Liu, and T. Zhang (2015). Dynamic design of sales territories. *Computers & Operations Research 56*, 84–92.

Lin, Shen and Brian W Kernighan (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations research 21*(2), 498–516.

Merchán, Daniel and Matthias Winkenbach (2019). An empirical validation and data-driven extension of continuum approximation approaches for urban route distances. *Networks 73*(4), 418–433.

OpenStreetMap contributors (2025). OpenStreetMap. Accessed: 2025-04-25.

Statista (2025). Global parcel shipping volume between 2013 and 2027 (in billion parcels)*.

9 Appendix

9.1 Listings

Listing 1: The algorithm to connect the buildings to the road network

```
for building in building_nodes:
    building_point: Point = building.point_lat_lon()
    nearest_index = tree.nearest(building_point)
```

```
        nearest_line: LineString = linestrings[nearest_index]
        nearest_edge: Edge = edge_map[nearest_line]
        start: Node = nearest_edge.start_node
        end: Node = nearest_edge.end_node

        projected_point: Point = nearest_line.interpolate(
            nearest_line.project(building_point)
        )
        projected_coords: tuple[float, float] = (
            projected_point.x,
            projected_point.y,
        )

        if projected_point.equals(Point((start.lat, start.lon))):
            self.add_edge(Edge(building, start))
            self.add_edge(Edge(start, building))
            continue
        if projected_point.equals(Point((end.lat, end.lon))):
            self.add_edge(Edge(building, end))
            self.add_edge(Edge(end, building))
            continue

        virtual_node_name: str =
            f"virtual_{next(node_id_counter)}"
        virtual_node: Node = Node(
            name=virtual_node_name,
            lat=float(projected_coords[0]),
            lon=float(projected_coords[1]),
            is_building=False,
        )
        new_nodes.append(virtual_node)

        self.delete_edge(nearest_edge)

        new_edges.append(Edge(start, virtual_node))
        new_edges.append(Edge(virtual_node, end))

        new_edges.append(Edge(building, virtual_node))
        new_edges.append(Edge(virtual_node, building))

self.add_vertices(new_nodes)
self.add_edges(new_edges)
```

9.2 Tables

Table 3: Empirical estimates for β, with prediction errors this beta gives for TSP path length in selected neighborhoods.

Province-Neighborhood	β	MAE (m)	MAPE (%)
groningen-Hortusbuurt	3.7867	787.0906	7.6233
groningen-Binnenstad	3.0028	944.3530	6.3492
groningen-Oosterpoort	3.1072	730.2659	7.2751
groningen-Rivierenbuurt	4.2068	747.5737	7.9876
groningen-De Wijert	2.4460	889.6834	6.1522
groningen-Oosterparkwijk	2.9148	1116.9408	6.4571
groningen-De Hoogte	3.0723	999.5812	9.1431
groningen-Korrewegwijk	2.7448	1080.9511	6.4441
groningen-Schildersbuurt	3.2861	729.9252	6.4071
groningen-Paddepoel	2.4538	729.3158	5.2808
groningen-Oranjewijk	2.9974	845.0706	6.8570
groningen-Tuinwijk	8.1263	1561.6483	14.7552
groningen-Selwerd	3.4082	782.9060	7.2691
groningen-Vinkhuizen	2.1256	743.4229	5.4375
groningen-Hoogkerk-zuid	2.4559	809.7548	6.3112
groningen-Gravenburg	2.5071	1681.3850	13.0166
groningen-De Held	4.2467	518.8655	4.4156
groningen-Reitdiep	2.7748	637.4352	5.1925
groningen-Hoornse Meer	2.7903	829.5638	7.2601
groningen-Corpus den Hoorn	2.7066	830.2959	6.7827
groningen-Eemspoort	3.4522	503.5335	4.0830
groningen-Euvelgunne	4.3578	1378.8606	9.7775
groningen-Driebond	4.0642	1604.2935	11.9177
groningen-Winschoterdiep	4.9716	2655.1958	13.9624
groningen-Eemskanaal	3.2722	619.2529	4.2074
groningen-Helpman	3.3251	863.0136	6.7162
groningen-Lewenborg	3.2654	1289.0307	6.2147
groningen-Beijum	2.5001	737.1724	4.1504
groningen-Maarsveld	3.2495	602.9083	6.2278
drenthe-Assen Oost	2.1210	1138.7330	5.4455
drenthe-Assen West	2.1581	1965.7756	10.8618
drenthe-Centrum	2.4654	1245.6113	6.6355
drenthe-Kloosterveen	2.1301	2111.6730	8.8265
drenthe-Lariks	2.4107	1149.6956	7.7865
drenthe-Marsdijk	2.1402	1857.0997	7.1529
drenthe-Noorderpark	2.3760	837.7575	5.0655
drenthe-Peelo	2.5848	860.6995	6.0705
drenthe-Pittelo	2.4930	702.9646	5.4831
drenthe-Ter Borch	3.1205	699.8987	4.0286
friesland-Achter de Hoven	3.0386	621.8266	8.0718
friesland-Aldlân	2.2496	842.2286	5.3572
friesland-Bilgaard	2.5471	907.3997	6.9380
friesland-Binnenstad en Stationskwartier	3.2897	878.3273	6.4701
friesland-Blitsaerd	3.0032	831.4206	6.1789

Province-Neighborhood	β	MAE (m)	MAPE (%)
friesland-Buitengebied Noordwest	2.6361	1581.5645	6.4575
friesland-Camminghaburen	2.2937	853.8815	4.8047
friesland-De Hemrik	2.7197	720.2788	4.4492
friesland-De Zuidlanden	2.3131	855.9684	6.0404
friesland-De Zwette	2.9526	1573.9905	6.6149
friesland-Heechterp	2.2511	824.5581	11.0612
friesland-Hempens, Teerns, en Zuiderburen	2.7328	1536.6074	6.3137
friesland-Huizum-Oost	2.9399	582.0815	4.8391
friesland-Huizum-West	2.2300	740.7885	6.4340
friesland-Middelsee	2.6517	872.5132	8.9713
friesland-Nijlân	2.4011	540.6227	5.3555
friesland-Oranjewijk	3.8885	580.7334	6.5334
friesland-Oud Oost	2.0714	778.7536	5.6172
friesland-Schepenbuurt	2.4919	1096.8162	15.6508
friesland-Schieringen en De Centrale	3.0352	923.4431	9.2577
friesland-Techum	3.2903	493.5220	5.2379
friesland-Vlietzone	2.3344	576.4998	5.8311
friesland-Vogelwijk en Componistenbuurt	3.1914	570.8451	6.5960
friesland-Vrijheidswijk	3.2394	499.8996	4.8038
friesland-Westeinde	3.0703	395.4296	3.6362
friesland-Wielenpôlle	2.9167	776.0476	10.8370
friesland-Zuiderburen	3.0009	1400.3054	5.8648
flevoland-Polderwijk	3.5001	1145.4509	5.9479
overijssel-Baalder	2.3476	717.4682	6.4571
overijssel-Baalderveld	2.3713	871.0699	6.2003
overijssel-Berflo Es	2.1673	1102.5143	6.3205
overijssel-Bergweide	3.2467	928.4136	5.8873
overijssel-De Graven Es	2.7472	1471.4149	8.5628
overijssel-De Meijbree	3.3606	542.7416	6.8872
overijssel-De Riet	2.7349	684.2054	5.2536
overijssel-De Thij	2.1556	1031.6587	7.0723
overijssel-Groot Driene	2.2700	805.2004	4.2448
overijssel-Haardijk	4.2415	1097.9543	8.5854
overijssel-Hanzeland	3.6497	748.7525	8.7452
overijssel-Hasseler Es	2.1438	1101.6226	6.0418
overijssel-Heemsermars	1.8079	508.6826	6.1520
overijssel-Het Onderdijks	2.7518	447.5038	4.3605
overijssel-Hofkamp	2.5384	689.9057	5.3236
overijssel-Ittersum	2.1437	1267.4431	6.1395
overijssel-Keizerslanden	2.6409	1831.4711	9.0887
overijssel-Kloosterlanden	2.8173	973.5713	5.8996
overijssel-Marslanden	3.7091	1607.2791	9.0062
overijssel-Nieuwe Haven	4.2211	1317.1045	7.1460
overijssel-Nieuwstraatkwartier	2.9653	627.9651	6.4148
overijssel-Noorderkwartier	2.3399	787.4246	6.0342
overijssel-OosterDalfsen	4.5740	412.0292	5.6533
overijssel-Ossenkoppelerhoek	2.4856	835.1963	5.9376

Province-Neighborhood	β	MAE (m)	MAPE (%)
overijssel-Rivierenwijk	2.7652	504.6057	4.8158
overijssel-Schelfhorst	2.4250	878.0725	4.8789
overijssel-Slangenbeek	2.1255	1138.7637	5.3182
overijssel-Sluitersveld	2.3433	1080.9257	6.7078
overijssel-Twekkelerveld	2.0640	755.1642	4.8439
overijssel-Veerallee	3.4888	806.9158	9.1483
overijssel-Voorstad	2.0326	883.2839	6.0335
overijssel-Wierdensehoek	2.6633	868.5459	5.8949
overijssel-Windmolenbroek	2.1643	1492.4184	5.8826
overijssel-Zandweerd	1.9236	819.2842	5.6229
noord holland-Schrijverswijk	3.0188	604.7393	5.5718
noord holland-Stad van de Zon	2.5632	2311.5860	15.9044
noord holland-Stadshart	4.3034	521.1349	4.8701
noord holland-Jordaan	2.8857	1214.6083	6.8566
noord holland-Slotervaart	2.1868	755.9461	4.8525
noord holland-IJburg	2.2591	898.3405	4.5361
noord holland-Oostelijke Eilanden	2.7433	623.7412	5.1655
noord holland-Oostelijk Havengebied	3.3862	1045.0941	4.7407
noord holland-Frederik Hendrikbuurt	4.1417	1038.7155	8.9046
noord holland-Van Lennepbuurt	3.9625	768.0134	6.5499
noord holland-Da Costabuurt	7.9045	930.6030	8.6875
noord holland-Kinkerbuurt	6.4267	998.1282	8.7462
noord holland-Kersenboogerd	2.2194	1145.6057	5.2513
noord holland-Pax	3.5385	562.4764	3.9765
noord holland-Graan voor Visch	3.1503	547.0381	5.5134
noord holland-Vrijschot-Noord	4.5905	710.2193	8.0498
noord holland-Toolenburg	2.5113	987.3229	4.3092
noord holland-Floriande	2.6981	1117.5578	4.3851
noord holland-Overbos	2.2915	651.6643	4.0632
noord holland-Bornholm	2.6886	642.8920	4.3094
noord holland-Beukenhorst-Oost	3.9371	1040.6298	8.1963
noord holland-De Hoek	5.1390	1003.2119	6.2698
noord holland-West	3.0370	446.6162	5.8976
noord holland-Zuid	4.1365	903.2890	5.8866
noord holland-Oost	3.1791	766.3976	5.2967
noord holland-Noord	2.8073	706.6457	5.5368
noord holland-De President	2.7258	1122.8178	11.4476
noord holland-Graan voor Visch-Zuid	3.7450	687.4784	6.8385
noord holland-Zuidwijk	3.1964	456.7274	4.0636
noord holland-Buitenveldert-West	1.9102	1218.8317	6.4754
noord holland-Buitenveldert	1.7762	1492.3560	6.9382
noord holland-Apollobuurt	2.7182	933.7595	7.0048
noord holland-Stadionbuurt	2.6951	972.0317	7.9713
noord holland-Prinses Irenebuurt e.o.	4.1578	578.6442	6.8676
noord holland-Hoofddorppleinbuurt	2.8949	1169.3999	8.5072
noord holland-Willemspark	5.1071	1013.1195	8.3730
noord holland-Schinkelbuurt	5.7724	1289.2277	11.4147

Province-Neighborhood	β	MAE (m)	MAPE (%)
noord holland-Vondelparkbuurt	4.9184	855.1568	8.8287
noord holland-Helmersbuurt	3.6774	814.9797	6.9407
noord holland-Overtoomse Sluis	5.9286	746.8328	6.8497
noord holland-Museumkwartier	2.5895	1145.3162	7.3443
noord holland-Rivierenbuurt	2.3722	1242.1375	6.8000
noord holland-IJselbuurt	4.4211	1027.3968	8.9081
noord holland-Scheldebuurt	3.2151	970.1862	6.5056
noord holland-Rijnbuurt	4.1140	593.4113	5.2244
noord holland-De Baarsjes	2.6001	1255.6146	6.0825
noord holland-Landlust	3.2295	1084.6529	7.1576
noord holland-Staatsliedenbuurt	2.8589	815.2870	7.0086
noord holland-Spaarndammerbuurt	4.2194	811.4117	7.0284
noord holland-De Pijp	2.6711	1361.7288	6.8957
noord holland-Grachtengordel	3.0767	1425.1466	7.2483
noord holland-Oud-Zuid	1.9108	1461.1687	5.7338
utrecht-Bedrijventerrein Vathorst	4.5724	952.7612	5.2408
utrecht-Binnenstad City- En Winkelgebied	2.1682	687.2519	6.5254
utrecht-Binnenstad Woongebied	3.5171	1100.1967	5.6842
utrecht-Bosgebied	2.2748	1367.0046	6.3416
utrecht-Buitengebied Oost	2.1416	1397.7654	6.7521
utrecht-Calveen	3.9526	899.3645	8.5463
utrecht-De Berg-Noord	2.4867	821.9696	5.0387
utrecht-De Berg-Zuid	2.4160	932.6851	6.0254
utrecht-De Hoef	3.0760	1110.2511	8.6874
utrecht-De Koppel	3.2872	476.2108	5.2298
utrecht-Dichterswijk, Rivierenwijk	2.3013	787.9536	5.2042
utrecht-Eemkwartier	4.2864	695.4076	7.9055
utrecht-Hoogland	2.4644	1054.4349	5.7404
utrecht-Hoogland-West	1.9501	1942.9992	7.0928
utrecht-Hooglanderveen	3.4302	676.3181	4.4060
utrecht-Isselt	3.3007	999.1430	5.5305
utrecht-Kanaleneiland	2.2929	684.5170	4.6111
utrecht-Kattenbroek	2.6279	840.8940	5.0859
utrecht-Kruiskamp	3.0583	778.2176	6.4218
utrecht-Leusderkwartier	2.2569	1041.8890	7.3030
utrecht-Liendert	2.7391	590.8835	4.6034
utrecht-Lombok, Leidseweg	2.6515	848.2440	5.8439
utrecht-Nederberg	5.0700	731.6925	6.7624
utrecht-Nieuw Engeland, Schepenbuurt	2.5904	2120.2284	6.9265
utrecht-Nieuwland	2.5847	838.1784	4.7925
utrecht-Oog in Al	2.6330	1044.5446	7.2653
utrecht-Randenbroek	2.7071	782.4703	5.3747
utrecht-Rustenburg	2.6395	395.7474	3.9862
utrecht-Schothorst-Noord	2.2742	954.6565	6.7492
utrecht-Schothorst-Zuid	3.3918	649.1111	4.5797
utrecht-Schuilenburg	2.7636	552.0899	5.6674
utrecht-Soesterkwartier	2.4908	1101.6736	6.9427

Province-Neighborhood	β	MAE (m)	MAPE (%)
utrecht-Stadskern	3.8055	865.9577	5.9803
utrecht-Terwijde	2.9249	735.9790	3.9313
utrecht-Vathorst-Centrum	4.0171	774.4471	6.0928
utrecht-Vathorst-De Laak	2.5626	1214.6085	5.9675
utrecht-Vathorst-De Velden	2.9738	737.7602	4.2696
utrecht-Veldhuizen	2.5302	714.6507	5.2290
utrecht-Vermeerkwartier	2.3126	696.3630	5.2603
utrecht-Zielhorst	2.7619	833.9362	6.3967
gelderland-Aldenhof	3.7633	556.8623	5.8121
gelderland-Altrade	2.9492	783.9319	6.2079
gelderland-Benedenstad	3.4464	390.0724	4.3911
gelderland-Biezen	2.9673	827.2408	5.6937
gelderland-Bijsterhuizen	4.6890	1566.6764	12.7655
gelderland-Bottendaal	4.1688	659.9065	6.9027
gelderland-Brakkenstein	2.1399	754.9688	6.7204
gelderland-De Hoop	3.1701	522.6107	5.6437
gelderland-De Horsten	3.3684	566.2992	7.1855
gelderland-De Huet	2.2084	629.8263	3.9225
gelderland-De Kamp	3.3599	959.6935	6.1985
gelderland-De Pas	3.4216	692.0413	7.0011
gelderland-Dichteren	2.9144	1103.4474	7.4356
gelderland-Druten-Zuid	2.1562	765.5168	4.9191
gelderland-Ede-Zuid	2.1184	862.7631	6.0182
gelderland-Enka	3.2325	497.8380	4.9358
gelderland-Galgenveld	3.0197	942.6533	6.8689
gelderland-Goffert	2.3112	977.5300	5.7219
gelderland-Groenewoud	2.7905	633.9448	4.6512
gelderland-Grootstal	2.7495	752.7934	5.8744
gelderland-Hatert	2.4068	792.0939	5.0487
gelderland-Haven- en industrieterrein	2.9806	1524.4291	8.3785
gelderland-Hazenkamp	2.6265	822.3193	5.7842
gelderland-Hees	2.7873	556.8466	4.3939
gelderland-Heideslag	3.1727	925.1216	11.0459
gelderland-Heijendaal	2.9230	1183.4906	8.4884
gelderland-Hengstdal	2.8132	626.3845	4.6735
gelderland-Heseveld	3.1152	987.7851	6.6970
gelderland-Hunnerberg	2.8385	1043.7581	7.4868
gelderland-Kerkenbos	3.4369	931.5513	9.1516
gelderland-Kernhem	2.5550	1654.5925	9.7877
gelderland-Kortenoord	3.5167	406.3872	3.6853
gelderland-Kwakkenberg	2.3778	883.1879	6.4522
gelderland-Lankforst	4.2118	420.6216	4.6022
gelderland-Lent	2.5252	1670.4846	6.7304
gelderland-Maandereng	2.4887	569.2160	4.9578
gelderland-Malvert	4.3000	517.8930	5.0788
gelderland-Meijhorst	4.6143	523.7948	4.3021
gelderland-Methen	2.8990	385.6225	3.3874

Province-Neighborhood	β	MAE (m)	MAPE (%)
gelderland-Neerbosch-Oost	2.8874	1083.8216	6.9023
gelderland-Nije Veld	2.7230	690.8362	5.3239
gelderland-Noordwest	2.7652	500.2537	4.2772
gelderland-Nude	2.6217	521.1733	4.5426
gelderland-Oosseld	2.2472	721.2955	5.7656
gelderland-Oosterhout	2.8500	907.2217	5.2443
gelderland-Ooyse Schependom	4.3564	541.3111	9.9971
gelderland-Overstegen	3.0606	538.8126	3.6453
gelderland-Ressen	2.2535	2499.0589	15.6158
gelderland-Rietkampen	3.4892	568.7235	3.7161
gelderland-Rijkerswoerd	2.2949	768.0688	4.4788
gelderland-Roodwilligen	3.2931	214.7114	5.4052
gelderland-Schöneveld	3.0681	512.8591	5.2261
gelderland-Staddijk	3.9067	970.2252	5.1446
gelderland-Stadscentrum	2.9141	1043.4735	7.1672
gelderland-Stadsweiden	2.4108	502.7610	3.7071
gelderland-Tolhuis	3.5943	1161.7093	8.3833
gelderland-Veldhuizen	2.2858	673.1475	4.1493
gelderland-Weezenhof	3.3834	637.6382	4.4508
gelderland-Westkanaaldijk	3.9115	911.8447	5.6946
gelderland-Wijnbergen	3.3197	382.2284	4.4317
gelderland-Wolfskuil	3.0242	801.0343	6.4680
gelderland-Zwanenveld	3.4467	454.3413	3.7634
zuid holland-Afrikaanderwijk	4.4672	755.9314	6.4506
zuid holland-Berkel	2.6769	732.0691	5.5160
zuid holland-Beverwaard	1.9310	740.6548	5.6274
zuid holland-Binnenstad	1.2998	1091.9128	6.1208
zuid holland-Bloemendaal	2.2079	824.0422	4.3516
zuid holland-Bloemhof	3.2644	1095.6418	7.3129
zuid holland-Bospolder	3.3514	878.6328	7.5607
zuid holland-Buitenhof	2.8536	1278.5928	7.7260
zuid holland-Capelle-West	2.9026	736.4316	7.1468
zuid holland-Carnisse	3.0227	983.6190	7.5457
zuid holland-Cool	4.5108	1248.6295	8.0823
zuid holland-De Schenkel	3.2276	543.5715	4.9991
zuid holland-Delfshaven	4.3688	981.4805	7.7313
zuid holland-Fascinatio	2.6675	1086.6211	9.9432
zuid holland-Feijenoord	3.2514	668.7462	5.7364
zuid holland-Goverwelle	2.5131	850.3882	6.2137
zuid holland-Groente- en Fruitmarkt	3.7214	667.9367	6.3517
zuid holland-Groot-IJsselmonde	2.1149	1385.9490	5.7558
zuid holland-Heijplaat	3.9333	696.5381	7.5029
zuid holland-Hellevoet	2.9902	1068.7121	6.3925
zuid holland-Het Lage Land	3.1009	965.0769	6.3200
zuid holland-Hillegersberg-Noord	2.2673	1412.1092	8.7149
zuid holland-Hillesluis	3.2361	1022.6123	6.9756
zuid holland-Hof van Delft	2.1444	805.8094	5.8004

Province-Neighborhood	β	MAE (m)	MAPE (%)
zuid holland-Hoog Dalem	3.3333	463.4984	3.8859
zuid holland-Katendrecht	3.6542	974.0655	7.8884
zuid holland-Kleinpolder	2.9014	664.1459	5.0661
zuid holland-Kleiwegkwartier	2.7385	1105.2103	7.7506
zuid holland-Kop van Zuid-Entrepot	3.9468	1013.3788	7.3530
zuid holland-Kort Haarlem	2.5009	799.0229	5.5994
zuid holland-Korte Akkeren	2.0977	1048.7966	7.4106
zuid holland-Leyenburg	2.8984	1182.8900	7.1212
zuid holland-Lombardijen	2.7602	1012.2313	6.0138
zuid holland-Middelland	4.2484	1405.2185	8.7951
zuid holland-Middelwatering	2.3515	1199.0042	6.0586
zuid holland-Moerwijk	2.5645	1037.6902	5.6828
zuid holland-Molenlaankwartier	2.0507	1289.8516	7.8659
zuid holland-Morgenstond	2.8001	1169.2886	5.8592
zuid holland-Nesselande	2.6024	1201.7011	5.3897
zuid holland-Nieuw-Helvoet	2.8235	892.1968	5.9277
zuid holland-Nieuw-Mathenesse	4.4814	2485.7026	12.1612
zuid holland-Nieuwe Westen	2.5469	1053.2857	6.8357
zuid holland-Noord	2.2952	1034.7771	6.0713
zuid holland-Noordereiland	4.6708	623.5225	6.9333
zuid holland-Ommoord	2.7322	1142.8971	4.7820
zuid holland-Onyx	3.9642	351.0077	6.8568
zuid holland-Oosterflank	2.5586	986.9776	6.4671
zuid holland-Oostgaarde	2.4343	1008.7880	5.7550
zuid holland-Oud-Charlois	2.7080	991.6265	6.1135
zuid holland-Oud-IJsselmonde	2.9006	1100.4647	5.5309
zuid holland-Oud-Mathenesse	3.2260	792.9639	6.9276
zuid holland-Oude Westen	4.3041	1300.9461	8.6358
zuid holland-Overschie	2.6802	932.9413	6.1333
zuid holland-Pendrecht	2.8739	841.9458	6.3492
zuid holland-Plaswijck	2.4852	1000.2560	5.7854
zuid holland-Portlandsehoek	2.8983	533.4824	5.3000
zuid holland-Prinsenland	3.1325	676.7650	4.1853
zuid holland-RijswijkBuiten	3.1914	827.7648	5.3352
zuid holland-Rivium	3.9774	578.3450	6.4807
zuid holland-Ruiven	1.9212	1168.6130	8.7792
zuid holland-Rustenburg en Oostbroek	2.6635	1048.4608	6.4569
zuid holland-Schenkel	2.6379	705.7863	5.2924
zuid holland-Schiebroek	1.8483	1519.2914	7.7388
zuid holland-Schieweg	2.5122	1185.2443	8.3900
zuid holland-Schollevaar	2.6518	2209.5557	9.1045
zuid holland-Spangen	4.1717	888.4093	6.9012
zuid holland-Stadsdriehoek	3.0297	1132.8572	5.9851
zuid holland-Stolersluis	3.4386	918.4346	10.0803
zuid holland-Tanthof-Oost	2.7630	538.2743	4.8544
zuid holland-Tanthof-West	2.9826	750.5216	6.1426
zuid holland-Tarwewijk	2.7419	1128.1956	7.8602

Province-Neighborhood	β	MAE (m)	MAPE (%)
zuid holland-Transvaalkwartier	2.9510	1115.0289	7.2378
zuid holland-Tussendijken	5.5291	1038.8885	8.7457
zuid holland-Vogelbuurt	3.1903	379.0538	4.4785
zuid holland-Voordijkshoorn	3.0494	1023.5672	5.1693
zuid holland-Voorhof	3.3030	755.0064	5.6075
zuid holland-Vreewijk	2.5475	1340.8887	6.5759
zuid holland-Vrijenban	2.2327	1052.3212	7.2528
zuid holland-Wateringse Veld	2.1445	1024.2319	4.4440
zuid holland-Westergouwe	2.9003	1447.7450	8.5294
zuid holland-Westplaat	2.5707	495.1839	5.1190
zuid holland-Westpolder	3.3102	1222.4980	7.3129
zuid holland-Wielewaal	4.1737	673.9397	8.1491
zuid holland-Wippolder	2.3678	839.7443	5.0233
zuid holland-Zestienhoven	2.6052	1440.4121	6.3112
zuid holland-Zevenkamp	2.2138	1007.0375	5.9420
zuid holland-Zuid	3.1967	447.6298	5.1557
zuid holland-Zuiderpark	2.9400	814.2675	7.4086
zuid holland-Zuidwijk	2.2036	800.1075	5.8828
noord brabant-Binnenstad	2.0433	1061.1476	5.4125
noord brabant-Brandevoort	2.1454	1839.7531	9.5962
noord brabant-Brouwhuis	2.0615	1609.2903	8.9486
noord brabant-Dierdonk	2.1509	2250.6841	13.1282
noord brabant-Helmond-Noord	2.3109	855.0128	4.8511
noord brabant-Helmond-Oost	2.2907	845.5918	6.0105
noord brabant-Helmond-West	2.7346	914.0061	7.2283
noord brabant-Industriegebied Zuid	2.3321	1882.7544	6.9277
noord brabant-Mierlo-Hout	1.8139	858.5567	4.3769
noord brabant-Oosterheide	2.1485	1295.6620	7.2752
noord brabant-Rijpelberg	2.0572	3279.7709	18.9678
noord brabant-Stiphout	1.8670	1479.6214	6.8681
noord brabant-Warande	2.6017	881.0406	4.7756
noord brabant-West	2.1057	2131.3435	7.8855
noord brabant-Zuidoost	1.8123	2389.1777	11.3801
limburg-Blerick-Midden	2.8199	775.1008	5.9964
limburg-Blerick-Noord	2.7990	768.3499	6.2648
limburg-Blerick-Zuid	3.1077	468.5901	5.0028
limburg-Boekend	2.3085	1603.2939	11.3346
limburg-Centrum	2.2607	1642.7264	7.0409
limburg-Donderberg	2.4346	577.4141	4.5257
limburg-Hoogvonderen	2.9420	379.7682	3.7412
limburg-Hout-Blerick	2.1335	1632.7321	9.6315
limburg-Klingerberg	2.5419	613.5472	6.3093
limburg-Lindenheuvel	2.1341	1057.2441	6.1935
limburg-Maasniel	2.0842	1445.4235	7.2838
limburg-Maastricht-Centrum	3.1251	1329.2311	5.8712
limburg-Maastricht-Noordwest	2.9473	998.9314	6.5367
limburg-Maastricht-Oost	1.8391	1334.7535	4.7881

Province-Neighborhood	β	MAE (m)	MAPE (%)
limburg-Maastricht-West	1.9095	1194.1614	4.4141
limburg-Maastricht-Zuidoost	1.7178	1189.8679	5.0765
limburg-Meuleveld	3.9116	665.7765	6.9355
limburg-Roermond-Oost	2.0713	743.0770	5.6894
limburg-Roermond-Zuid	1.8364	2222.7765	11.9935
limburg-Sanderbout	3.5961	793.4146	8.2930
limburg-Vossener	3.0752	541.2249	5.7338
zeeland-Binnenstad	2.3986	1169.1696	7.9141
zeeland-Burgh	3.0609	647.7810	7.3633
zeeland-Haamstede	2.9629	647.5954	4.9538
zeeland-Lammerenburg	2.2395	1272.1640	5.7762
zeeland-Middelburg-Zuid	2.8707	1052.4194	7.1346
zeeland-Othene	2.9252	861.1507	5.6943
zeeland-Paauwenburg	2.1851	1093.2731	7.0659
zeeland-Terneuzen-Centrum	3.5215	668.4812	5.7383
zeeland-Terneuzen-Noord	2.5650	804.5580	4.9723
zeeland-Terneuzen-West	2.3211	1023.0001	6.7493
zeeland-Terneuzen-Zuid	2.2961	653.2935	4.1900
zeeland-West-Souburg	2.9608	786.4329	6.8031
Average	2.9643	957.7539	6.5862
Total	2.9643	3479.5485	21.9348