## 1 Introduction

Bike-sharing rentals have been a new and environmentally friendly trend shaping modern urban life for quite some time. Among the benefits of bike-sharing rentals, local citizens can be more flexible in arranging their travel plans and destinations, while workers can save time for their first and last mile. In short, bike-sharing rentals can greatly enhance traffic capacity and efficiency in the city.

As the research objective of our term paper, MOBIbike is considered the largest bike rental company in Germany. It boasts a highly active user base and an extensive usage zone throughout the city. Using the data provided by MOBIbike, the main task of our project will be to identify important variables influencing MOBIbike demand in Dresden. Using this information the project aims to develop prediction models to predict bike demand by using learners such as Random Forest, Boosting and Geographically Weighted Regression.

## 2 Main Body

Chapter 2.1 MOBIbike Data

### *2.1.1 MOBIbike data description*

The MOBIbike company has provided us with a substantial volume of data spanning from February to April. This raw data presents us on every minute of every day throughout the observation period. Table 1.1 provides us an overview of MOBIbike data in its raw form.

*Table 1: Columns and description of MOBIbike Data*

| *Variable* | *Description* | *Variable* | *Description* |
|---|---|---|---|
| *City* | *City Location* | *maintenance* | *Maintenance Assessment* |
| *timestamp* | *Timestamp accurate to the second* | *terminal_type* | *Bike free to use or not* |
| *uid* | *Last User ID used this Bike* | *Place_type* | *Numeric Variable (description not provided)* |
| *lat* | *Latitude Data* | *rack_locks* | *Has rack lock or not* |
| *lng* | *Longitude Data* | *no_registration* | *Status of officially registration* |
| *bike* | *Located in bikestation or not* | *bike_number* | *ID of MOBIbike* |
| *name* | *Name of bikestation* | *bike_types* | *Has electric lock or not* |

| | | | |
|---|---|---|---|
| *station_number* | *ID of the bikestation* | *active* | *Bike's utilisation status at the given timestamp* |
| *Booked_bikes* | *Reservation status* | *state* | *Availability of station* |
| *Bikes_availabe to rent* | *Number of available bikes in the station* | *electric_lock* | *Status of electrical lock,work or not* |
| *bike_racks* | *Bike racks in station* | *boardcomputer* | *Board computer ID* |
| *free_racks* | *Available racks in station* | | |

*2.1.2 Pre-processing the raw data*

The table above not only contains bike-related data but also includes information from the bike station. As our task pertains specifically to the MOBIbike, we would like to focus only on the bike-related information.
Considering our task involves analysing MOBIbike distribution, we need only the following columns from its raw data frame: the ID of MOBIbike; the timestamp for merging three months data together; and precise geographical data, such as longitude and latitude for individual bikes.

Because our observation period is three months, with each minute (of each day) having ~999 records, the total number of raw observations for MOBIbike would roughly be around 1,438,560 multiplied by 89 days. This rough calculation illustrates how huge the data actually is. After subsetting the data, the final variables used for analysis are shown in Table 2.

*Table 2: Subsetted columns of the MOBIbike Data*

| (Row Index) | time_stamp | lat | lng | bike_number |
|---|---|---|---|---|
| 1 | 2023-02-01 00:00:00 | 51.0433 | 13.78804 | 930010 |
| … | | | | |
| 999 | 2023-02-01 00:01:00 | 51.04664 | 51.04664 | 931131 |
| … | | | | |
| 1999 | 2023-02-01 00:02:00 | 51.07268 | 13.73085 | 930870 |

*2.1.3 Visualising a distribution instance on a map*

By using the spatial coordinates of MOBIbikes during the observation period along with geodata of Dresden's borders, we get Figure 1 i.e. the distribution of MOBIbike in Dresden for the instance *2023-02-01 00:00:00*. The maps in the project have been created using leaflet and ggplot2 packages in R. The bike location information has been geo-referenced using the EPSG coordinate reference system. Throughout this project, the projection used for the spatial reference is EPSG:4326. The blue points represent the location of MOBIbikes and the polygon with light yellow background indicating the area of Dresden.

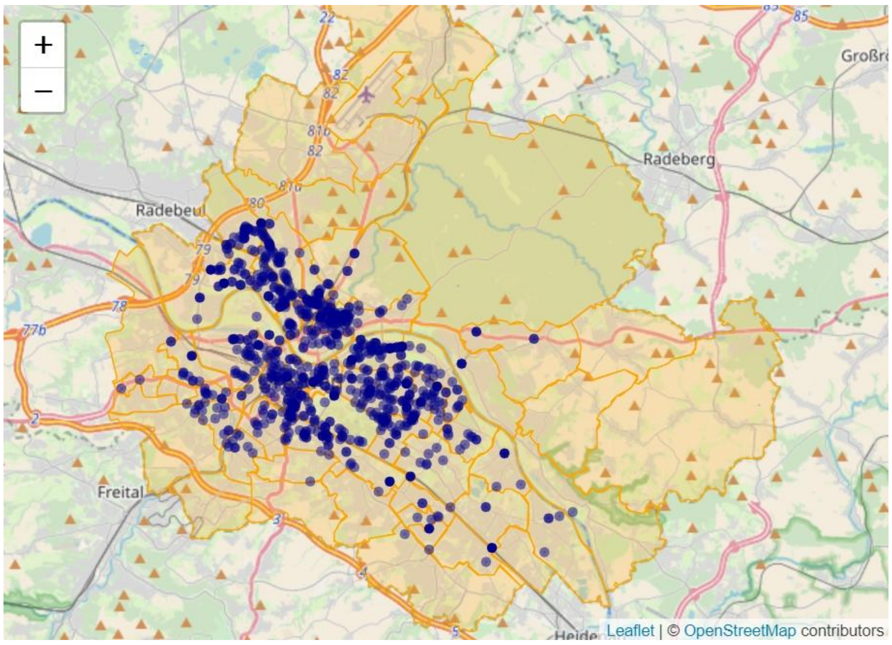

*Figure 1: Bike distribution for 2023-02-01 00:00:00*

### *2.1.4 Convex Hull and Outliers*

After analysing multiple instances, it is found that the distribution of bikes is not spread out throughout Dresden. The areas particularly along the outskirts of Dresden don't include a significant number of bikes, for the data used in this project. The core area where the bulk of MOBIbike distribution is found is visualised in Figure 2. The blue region represents the core area for bike distribution in Dresden.

In the raw data, some bikes were found to be outside Dresden. These bikes (outliers) have been filtered out during the pre-processing stage so that the data being used for the project has bikes that are present inside Dresden only. This was done by using the 'st_intersection' function that is located in the 'sf' package in R. Only the data points that are intersecting the polygon of Dresden have been preserved.

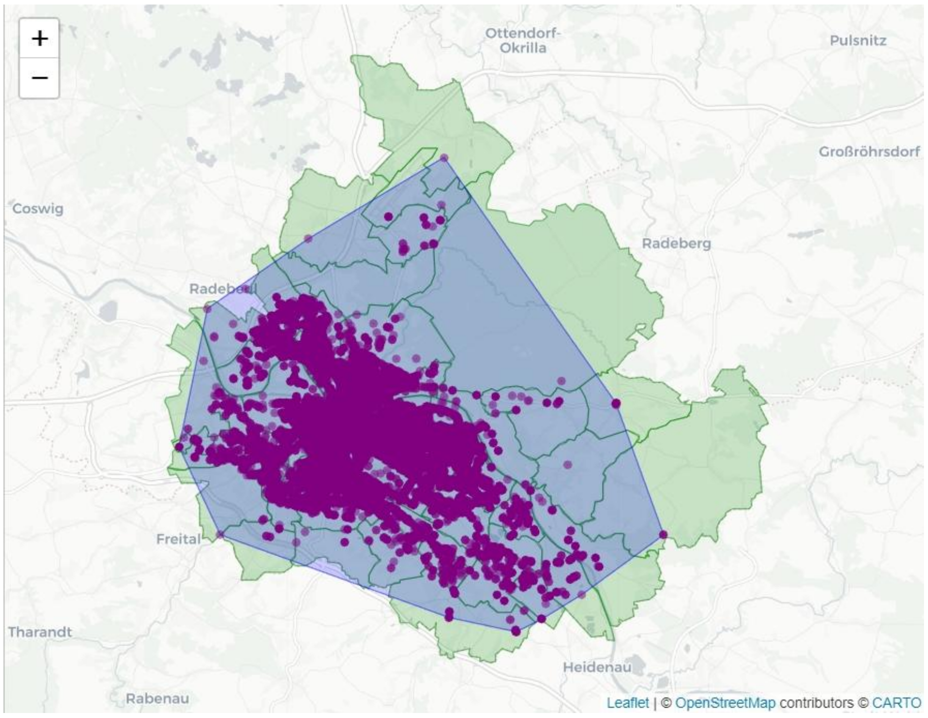

*Figure 2: Convex Hull*

### *2.1.5 Training and Test Data*

For training and performance assessment of the models created in this project ahead, various ways of performing resampling could have been chosen. Some resampling methods such as bootstrap, k-fold cross validation and sub-sampling are generally more appropriate when data is small in size. Since the data used in this project is very large, training and test datasets were created from the original data using a simple holdout approach. A holdout approach splits the entire data exactly into two parts, with one part assigned as training data and the other part as test data. Usually the split is unequal, with the larger part (> 50%) being the training data. Additionally in each of training and test datasets, the observations chosen are selected at random such that no observation is common in both datasets.

However the complete data could not be used for the holdout approach. This is because the data could not be loaded in the computer memory due to the huge size. To avoid this problem, the holdout approach is slightly altered. Two random samples are drawn from the entire data, where one sample is assigned as the training data and the other sample as the test data. After assessing how many observations the computer memory can accommodate, a sample of ~240000 observations is chosen for training data and a sample of ~120000 observations is chosen for test data. Both samples are prepared in such a manner that no observations between the two samples are common.

## Chapter 2.2 Clustering

The previous chapter shows what the bike distribution looks like within Dresden. However, Dresden is a very large area. For any customer using MOBIbikes, he/she is not likely to rent any bike available anywhere in Dresden. Rather, he/she would likely prefer to rent a bike that is in close proximity to the individual’s existing location. Hence, it is more appropriate to perform analysis on the data after segmenting the bike distribution into smaller representative groups. This is done by carrying out clustering. There are other motivations to carry-out clustering which are explained in later sections.

### *2.2.1 Clustering Methods*

There are various clustering algorithms available, but for our project, we are focusing on comparing the three most common clustering algorithms: DBSCAN, Hierarchical clustering, and K-means. DBSCAN is a density-based clustering algorithm. Given that our project involves spatial data and the variables used in the analysis are computed based on regional densities, DBSCAN is particularly well-suited for our study. However, this approach does come with certain limitations. Firstly, we need to define appropriate values for the radius and minimum points parameters before running the algorithm. Identifying values that result in optimal outcomes can be challenging. Secondly, in areas where the number of residences is low, the count of bikes is also lower. In these regions, where observations are less dense, DBSCAN's performance may be less effective. Taking these factors into consideration, we have decided not to pursue this approach. For hierarchical clustering algorithms, which include both agglomerative and divisive clustering methods, linkage methods are used to define distances among groups. There are four linkage methods: single linkage, complete linkage, centroid linkage, and average linkage. It is uncertain which linkage method would result in a good model fit for our project. Additionally, this type of clustering algorithm is computationally expensive for large datasets. Since in this project the dataset is very large, using this approach is not considered to be very practical. As a result, we have decided to employ the K-means clustering algorithm.

The core idea of the K-means clustering algorithm revolves around defining the value of 'k,' which signifies the desired number of clusters to be formed. In our project, selecting an overly small value for 'k' would render the results less meaningful. When individuals notice that bikes are placed in locations far away from them, they are likely to abandon the idea of using bikes and instead opt for trams or buses. Clearly, identifying an appropriate value for 'k' can significantly reduce the distance for bike users, encouraging them to choose bikes as their mode of transportation. Once the number 'k' is established, the algorithm proceeds by randomly selecting 'k' data points as initial cluster centres. It then calculates the Euclidean distance between each data point and these initial centres. The data points with the shortest Euclidean distances are assigned to their respective clusters. Upon forming new clusters, new centroids are determined by calculating the mean of all the data points within each cluster. This process is repeated iteratively until the data points within each cluster no longer change.

However, the K-means clustering algorithm also presents certain limitations. Firstly, the choice of random initial means significantly impacts the results, leading to variations each time the code is executed. We plan to use the 'set.seed' function to address this issue, making sure that the results we get are reproducible. Secondly, the determination of the appropriate number of clusters 'k' also holds substantial sway over the results. To accommodate this issue ‘k’ has been chosen to be ‘100’, so that the clusters are small enough to capture variation in spatial relationships (explained later) throughout the entire bike distribution and are ‘realistic’ enough (around the core area) to approximate the proximity within which a MOBIbike user would search a bike. Furthermore, the K-means clustering algorithm is primarily suited for identifying convex clusters. Due to the nature of its objective function, which monotonically decreases, achieving convergence can be challenging. Convergence only occurs when data points within each cluster remain unchanged, allowing the algorithm to arrive at an optimal solution. In cases involving non-convex clusters, different initializations may yield divergent outcomes.Finally, the K-means algorithm is sensitive to outliers. In our project, for instance, some bikes are located outside Dresden, significantly distanced from the majority of bikes. When mean values are employed as centroids, these outliers disproportionately influence the results. Consequently, we have taken steps to preprocess our data, excluding outliers to establish reliable data.

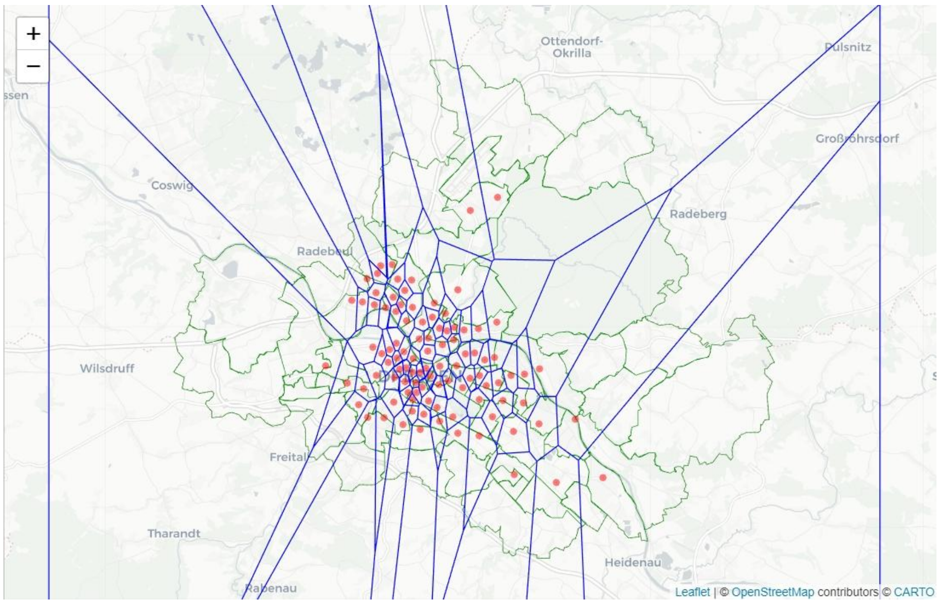

*Figure 3: Voronoi Diagram for Training Data*

### *2.2.2 Interpreting Clusters*

Figure 3 and Figure 4 visualise how the entire bike distribution is segmented, for both training and test data. Each cluster is represented by a convex polygon and the red points represent the spatial centroid for each cluster respectively. In the context of spatial modelling, the red points can be seen as regression point/data point/observation embodying a representative value for the entire cluster it belongs to. In other words, a red point can be interpreted as a virtual MOBIbike station providing us with valuable information about the area it covers (cluster perimeter).

The GeoPoints that are shown in figure 1 and figure 2 are merely bike locations that are plotted on a map. However to construct meaning variables for analysis, it is essential to use the information from all the GeoPoints and the geographical area present inside a cluster. The approach to calculate values for target and predictor variables is explained in chapter 2.3. But the core idea behind the approach is to make use of different aggregation methods (per cluster) to generate a single observation for all the variables. Since the observations created are dependent on the clusters, we have performed clustering separately for training and test data so that different clusters are formed in each dataset. This ensures that training observations are different from test observations.

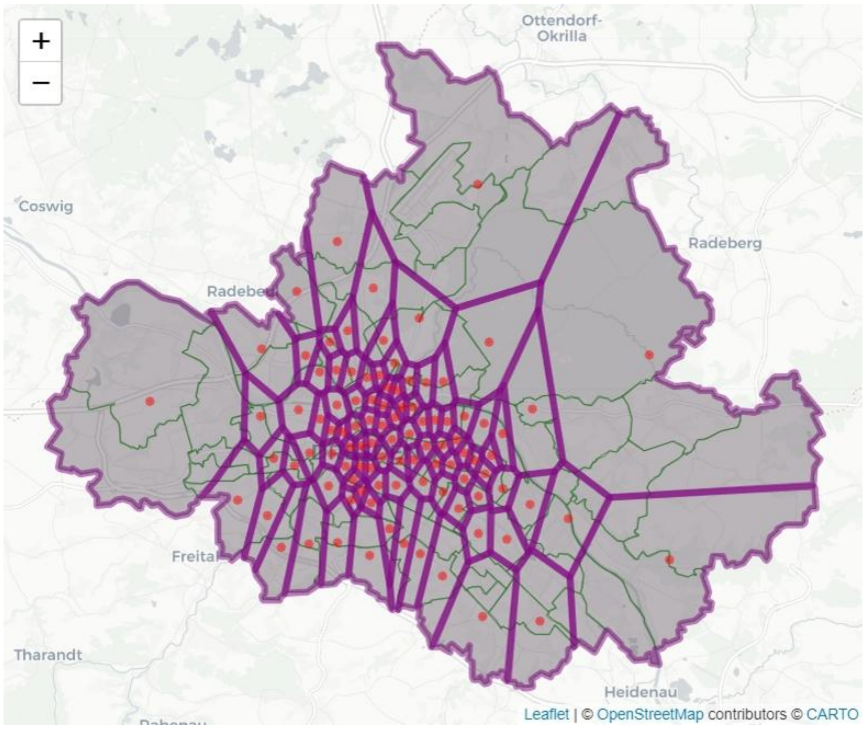

*Figure 4: Voronoi Diagram for Test Data*

<u>Chapter 2.3 Variables of Interest</u>

*2.3.1 Target Variable*

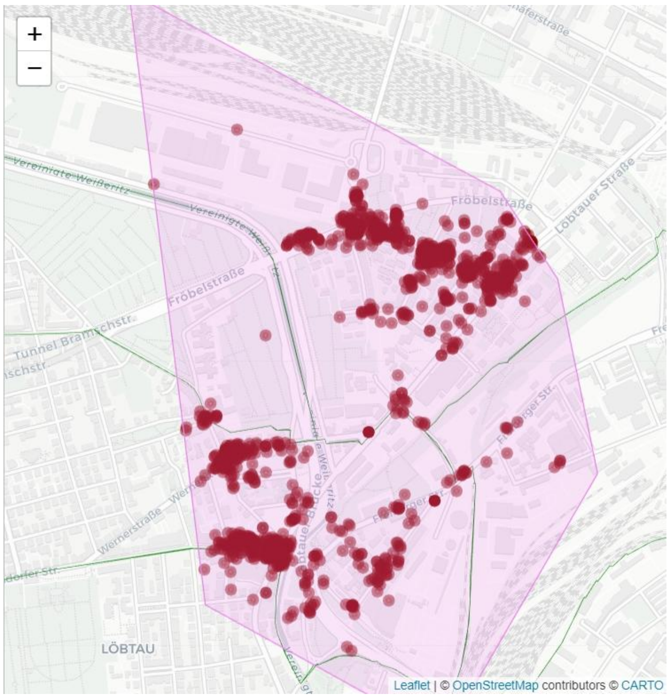

*Figure 5: Calculating Bike Demand*

The target variable for the models used in this project is meant to be demand for MOBIbikes. With the MOBIbike data that is provided to us, there is no direct measure or variable that can provide us with a tangible value for bike demand for different regions in Dresden. To have an approximation for bike demand, we use ‘bike availability’ as an instrument for ‘bike demand’. Bike availability has been determined by summing the number of GeoPoints (i.e. bikes) and assigning the summed number to the spatial centroid (i.e. observation), for each cluster respectively. This summed number is meant to represent the bike demand for the region enclosed in the cluster perimeter.

The format for this variable is consistent with clustering performed in Chapter 2.2. This aggregation approach has been carried out for both training and test data to attain the ‘bike_demand’ variable for different regions throughout Dresden. In each training and test dataset, 100 observations are attained for this variable since there are 100 clusters. Figure 5 visualises the GeoPoints (i.e. bike locations) aggregated for a cluster.

### *2.3.2 Predictor Variables - Data Source*

As our final number of clusters is 100, we are also considering using predictor variables within small regions like having the same number of clusters, to explain bike demand. Upon reviewing the OpenDataPortal Dresden website, we found that predictors measured across 64 semi-districts of Dresden matched pretty well with our requirements. From the OpenDataPortal Dresden website, we have compiled the following variables from the semi-districts for predicting bike demand: “population density”, “registered car density”, “density of young people (<60 years)”, “density of elderly people (>60 years)”, and “density of people receiving SGBII subsidies". Furthermore, because our data have a very crucial character: a spatial dimension. So, we also collected a geojson data with all the polygon data called from every semi-district in Dresden and named it “Disctrict_polygons.geojson ".

### *2.3.3 Predictor Variables - Format*

The predictor variables' formats are all in CSV data type. Those predictor variables showed the performance of variables in all the semi-regions of Dresden. For example, below is an example to the data frame of predictor "Population_Age":

*Table 3: Original data frame of predictor "Population_Age"*

| Stadt/Stadtteil Dresden | Total number | 0-5 year | 6 - 17 year | 18 - 24 year | 25 - 44 year | 45 - 59 year | > 60 year |
|---|---|---|---|---|---|---|---|
| 01 Innere Altstadt | 2714 | 138.414 | 130.272 | 344.678 | 1199.588 | 314.824 | 586.224 |

We can see, each semi region have its unique name and ID and the performance of population will be measured with 6 dummy variables “0-5 year”, “6-17 year”, “18-24 year”, “25-44 year”, “45-59 year”, “>60 year”. In this original data frame of population, we must pay attention to two points: 1. In Region “Dresdner Heide” and “Flughafen/Industriegebiet Klotzsche”, there are no predictors’ performance. But in the geojson data that contain the whole polygon data of semi-distract in Dresden. For later joining the polygon data to the predictor data frame and MOBIbike data frame, we will add two the row for those two regions and added NA value under all predictor variables. 2. For our desired predictor variables “density of young people (<60 years)”, we chose the columns “0-5 year”, “6-17 year”, “18-24 year”, “25-44 year”, “45-59 year” to present this variable. 3. The original data only provide the number of predictors, like population numbers in each semi-region. We believe with density can better predictor bike demands, so we will use the number of predictors divided by space of semi-regions to get our wanted predictors.

### *2.3.4 Preparation of the Predictor Variables*

In this section, our purpose is to combine all the individual predictors' data frames into one data frame. We begin by splitting like” Stadt/Stadtteil Dresden” into two separate columns “ID” and “Stadtteil”.  Following this, we used the function left_join, targeting the column with district name, to effectively merge all the predictor data frame and geojson data frame together. In conclusion, our predictor variable data frame will provide comprehensive insights. It encompasses not only the predictor information for each semi-district, but also the polygon data corresponding to each semi-region.

*1.*How to calculate cluster information from district information?

After preparing the target variable and predictor variables, we have right now two data frames with different polygon data.  As we can see from Figures 2, 3 and 4, the cluster normally crosses over a few semi-districts.

To determine the extent of overlapping areas formed by the semi-districts and their respective percentages. Our approach involved the following procedures:

- We used the st_intersection function to get intersections between polygons of the cluster and semi-district, resulting in overlapping regions.
- The st_area function was applied to calculate the enclosed area of both the overlapping region and the cluster area.
- The final percentage contribution was derived by comparing the overlapping area to the cluster area.

What we would like to emphasise regarding this approach is that we employed the above procedures with the functions st_intersection and st_area in a loop. Our aim was to compare the target polygon to each semi-district. Through this method, we obtained the overlapping area within each semi-district's polygon and calculated the corresponding percentage.

An illustrative example of a cluster can be seen as below:

*Table 4: Clusters Comprising Semi-District Areas*

| **Cluster ID** | **District ID** | **District name** | **% share in cluster** |
|---|---|---|---|
| 18 | 5 | Friedrichstadt | 15.27 |

| | | | |
|---|---|---|---|
| 18 | 25 | Pieschen-Nord/Trachenberge | 2.80 |
| 18 | 21 | Pieschen-Süd | 81.93 |

Figure 6 provides us with a visualisation for this cluster.

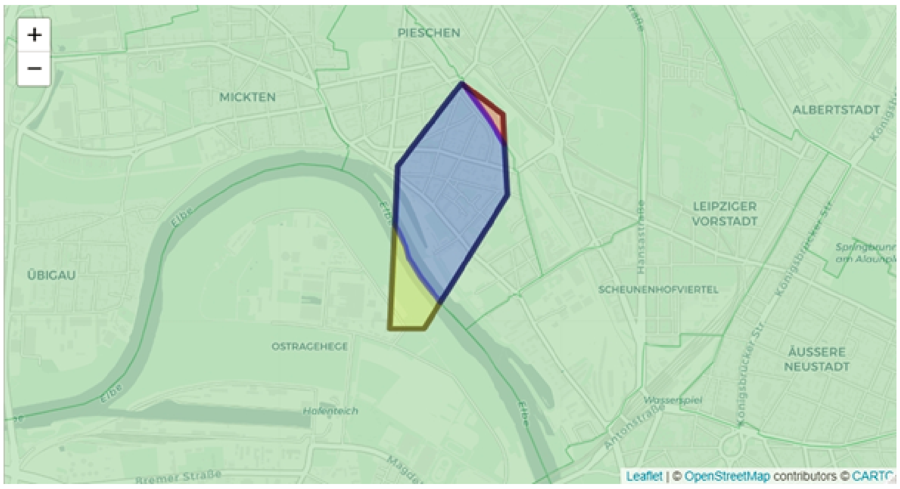

*Figure 6: Visualisation of a cluster composed of three portions*

*2.3.5 Calculate cluster predictor value by using Area-Weighted Average*

After applying the approach outlined above, we obtained the share of semi-district shares within each cluster. Since our predictor variables are measured within individual semi-districts, it became necessary to calculate the specific values of all predictors within our clusters. To obtain predictor values for each cluster, we chose to use the principle of area-weighted average as our measurement approach.

The concept of area-weighted average involves calculating the average value of a variable while considering the relative sizes of different regions. In our case, for example, Cluster 18 comprises 15.27% of Friedrich Stadt, 2.80% of Pieschen-Nord/Trachenberge, and 81.93% of Pieschen-Süd. Therefore, the weights of these three share parts are 15.27%, 2.80%, and 81.93%, respectively. The "weighted_population_density" of these three portions are calculated by multiplying the weight by the population density of each semi-district.

In the end, we derived the final value for each cluster by using the "group_by" function to aggregate the weighted predictor values, such as weighted population density, in the same cluster.

## Chapter 2.4: Bike Prediction Model

In order to determine the types of relationships that exist between bike demand in each cluster and our independent variables, we are considering the most suitable analytical approach. It appears that a regression task is more appropriate for our objective compared to classification or clustering methods, as it can effectively demonstrate these relationships. As a result, we intend to construct regression tasks to represent these associations. In the realm of machine learning, a variety of regression algorithms are available, such as bagging, random forest, boosting, and stacking. Among these, random forest and boosting methods have demonstrated favourable performance in numerous scenarios, leading us to opt for their utilisation in our project. Additionally we also aim to use spatial attributes in our data, by making use of Geographic Weighted Regression (GWR) modelling.

### *2.4.1 Random Forest*

From Figure 7, we can observe that as the number of nodes in the trees increases, the error associated with the training model starts to decrease. Once the number of nodes in the trees reaches around 120, the error of the training model stabilises. This graphical representation aids in understanding the relationship between the number of nodes in the trees and the model's performance in the context of random forest. It allows us to determine the optimal size of the node count in the trees that would result in optimal performance.

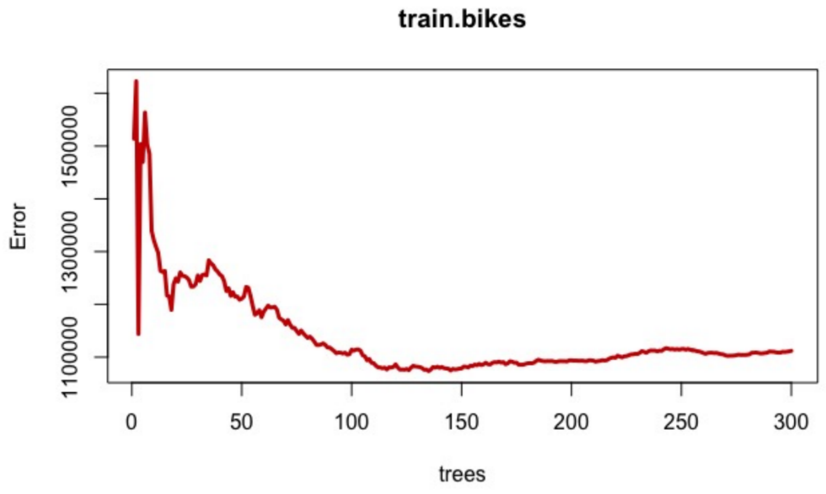

*Figure 7: Error on Train Model on Random Forest*

The Random Forest algorithm builds upon the principles of decision trees and addresses the issue of overfitting often encountered in decision trees. It is a versatile algorithm used for both classification and regression tasks in the field of machine learning. The fundamental concept of Random Forest involves not only selecting a random subset of observations from the entire dataset but also choosing a random subset of variables from the total set of variables. This randomness is introduced through techniques of bootstrap sampling, which helps create multiple random decision trees. The performance of a random forest regression model is typically assessed based on evaluation metrics such as RMSE (Root Mean Squared Error) or MAE (Mean Absolute Error), both of which are derived from the concept of MSE (Mean Squared Error). A lower value of these error metrics indicates a better-performing model. By using these error metrics, we can determine the effectiveness of a random forest regression model.

(1) $$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y_i})^2$$

(2) $$\text{RMSE} = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y_i})^2}$$

(3) $$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \widehat{y_i}\right|$$

where n is the number of points,$\widehat{y_i}$ is predicted value for the i-th data points, $y_i$ is the true value for the i-th data points.

There are two ways to understand the relationship between variables and the target variable: importance plots and partial dependence plots. The importance of a variable can be measured by the increase in MSE. Permuting one variable while keeping all other variables unchanged, and using other variables in other dataset to replace this variable, allows us to calculate the increased error in MSE. If this error is significantly large, it indicates that the variable is very important for our model.

(4) $$\Delta MSE = MSE_t(x_1,...,x_d) - MSE_d(x_1,...,x_{j-1},...,\widehat{x_j},\ x_{j+1},...,x_d)$$

From Figure 8, in the left picture, we can observe that population density has the highest importance, followed by car density, density of young-aged group people, and density of old-aged people. The variable with the lowest importance is the density of people receiving SGBII subsidy. In the right increased Node Purity picture, when we permute the population density variable while keeping other variables unchanged, it leads to the highest increase in node purity for the model. However, based on this criterion, car density becomes the third most important variable, with the second one being the density of old-aged group people. The last two variables contribute less to improving the model's node purity.

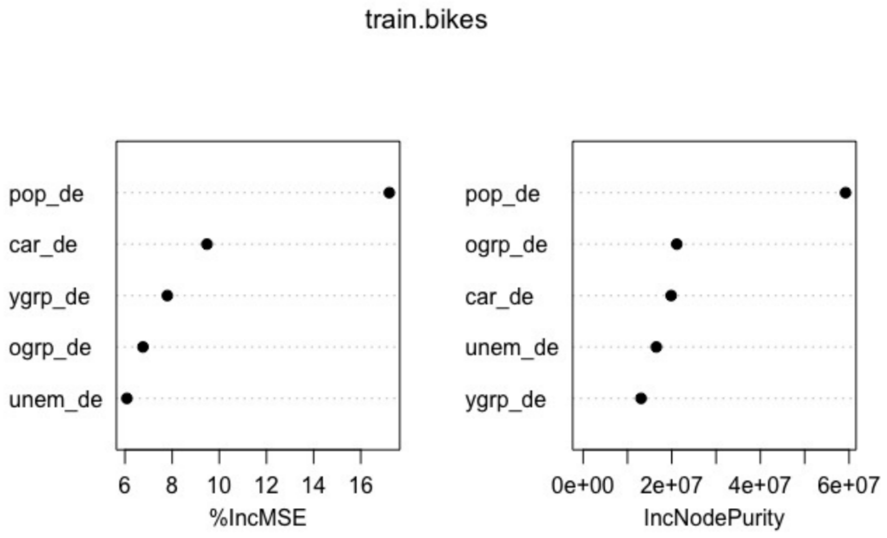

*Figure 8: Importance Plots on Training Model*

Partial dependence plots visualise the marginal impact of a variable on a model's prediction. To create a partial dependence plot, one variable is varied while keeping the other variables fixed at a certain value (usually their average or a specific value), allowing us to understand the relationship between that variable and the target variable.

(5) $\widehat{f_j}(x) = \frac{1}{n}\sum_{i=1}^{n}\widehat{f}(x_{1,i},..., x_{j-1,i},..., x,\ x_{j+1,i},..., x_{d,i})$

From Figure 9, in the first picture, the impact of population density on the aggregated number of bikes is increased. As population density increases from 0 to around 14000, there is a significant increase in the aggregated number of bikes. This impact becomes even more pronounced between 14000 and 18000, followed by a sharp drop up to 20000 and a gradual increase thereafter. Overall, this relationship exhibits a complex pattern that is not purely linear. Similarly, variables like young-aged people density, old-aged people density, and car density also show similar tendencies with fluctuations. Notably, young-aged people density demonstrates distinct fluctuations, indicating a more intricate relationship. Young-aged people density is positively related to the aggregated number of bikes, contrary to our initial assumption that a high density of young people would correlate with higher bike usage. On the other hand, old-aged people density's relationship is less intuitive, as it also follows a similar fluctuating trend. The relationship between car density and bike demand contradicts our assumption that areas with more cars would have reduced bike demand. The partial dependence plot for car density does not clearly indicate such a relationship. Conversely, the density of people receiving subsidies displays a negative relationship with bike demand. This implies that areas with a higher density of economically disadvantaged individuals have reduced demand for bikes. One possible explanation is that individuals in these areas prioritise basic necessities over bike uses, even when they receive financial support.

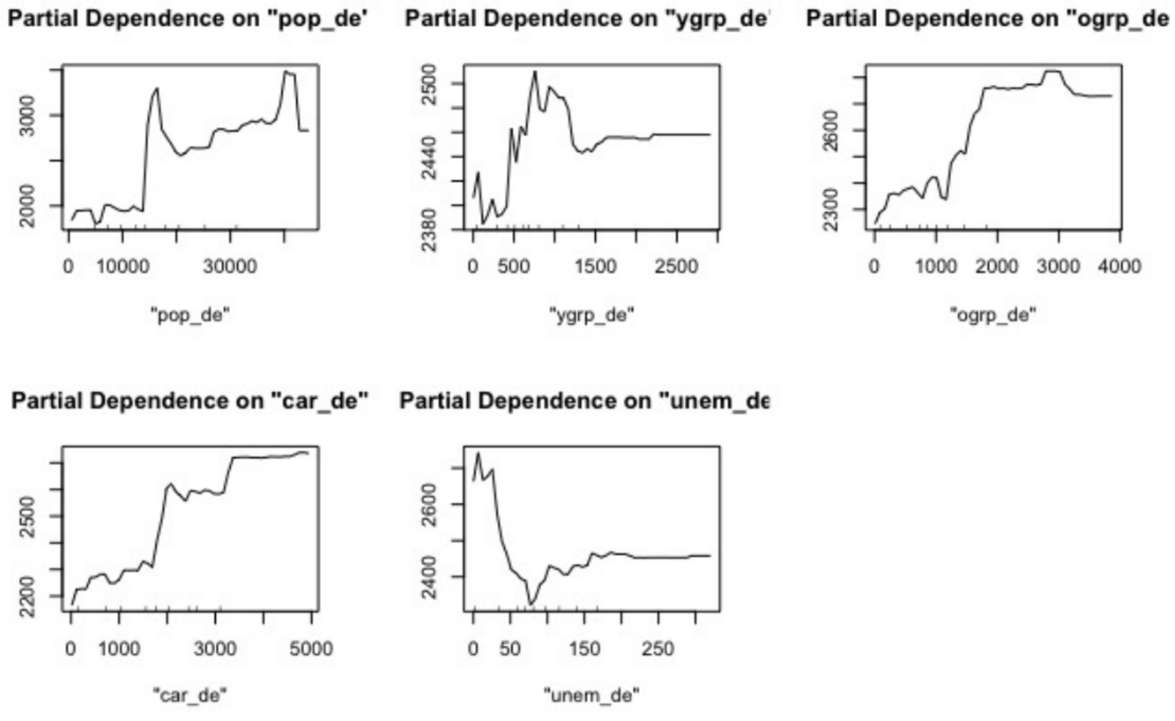

*Figure 9: Partial Dependence Plots on Training Data*

To assess the goodness of fit of our model on the training data, we calculated the R-squared (R2) value, which is 0.84. This indicates that our model demonstrates a strong fit to the training data, capturing a significant portion of the variance in the dependent variable. And to evaluate the performance of our random forest model, we calculated the Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE), resulting in Table 3:

From the table, it is evident that both the Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) values are higher when compared with the mean and median of the actual bike demand. This indicates that our random forest model is not performing so well.

*Table 5: Performance on Random Forest*

| RMSE | MAE | Mean | Median |
|---|---|---|---|
| 1522 | 1343 | 1199 | 1154 |

### *2.4.2 Boosting*

Boosting is a powerful method for ensemble learning, a set of classifiers are combined to make a better work for any classifier and weak ones that are easy to learn are usually chosen. A main idea of boosting is to use initial weights to obtain a weaker classifier on the training data. Then, based on the learning error rate of the weak classifier, update the weights for the training data. This involves assigning higher weights to the points that are incorrectly classified by the weak classifier. This increased weighting aims to focus more on the data points with a high learning error rate, effectively giving them more attention. Subsequently, these updated weights are used to train another weak classifier. The process is repeated iteratively, until the number of weak classifiers reaches a predefined number. The final classifier is a combination of weak classifiers. The boosted version of the same classifier almost always produces better results, boosting a bad classifier is often better than not boosting a good one, finally, boosting a good classifier is often better, can take more time. A common evaluation approach for boosting involves calculating metrics such as Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) to assess its predictive accuracy and generalisation ability. A lower value of these error metrics indicates a better-performing model. By using these error metrics, we can determine the effectiveness of a boosting model.

R2 on the training model in boosting is 0.75, indicating that the boosting training model also exhibits a strong fit, similar to the performance observed in the random forest training model.

*Table 6: Performance on Boosting*

| RMSE | MAE | Mean | Median |
|---|---|---|---|
| 1520 | 1335 | 1199 | 1154 |

From the table, it is apparent that the RMSE and MAE values obtained from the boosting model are higher when compared with the mean and median of the actual bike demand values. This suggests that the model's performance on boosting might also not be very satisfactory. However, it is important to note that the overall performance on boosting seems to be better than the performance achieved with random forest.

There are several reasons that could contribute to this result. One significant factor is the size of our dataset, which is large enough to pose computational challenges. As a result, we had to randomly select subsets of the data for training and testing purposes, as previously mentioned. This random sampling process can significantly impact the model's performance. Given these considerations, we have decided to compute the GWR (Geographically Weighted Regression) model again. This will allow us to make a comprehensive comparison between the three models, in order to improve the overall fit of the model.

### *2.4.3 Geographically Weighted Regression - Accommodating spatial variation*

The simple linear model used for GWR is:

$$\begin{aligned}\text{Bike Demand} = {} & \beta_0 + \beta_1 \text{ (Population Density)} \\ & + \beta_2 \text{ (Car Density)} \\ & + \beta_3 \text{ (Density of young people (< 60 years))} \\ & + \beta_4 \text{ (Density of old people (> 60 years))} \\ & + \beta_5 \text{ (Density of people receiving SGBII subsidy)}\end{aligned}$$

When a simple linear OLS model is used for regression analysis, the relationships that are captured between the dependent and explanatory variables are global in nature. This means that the relationships between these two variables are identified by aggregating the information throughout Dresden. While this can help us understand how a change in one explanatory variable can affect bike demand on average, this approach suffers with the over-aggregation problem. This is because local patterns in relationships are missed out. For example, the relationship between bike demand and car density might be positive in one region in Leuben, Dresden. However, this relationship can also be negative in a region in Neustadt, Dresden. GWR provides us an opportunity to capture these local relationships and provide us with a deeper insight. If there appears to be no relationship between bike demand and car density from the generalised OLS model, but the relationship comes out to be positive for urban areas in Dresden and negative for rural areas in Dresden, then the generalised result is misleading. Hence, it is worthwhile to give weight to local relationships.

So, how does GWR manage to capture these localised patterns? A simple OLS model can be represented with the expression: $Y = X\beta + \varepsilon$, where the estimated coefficient is $\beta_{est} = (X^TX)^{-1}X^TY$. In this calculation, neither the variable nor the coefficient has any spatial dimension, and the estimated coefficient is static. With GWR, a spatial dimension is introduced and the linear model is altered with the expression: $Y = (\beta \otimes X).1 + \varepsilon$, where '$\otimes$' represents vector-product between the coefficient vector and the explanatory variables vector. The estimated coefficient now becomes dynamic:

$$\beta_{est} = \begin{bmatrix} \beta_0(u_1, v_1) \ldots \beta_p(u_1, v_1) \\ \beta_0(u_j, v_j) \ldots \beta_p(u_j, v_j) \\ \beta_0(u_n, v_n) \ldots \beta_p(u_n, v_n) \end{bmatrix}$$

The estimated coefficient now holds a spatial dimension i.e. coordinates (u,v) for the respective regression points/observations. In simple words, the estimated coefficient for each variable is varying with variation in space (i.e. coordinates).

For the clustering approach explained in chapter 2.2, it could be argued that instead of choosing clusters this project could have made use of Dresden districts to determine the values of predictor and target variables. While this can also be done, Fotheringham et. al. (2002) assert that this approach is naive. According to their assessment, considering boroughs/districts poses the same problem that arises when we consider aggregate over the entire area of Dresden. This problem is missing out on variations in local relationship within a borough/district, since uniform local relationship throughout the entire district cannot be assumed. In the view of Fotheringham et. al. (2002) are only administrative boundaries and discontinuities in the occurrence of spatial variation may not happen at these boundaries. To cater to this concern, this project uses clustering so that Dresden is randomly divided into chunks that are much smaller in size compared to boroughs/districts. This is done with the intention to capture more local variation in the data.

### *2.4.4 GWR – Calculating Estimated Coefficient*

The simple OLS model arrives at the estimated coefficient by giving equal weights to all the data points in the data. This cannot be done in GWR, since we are not estimating the coefficient by analysing the average partial effect over all the data points. We want to capture the local partial effect, over location specific data points. The approach that GWR follows to achieve this can be understood by observing Figure 10. This figure represents the kernel function used in GWR. For any regression point, the data points that are closer to the regression point are given higher weightage while the data points that are further away from the regression point are given less weightage. The kernel function is responsible for assigning weightage based on distance from the regression point. The weights calculated from this function are then stored in a weight matrix ‘W’. The width of the curve created by the kernel function is set by the ‘bandwidth’ parameter of the function, which varies with distance. With this approach, the localised partial effect gets captured since data points that are away from the regression point get discounted based on bandwidth. Utilising the created weight matrix, the GWR estimate of the coefficient gets contrasted from the standard OLS estimate and comes out to be: $\beta_{(ui,\ vi)} = (X^T W_{(ui,vi)} X)^{-1} X^T W_{(ui,vi)} Y$.

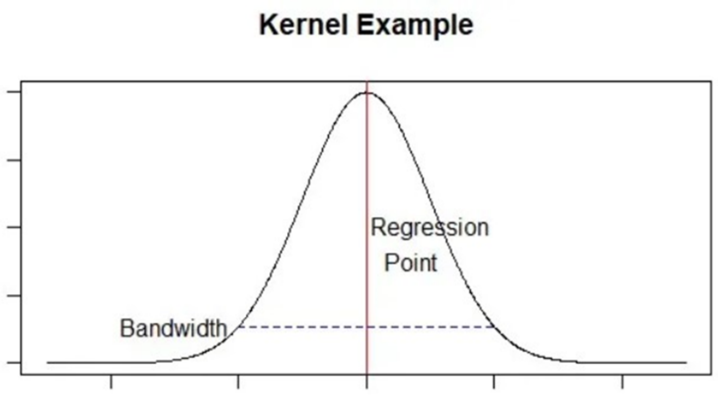

*Figure 10: GWR Kernel*

*2.4.5 GWR - Fixed versus Adaptive Kernel*

The kernel function can use the same weight matrix for all regression points in the data. Thus for every regression point the weight of other data points decreases at the same rate as distance increases, independent of the number of data points in the neighbourhood of the regression point. This approach is known as a 'fixed kernel' and can be interpreted as having a fixed 'bandwidth'. However, there can be instances where the number of neighbouring data points may not be uniform, as shown in Figure 11. For this project, the number of neighbouring data points are high when regression points are located in the centre of Dresden compared to the outskirts of Dresden. Additionally, the fixed kernel approach is slightly rigid because it holds an implicit assumption that spatial variation in the data is constant (fixed bandwidth condition). This may/may not be the case in reality. As a remedy, a more flexible method known as 'adaptive' kernel is used in the project. Adaptive kernel is more flexible because it deploys variable bandwidth instead of a fixed one, for each regression point. This allows it to have kernels with larger bandwidths in areas where neighbouring data points are less in number and vice-versa. Another advantage is that the robustness of results increases because it allows for spatial heterogeneity in the data (the assumption of constant spatial variation is not made).

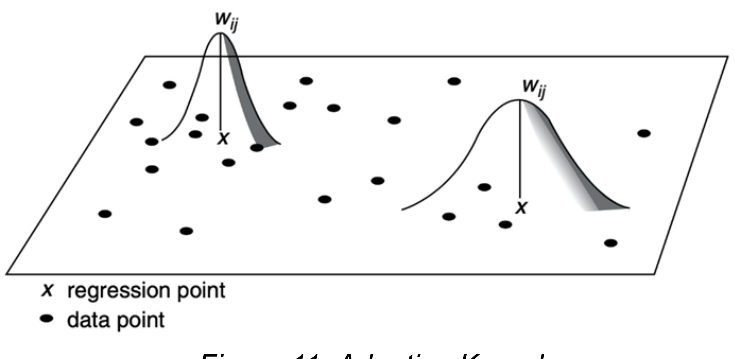

*Figure 11: Adaptive Kernel*

### *2.4.6 GWR - Results (Training Data)*

For the global GWR model, the $R^2$ comes out to be 0.5578367. This suggests that more than 55% of the variation in the data can be explained by the GWR model. However, the fit of the model can be observed more closely by observing the $R^2$ values of the local models in Figure 12.

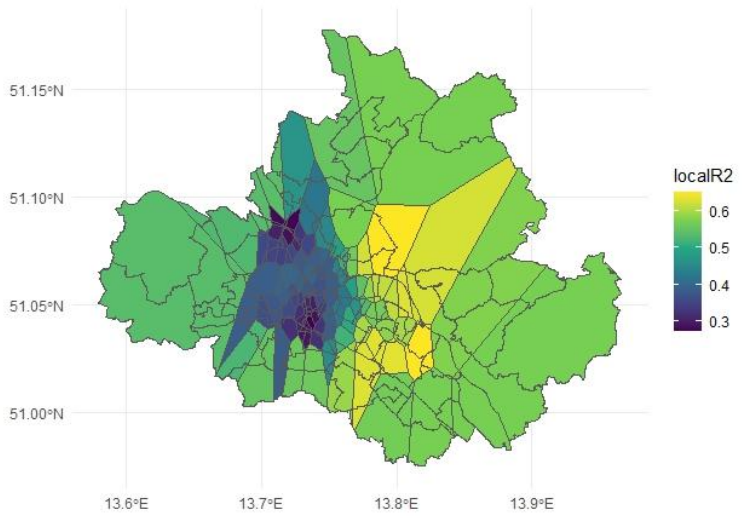

*Figure 12: GWR - Local* $R^2$

The $R^2$ value of local models for the majority of areas throughout Dresden appears to be greater 0.52 for regions coloured in green. In central-eastern regions, the $R^2$ value appears to even be greater than 0.6, for regions coloured in yellow. This shows that local models built using GWR actually have a high goodness of fit. The issue appears to be in some clusters in the central-western region, where-in the $R^2$ values are reaching as low as 0.2 (regions shaded in blue). For these areas, the local models are unreliable since the goodness of fit is poor. It is due to these values that the $R^2$ in the global model is being dragged-down to 0.5578367. The estimated coefficients from the global model are shown in the table below:

*Table 7: Estimated Coefficients in GWR global model*

| **Variable** | **Estimated Coefficient** |
|---|---|
| Intercept | 1387.4983 |
| Population Density | 0.0425 |
| Car Density | 0.1904 |
| Density of young people (<60 years) | 0.4164 |
| Density of old people (>60 years) | 0.1614 |
| Density of people receiving SGBII subsidy | -5.4502 |

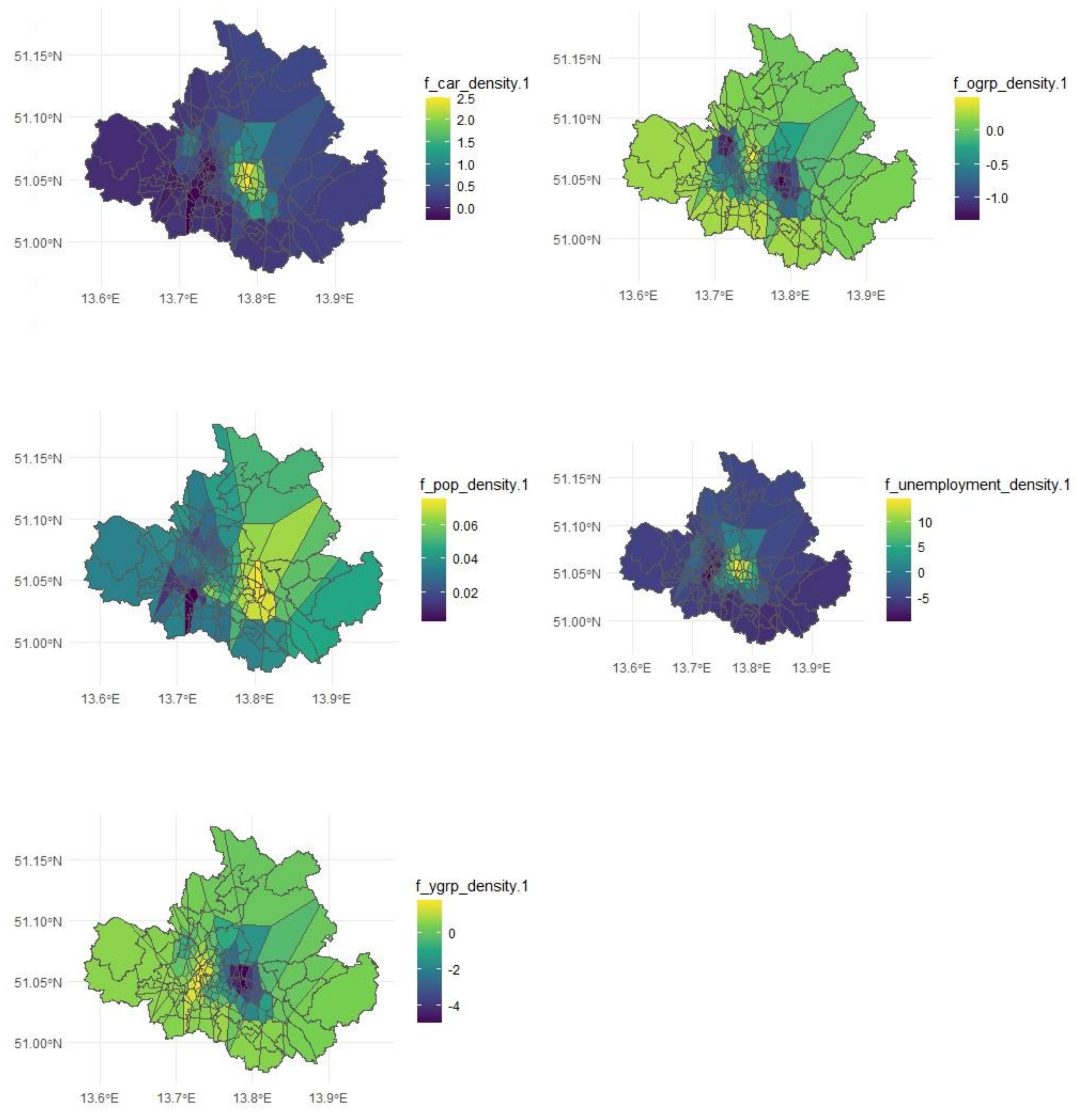

*Figure 13: Spatial Coefficients - GWR*

The estimated coefficients from the global model gives us an overview of the relationship between bike demand and the predictor variables but doesn't really tell us anything about local relationships that are varying with space. Figure 13 helps visualise the estimated coefficients for each variable, varying with space.

If we observe the 'Density of young people (<60 years)' variable i.e. 'f_ygrp_density' variable, we find that the relationship of this variable with the target variable is positive (yellow regions), negative (blue regions) and neutral (light green regions) for different regions throughout Dresden. This is in stark contrast with Random Forest learner which appears to show only a somewhat positive relationship (Figure 9). The advantage of GWR is in its ability to capture changes in the relationship between target and predictor variables at a local level which makes it better at learning structural relationships at a deeper level.

*2.4.7 GWR – Prediction*

Amongst the list of available learners in 'mlr3' and 'mlr3learners', there is no learner that is dedicated to apply GWR regression. To make prediction on the test observations using the training model, we tried to develop a learner with the 'create_learner' function by installing the 'mlr3extralearners' package from GitHub. Using the 'spgwr' package we tried to develop a 'gwr' learner but unfortunately it didn't work. Hence, we tried to make predictions manually. For the clusters available in the test data, we determined the target and the predictor variables as shown in Chapter 2.3. The variable values were then assigned to the cluster centroids, as the centroids behave as regression points/observations. To make use of the same training model formulated by GWR we are supposed to assign the calculated variable coefficient (during training) to regression points in the test data, based on location. This was resolved by identifying the cluster (in the training data) that the regression point (in the test data) belongs to. Since every Geo-Point in a training cluster is supposed to have the same value for variable coefficient as the centroid of the training cluster, the regression point in the test data is also assigned the same value.

After identifying the spatial coefficients for test observations, predicted values of the target variable were compared with actual test values. For determining the prediction quality, the RMSE came out to be 1519.55 and the MAE came out to be 1336.458.

## 3 Conclusion

Based on the three learners we employed—random forest, boosting, and geographically weighted regression—our analysis reveal the following performance metrics: For random forest, the Root Mean Squared Error (RMSE) was found to be 1522, while the Mean Absolute Error (MAE) was calculated to be 1343. Moving to the boosting model, we obtained an RMSE value of 1520, alongside a corresponding MAE value of 1335. Lastly, with GWR, we observed an RMSE of 1520 and a MAE of 1336. Across these three models, our findings indicate remarkable similarity in their prediction performance. This suggests that for this project, no learner is clearly superior for predicting the target variable. If the motivation is to learn structural relationships between the target and the predictor variables, GWR is far superior. This is because the coefficients provided are easy to interpret and the relationships determined are not over-generalised as they are captured at a local level. Coming to the goodness of fit—the $R^2$ value for the training model using random forest is 0.84, for boosting is 0.75, and for GWR (global model) is 0.55. From these values it appears that variability in the dependent variable can be better explained by independent variables using random forest and boosting. However, as discussed in section 2.4.6, if the blue regions in Figure 12 are excluded GWR also provides a reasonably high goodness of fit for a majority of regions throughout Dresden, with some regions even exceeding an $R^2$ value of 0.6.