Clustering Ward-Level Poverty Using Satellite Imagery

Presented by:

Andomei Smit
SMTAND051

Ryan S. Anderson
ANDRYA005

Supervisors:

Dr Şebnem Er

Mr Joshua Arendse

Submitted to the Department of Statistical Sciences at the University of Cape Town in partial fulfilment of the academic requirements for a Bachelor of Science Honours in Statistical Sciences

November 25, 2020

Keywords: Poverty, Satellite Imagery, K-means, HDBSCAN

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

Signature: .....
A. Smit

Signature: .....
R.S. Anderson

Date: November 25, 2020

Acknowledgements

Dear Şebnem and Josh

Ryan and I would like to say
on this sunny and beautiful day
this year has been a blast
and at last:
here we are at our final draft!

We have learnt so much
about statistics and such
but what made it worth it all
is having supervisors that care a whole lot
about all our concerns, no matter how small.

Abstract

South Africa is one of the most unequal countries in the world. It is crucial for key decision- and policymakers to have up-to-date and accurate information regarding the distribution of poverty, in order to use scarce resources effectively to target interventions to specific areas. However, the traditional methods for acquiring this information has severe limitations. Thus, this paper seeks to use an alternative to the traditional techniques for data collection - the use of publicly-available satellite imagery. Furthermore, machine-learning techniques are applied to the imagery in an attempt to create a ward-level poverty-map for Gauteng, without the use of survey data.

To do this, a pre-trained convolutional neural network (CNN) was used to extract features from the satellite imagery that are associated with nightlight intensity. As nightlight intensity is assumed to be a proxy for poverty, the extracted features are also expected to be predictive of poverty. The wards are then clustered using K-means and HDBSCAN based off the principal components of these features to create the before-mentioned poverty-map.

The results showed that the clusters performed poorly when distinguishing the wards based on their level of poverty. Instead, the clusters seemed far more effective at identifying wards with similar nightlight intensity. This suggests that a regression-based method is potentially required to map the nightlights-related features to poverty estimates.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background to the Study

This paper can be seen as the intersection of one of the greatest inhibitors of economic development across the globe - poverty - and arguably the most enabling advancement of recent times - machine learning. Thus, this study aims to develop a method to help address the unacceptably high levels of poverty present ain South Africa, with a focus on Gauteng, by leveraging machine learning and publicly-available data.

1.1.1 Poverty in South Africa

South Africa is one of the most unequal countries in the world. Inequality was already high in the 1990's, largely due to Apartheid where a major part of the population was prohibited from pursuing economic opportunities [9]. The country's GINI-index[1] inflated further during the 2000's and has remained high relative to other emerging markets [9].

Many private and government-driven efforts have been made to redistribute resources to aid those who are most impoverished. In order to use scarce resources effectively to target interventions to specific areas, it is crucial for key decision- and policymakers to have up-to-date and accurate information regarding the distribution of poverty.

Historically, household data, collected through surveys, has been the main tool to acquire this information. Usually, there are 5 domains of deprivation that these surveys focus on, namely:

1. Income and material deprivation
2. Employment deprivation
3. Health deprivation
4. Education deprivation
5. Living environment deprivation

[1]The Gini-index is a score of how unequal a country is with respect to income. A score close to 1 relates to a perfect inequality, for example where a very small fraction of the entire population earns all the income in the country.

These domains are then combined to create a poverty index. An example of such an index is the South African Multidimensional Poverty Index [10] (SAMPI).

However, using surveys for data collection and processing is often time consuming and expensive, leading to a large time lag between when surveys are conducted and decision-makers can use the information.

Thus, three main problems with the use of survey data exist:

1. Time consuming and expensive data collection processes
2. Prior ethical clearance requirements to access survey data
3. Data processing to create a usable poverty index further delays access to relevant information

This paper seeks to use an alternative to the traditional techniques for data collection- the use of publicly-available satellite imagery. These can potentially be used to map poverty in Gauteng, South Africa.

1.1.2 Related Work: Poverty Estimation

The use of satellite imagery for poverty estimation is a topic of growing interest as high-resolution satellite imagery is increasingly available at the global scale. A popular approach, particularly for developing countries, has been to estimate economic activity by using the light intensity of nighttime satellite imagery (also known as "nightlights") [11, 12]. These studies found that the use of nightlights to improve estimates of economic activity is subject to high measurement error and, as a result, is most likely to only add significant value for countries with low-quality statistical systems[2] [11, 12].

However, this is not to say that the models performed better for these countries, but that, in the absence of alternative economic indicators, the estimates were "better than nothing". In fact, these techniques are less capable of distinguishing differences in economic activity in areas with impoverished populations as luminosity levels are generally very low and show little variation [1, 12].

Jean et al. [1] used an indirect approach to incorporate the noisy but informative nightlights data to predict well-being. Rather than using the nightlights for their predictive

[2]For example, countries with no recent population or economic censuses.

power - which has been shown to be limited [11, 12] - they used it to fine-tune a convolutional neural network (CNN) as a feature-extractor. The "learned" features are then used to predict (i) average household consumption expenditure and (ii) asset wealth at the cluster level (roughly equivalent to villages in rural areas or wards in urban areas) in five African countries [1].

Due to the "data dependence" (the need for vast amounts of training data) of CNNs, this involved deploying a "transfer learning" approach - a method of "transferring" the knowledge gained from a base task to a target task, reducing the training required on the target data set and task [1, 13].

This transfer learning approach involved two main steps. Firstly, the CNN was pre-trained on ImageNet (a large visual database designed for use in visual object recognition [14]) in order for it to "learn" to identify low-level features, such as edges and corners [1]. Then, using this base "knowledge", the CNN is fine-tuned on a new task - training it to predict nightlight intensity from input daytime satellite imagery.

During the training, the model "learns" to summarize the input daytime satellite images as a lower-dimensional set of image features that are predictive of the variation in nightlights [1]. Thus, the nightlights are used to train the CNN as a feature-extractor - "learning" to identify features from daytime satellite images that are correlated with economic well-being [1]. They showed that the model was able to identify some "livelihood-relevant" features in the images such as urban areas, roads, bodies of water and agricultural areas [1].

These features are then aggregated to the cluster level and are used to train ridge regression models to estimate cluster-level expenditure and assets. Referring to the diagonals in Figure 1.1, they found that the ridge regression models were able to explain 41 to 56% of the variation in average household expenditure and 55 to 75% of variation in average household asset wealth across the African countries investigated [1]. Their model was, on average, substantially more predictive of variation in consumption and assets than nightlights alone - despite being trained partially on nightlights. Furthermore, their models "travel well" across borders, with out-of-country predictions often approaching the accuracy of in-country predictions (as seen by the off-diagonal values in Figure 1.1) [1].

Interestingly, Piaggesi et al. [15], adopted a similar technique to Jean et al. [1] but directly used a variety of CNNs pre-trained on ImageNet as their feature extractors - they made no use of the nightlights data. Despite this lack of "fine-tuning", they too were able to explain a "significant fraction" of the variation in household income across urban cities of developed countries [15]. Furthermore, their models were found to also

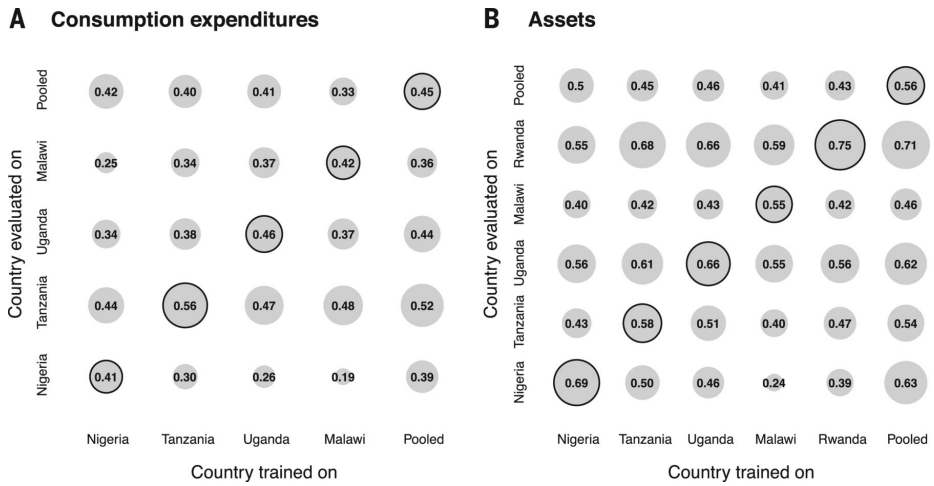

Figure 1.1: Results taken from Jean et al. [1]. Cross-validated r^2 values for consumption and asset predictions for models trained in one country and applied in other countries. Countries on the x axis indicate where the model was trained, countries on the y axis indicate where the model was evaluated. The reported r^2 values are averaged over 100 folds (10 trials, 10 folds each) [1].

"travel well" across cities [15].

Although the results of Piaggesi et al. [15] are promising, their model was trained and validated using US cities which may differ significantly compared to South Africa with regards to urban landscape and the manifestation of poverty. On the other hand, the Jean et al. [1] model was trained on clusters in Nigeria, Tanzania, Uganda, Malawi and Rwanda - countries that are far more alike to South Africa. Thus, it seems that the results of Jean et al. [1] are more applicable to this study.

With the success of the methodology used by Jean et al. [1] in mind, and the ability of their models to "travel well", the objectives of the study can be defined.

1.2 Objectives of this Study

The aim of this research is to cluster Gauteng wards based on their level of poverty and, as a result, create a poverty map. Furthermore, the clustering will purely be based on publicly available data - assessing the potential for census-less poverty mapping. Since high-resolution satellite imagery is publicly-available, the wards will be clustered from the information that can be extracted from this imagery.

The absence of up-to-date poverty data in South Africa makes fitting a regression model to predict ward-level poverty (like Jean et al. [1]) problematic, since the satellite images are current. Thus, the methodology of Jean et al. will be revised as follows:

Step 1: Obtain a set of satellite images for each of the Gauteng wards

Step 2: Extract a feature set for the wards from the satellite imagery using a pre-trained CNN[3]

Step 3: But then, rather than using ridge regression to map the features to a poverty estimate, the features will be clustered directly

To be consistent with Jean et al. [1], the satellite images are sourced from Google Static Maps API. They are 400x425 pixels in size, with a zoom level of 16. This corresponds to each image covering roughy 1km^2. An example of one of these images can be seen in Figure 1.2.

Figure 1.2: An example of the type of satellite image that is going to be used from Google Static Maps API. The parameter configurations are 400x425 pixels in size and a zoom level of 16.

To clarify what the Gauteng wards are, refer to the map of Gauteng in Figure 1.3. The coloured regions are the Gauteng municipalities, while the polygons within these municipalities are the wards - there are 508 in Gauteng.

1.3 Scope and Limitations

The paper is going to focus on the theory behind and methodology of the clustering algorithms implemented. Conversely, there will be little explanation of the inner-workings of CNNs or the algorithm devised for extracting the satellite images for the wards[4].

As with all research, there are some limitations:

[3]The pre-trained CNN that will be used is a PyTorch replication [16] of the Caffe CNN developed by Jean et al. [1] (the replication is recommended by Jean et al. on their GitHub repository). The Caffe CNN was developed in Python 2 and has not been maintained by the authors, hence the need to make use of the replication.

[4]The image-extraction algorithm is outlined in Appendix A.

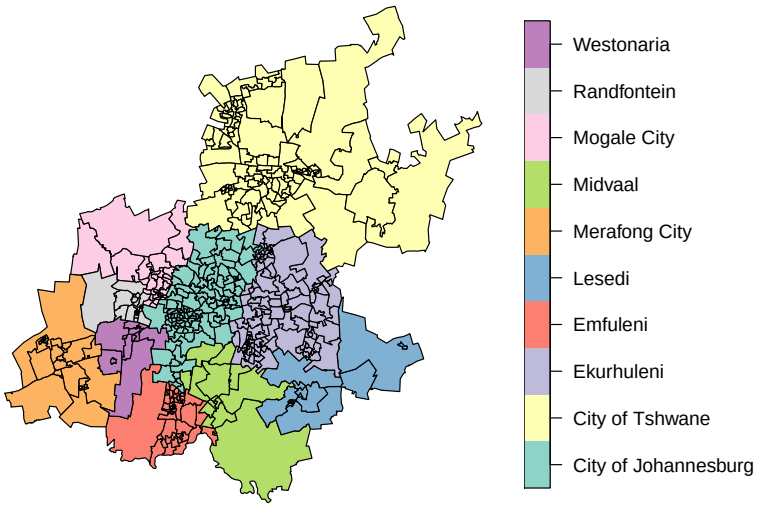

Figure 1.3: A map of the Gauteng wards coloured by municipality.

1. The use of a pre-trained CNN has removed any flexibility to modify the CNN architecture to better suit this use-case.

2. Some of the Google Static Maps images displayed inconsistencies (see Figure 1.4). It appears that some images have been constructed by stitching together images taken at different times of day and different time periods. This could lead to undesirable information being encoded into the feature vectors in Chapter 2.

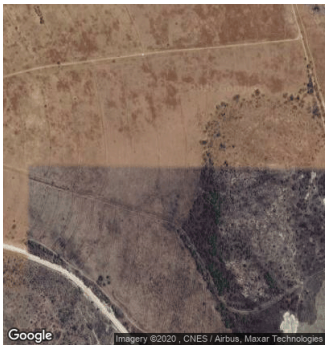

Figure 1.4: A satellite image demonstrating the stitching of images by Google Static Maps API.

1.4 Report Outline

As the methodology of this study is largely inspired by and based on Jean et al. [1], a similar, logical flow is naturally associated with this study. The use of the term "logical" follows from the fact that the structure replicates the model-building process.

In Chapter 2, the report gives an in-depth explanation of the feature-extraction process, with an emphasis on the modifications made to the pre-trained CNN in order to utilize it as a feature-extractor, and the proceeding use of the CNN to develop a feature vector for each of the Gauteng wards.

Due to the high-dimensionality of the resulting feature set (and the associated negative consequences on clustering algorithms), in Chapter 3, a dimension-reduction technique is implemented to find a low-dimensional approximation of the features.

In Chapter 4, based on the dimension-reduced features, the K-means and HDBSCAN algorithms are then used to examine the extent to which the Gauteng wards are clustered. These algorithms are thoroughly investigated: firstly, exploring the theoretical background; before outlining the specifics of their implementation. After which, various internal validation metrics are used to determine the strength of the resulting clusters. Lastly, poverty data is used to investigate whether the clusters are in fact related to poverty.

In Chapter 5, the results are consolidated and summarized, with possible explanations for the inability of the algorithms to effectively cluster the wards being noted.

Chapter 2

Feature Extraction

Before any clustering algorithms can be trained, a vector of features for each ward in Gauteng needs to be extracted. These features will be extracted from the satellite images that cover each ward. Thus, satellite images for each ward first need to be obtained before moving onto obtaining the features from the images.

2.1 Image Extraction

The details of the image-extraction algorithm used to obtain the necessary satellite images can be seen in Appendix A. For each ward, the output of the algorithm is a number of image-pairs comprised of (1) a plain satellite image and (2) an image mask, with the white region of the mask outlining the area of the satellite image that falls within the ward of interest. The number of image pairs is dependent on the number of images needed to cover the entire ward. An example of one of these image-pairs can be seen in Figure A.11.

Once these pairs of satellite images and masks had been obtained, the feature extraction process could begin.

2.2 Feature Extraction

This section aims to explore the methodology used to map each plain satellite image, with the aid of its corresponding mask, to a set of features.

As mentioned previously, the methods used for feature extraction and feature aggregation are inspired by Jean et al. [1]. However, the pre-trained CNN used is a PyTorch implementation [16] of the Caffe CNN developed by Jean et al. [1]. Furthermore, certain modifications are made to the pre-trained CNN to account for the irregular shape of the Gauteng wards.

2.2.1 Model Architecture

The base model used is a VGG 11 CNN. VGG is a widely used CNN architecture proposed by Simanyan and Zisserman in 2014 [17]. The model architecture for VGG *16* can be seen in Figure 2.1. Although the VGG 11 architecture and not the VGG 16 is being implemented, the only difference between the two architectures is the addition of an adaptive average pooling layer after the final max pooling layer and the inclusion of batch normalization. Thus, the visualization is still informative.

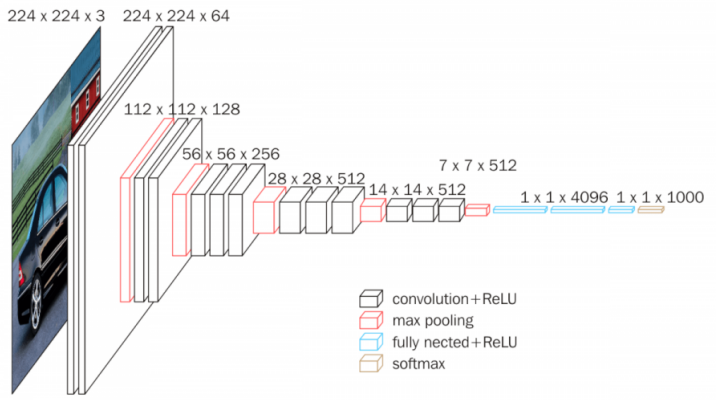

Figure 2.1: VGG 16 model architecture [2].

In most instances, CNNs are used for classification and regression tasks, however, they can also be used as feature extractors by removing all or some of the fully-connected layers. The latter is the motivation for using a CNN in this use-case. The model is used, through a "learned" non-linear mapping, to transform each input satellite image into a feature vector - encoding information relating to the low-level features present in each image (such as corners and edges) and features that are predictive of the variation in nightlights (such as urban areas, non-urban areas, water and roads) [1]. The logic is then that some of these same features may also be predictive of variations in poverty.

Before feeding the images into the CNN, the images are first resized to 224x224 pixels (to match the input dimensions expected by the VGG model), and then normalized to reduce the variation in the distribution of the data. Normalization as a preprocesssing tool is simple but has been proven to significantly improve CNN performance [18]. For a given colour channel, this involves computing two values from the pixels in that colour channel over all of the images in the training set: (1) the average pixel intensity and (2) the standard deviation. Then, for each image extracted, this channel-wise average is subtracted from each pixel, with the result being divided by the channel-wise standard deviation.

The actual channel-wise mean and standard deviation used in the implementation was

computed by Mather on their training set [16]. However, since the nature of the images is similar (daytime satellite imagery with the same resolution), there seems to be no reason why these values are not applicable to this study as well. Thus, they are left unchanged.

Unlike Jean et al. [1], the satellite images used have a particular region of interest (ROI) relating to the portion of the image inside the ward of interest. Referring back to Figure A.11, the ROI of a given satellite image is identified by the white region of its corresponding mask. The CNN needs to be modified to "ignore" all of the pixels of the images that fall outside the ROI as it would be undesirable to include information from outside a particular ward in the features for that ward.

2.2.2 Modifications to CNN

In order to introduce this modification, the VGG model needs to be broken down into its two core components: the feature extracting convolutional layers and the fully-connected classifier. In between these two components, the masks will be utilized to isolate the ROI. This modification has been visualised in Figure 2.2. The motivation for this modification is inspired by Fast R-CNN [19] and will be elaborated on further now.

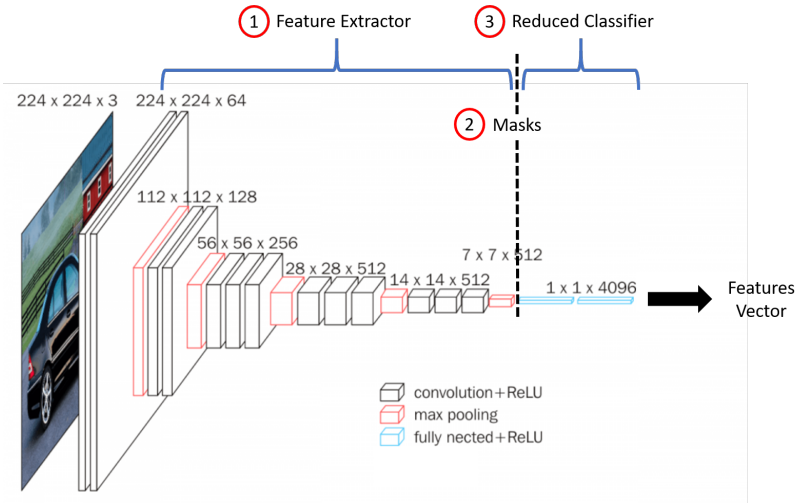

Figure 2.2: The proposed modifications to the VGG 16 model architecture. The intermediate steps of the process have been labelled 1-3.

Trying to extract features from an ROI is a common task - primarily in object detection. One such CNN architecture that aims to do this efficiently is Fast R-CNN [19]. As the location of the object(s) in the images is(are) unknown to the CNN, "proposals" are made for the regions where it(they) may lie. These proposals are made by a region proposal method such as selective search [20]. Referring to Figure 2.3, Fast R-CNN first processes the entire input image through the convolutional and max pooling layers to produce a

convolutional feature map. It then projects the ROIs (or proposals) onto the feature map by scaling the ROI based on the receptive field of the ROI in the feature map [21].

Due to the convolutional and max pooling layers of any given CNN, there is a one-to-many correspondence between the pixels in the feature maps and the pixels in the image domain. The receptive field for a particular feature-map pixel is then the pixels in the image domain that were used in the computation of that feature map pixel value. It follows that the receptive field for an ROI in a feature map is the union of all of the receptive fields for the pixels that comprise the ROI.

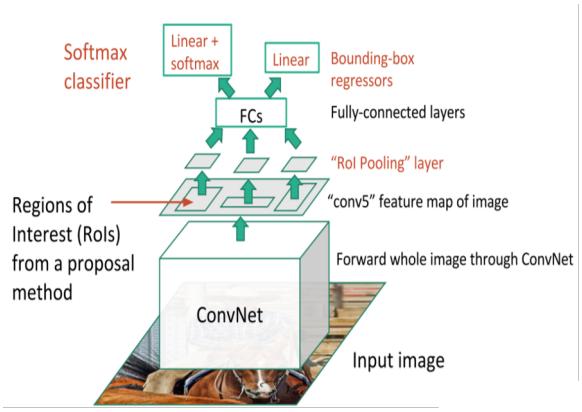

Figure 2.3: A visual representation of the Fast R-CNN architecture [3].

Now, a rectangular ROI in the image domain can be aligned with the feature maps by first projecting the top-left corner point of the ROI onto a pixel in the feature maps, such that this corner point in the image domain is closest to the center of the receptive field of that feature map pixel [21]. The process is then repeated for the bottom-right corner of the ROI. Let the top-left corner of the ROI be (x_1, y_1) and the bottom-right corner be (x_2, y_2), then their corresponding locations in the feature map can be computed as:

$$(x_1^*, y_1^*) = (\lfloor x_1/S \rfloor + 1, \lfloor y_1/S \rfloor + 1) \quad \text{and} \tag{2.1}$$

$$(x_2^*, y_2^*) = (\lceil x_2/S \rceil - 1, \lceil y_2/S \rceil - 1) \tag{2.2}$$

where S is the product of all previous strides in the CNN [21].

In the VGG 11 CNN, since each of the convolutional filters have stride 1 and the max-pooling layers have stride 2, it follows that $S = 32$. Thus, using Eqns. (2.1 & 2.2), any rectangular ROI could be simply mapped to the feature map.

The value for S can also be intuitively seen as the ratio of the dimensions of the input image to the feature map of interest. For example, in the VGG 16 architecture displayed in Figure 2.1, the input image has dimensions 224x224 and the feature map has dimensions 7x7, thus $S = 224/7 = 32$. It is easily shown that this is also the case for the VGG 11

architecture used, and thus is equivalent to the alternative calculation of S mentioned above.

However, these methods are specified for rectangular ROIs and the ward shapes are certainly not rectangular (or consistent in shape). This led Dai et al. [22] to introduce the idea of a Convolutional Feature Masking (CFM) layer to accommodate non-rectangular ROIs. Like the Fast R-CNN architecture, their method aims to isolate the relevant ROI from the last convolutional feature maps using the receptive field of the ROI in the image domain. However, they differ in that Dai et al.'s method also requires a binary mask outlining the ROI in the image domain [22].

In order to accomplish this, each activation (in the feature maps) is first projected onto the image domain as the center of its receptive field. Let the set of these centers be denoted $\boldsymbol{C}$. Each pixel in the binary mask for the image is then assigned to the nearest $c \in \boldsymbol{C}$. Thus, each center "collects" multiple of the binary mask's pixels. The pixels collected by each center are then averaged and thresholded (by 0.5) resulting in a binary mask in the feature map domain. This mask is then applied to the convolutional feature maps by multiplying each pixel in the feature maps by their corresponding mask pixel to obtain the masked convolutional feature maps [22]. If the pixels of the transformed mask are denoted m_i ($m_i \in \{0, 1\}$), then the pixels, f_i, in any masked convolutional feature map can be obtained from the following relationship with the unmasked feature map pixels, p_i:

$$f_i = m_i p_i = \begin{cases} p_i & \text{if } m_i = 1 \\ 0 & \text{if } m_i = 0 \end{cases}$$

It follows that these masked feature maps contain the original values for the pixels of the feature maps inside the ROI, while the remainder of the pixels are set to zero.

Although this method is relatively straightforward to understand, the implementation is far more complex (particularly the use of receptive-field arithmetic to "collect" the pixel values). Given the time constraints for this paper, it does not seem feasible. Thus, an alternative method that tries to capture the ideas of Dai et al. is implemented. The only difference being the manner in which mask pixels are "collected" (i.e. the use of receptive-field arithmetic), while the proceeding steps are equivalent: the collected pixels are averaged and then thresholded.

As an alternative to using receptive fields, the masks are projected onto the convolutional feature maps by resizing them to 7x7 (the same height and width as the convolutional feature maps in the VGG 11 model) before thresholding the resized masks back to binary[1].

[1]To clarify, the term "binary" is used here to mean that the mask pixels can only take on two values, 0 and 255, rather than the traditional values of 0 and 1.

It is noted that this ROI-projection method does not have grounding in the literature and is acknowledged as a limitation of the methodology followed.

However, there is no obvious reason to suggest why it would produce vastly different results from Dai et al.'s method as the use of the receptive field is, in essence, a very specific way of resizing the binary masks. Furthermore, their method is inevitably going to result in lost accuracy given that the binary masks in this study are being mapped from 400x400 pixels to only 7x7 pixels - an extremely lower-dimensional approximation. This information loss is amplified the more complex the mask shape is (i.e. more corners and diagonal lines), as the fewer pixels the decimated masks have, the harder it is to replicate these shapes (see Figure 2.4 for an example of this). Thus, it appears that the potential accuracy gain of using Dai et al.'s method in place of this paper's resizing method could be insignificant. However, this cannot be said for certain. With the hope that these methods will not produce vastly different results, the implementation of the alternative method will now be outlined.

As mentioned previously, the masks are first resized to 7x7 pixels. When decimating (or shrinking) an image, there needs to be some method defined to calculate the pixel values for the smaller image from the original. This method corresponds to the interpolation parameter when implementing the openCV resize method. The INTER_AREA interpolation method is regarded as the preferred method for image decimating as it is the only method that gives moiré-free results [23]. Furthermore, using this method, every pixel in the decimated image is calculated by taking the average of a neighbourhood of pixels in the original image, which seems to roughly replicate the collection and aggregation steps of Dai et al.'s method. Thus, this interpolation method is used to resize the image masks.

However, when shrinking a binary image using this interpolation method, the resulting image is not binary. After resizing the 400x400 pixel mask, the resulting decimated mask has pixel values ranging from 0 (pure black) to 255 (pure white) (for an example, refer to the top-right image in Figure 2.4). Following the method of Dai et al. [22], image thresholding is then required to convert the decimated mask back to binary. For a given threshold, t, the mask is mapped to a binary image using the following rule:

$$\text{pixel\_new} = \begin{cases} 255 & \text{if pixel\_old} \geq t \\ 0 & \text{if pixel\_old} < t \end{cases}$$

Since this interpolation method uses the average of neighbouring pixel values to determine the pixel values in the decimated image, the threshold used is $t = 127.6$ as this suggests

that there were more white pixels than black pixels in the average computation[2]. Put alternatively, there is marginally more "support" that the pixel should be white than black (a pixel intensity of 127.5 would equate to equal support). An example of this thresholding, in conjunction with the mask resizing, can be seen in Figure 2.4.

Figure 2.4: A demonstration of the mask resizing and thresholding. Left: A full 400x400 pixel mask. Top-right: The mask after decimating the image to 7x7 pixels. Bottom-right: The mask after thresholding the decimated mask.

The intuition behind setting the pixels in the feature map to zero if they fall outside of the ROI follows from the workings of the fully-connected layers. Without loss of generality, consider the fully-connected neural network in Figure 2.5. If one of the explanatory variables, x_{ij}, were to be set to zero, then that variable would have no impact on the activations in the first hidden layer. Since the activations in any proceeding hidden layer are dependent on the previous layer's activations, the variable would then have no effect on the network.

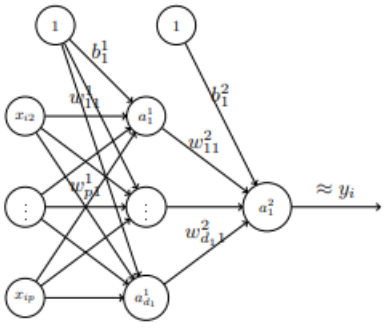

Figure 2.5: A generic neural network structure with one hidden layer.

Relating this idea back to the CNN of interest, after the feature maps are created, they are then flattened and passed through the fully-connected layers, where each pixel in each

[2]Dai et al.'s method used a threshold of 0.5 (or 127.5 in this study). However, it would be very surprising if the marginal difference in the thresholds had an impact on the results.

of the 512 feature maps can be regarded as an input variable. Thus, setting the pixels in the feature maps outside of the ROI to zero will ultimately eliminate the effect of those pixels on the fully-connected layers. A visualisation of the ROI-isolation for a particular feature map can be seen in Figure 2.6. Referring to Figure 2.6, it is evident that the feature map pixels outside of the ROI have been set to zero.

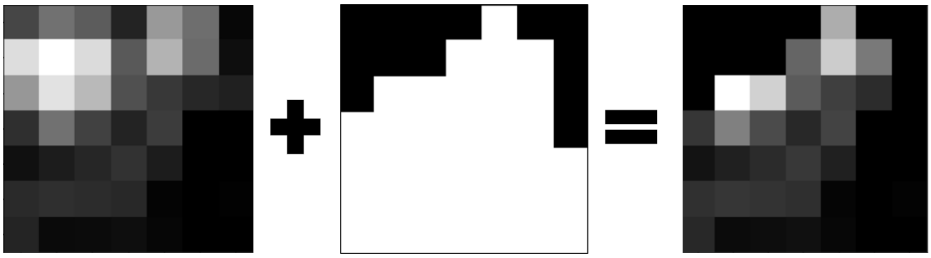

Figure 2.6: A demonstration of the use of a resized mask to isolate the ROI in a feature map.

Once, the ROIs have been extracted from the feature maps, the masked feature maps are then flattened and run through the first two fully-connected layers to produce a feature vector of length 4096 for each image. Unfortunately, this is where the replication by Mather [16] differs from that of Jean et al. [1]. The latter made use of a fully-convolutional CNN and thus replaced the fully-connected layers of the VGG architecture with more convolutional layers. However, using the CNN as a feature extractor by removing the final fully-connected layer (as done by Mather [16]) is common practice in the literature [24].

Although it would be preferable if the methodology of Jean et al. [1] was followed exactly, the use of the pre-trained CNN by Mather [16] has restricted this aim since more convolutional layers cannot simply be added to the CNN without retraining the model. However, the method followed by Mather [16] (i.e. the removal of the final hidden layer) has support from the literature and thus there appears to be no reason why it should not be applicable here [24].

It is common practice for the feature vectors to be passed through a ReLU activation function so that all of the features are non-negative [24]. It has been shown that, in the absence of this, the quality of the features extracted can be slightly worse [24]. In this paper, the ReLU has not been applied due to time constraints.

As a summary, all of the modifications to the VGG architecture can be put together to form Figure 2.7. The masking of the feature maps and the removal of the final hidden layer can clearly be seen.

Once the feature vector for each image relating to a particular ward has been extracted, following the methodology of Jean et al. [1], the feature vectors are then aggregated. This is computed by taking the component-wise average over the feature vectors corresponding

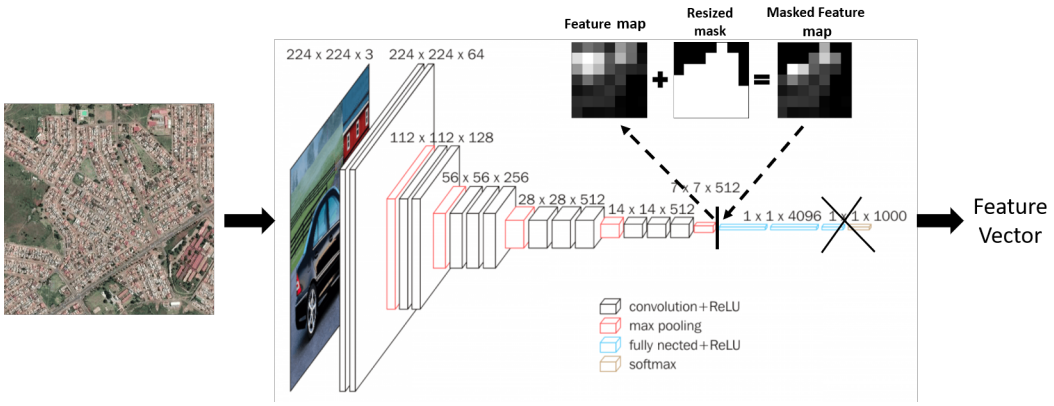

Figure 2.7: The final modifications made to the VGG architecture.

to the ward of interest. Thus, each ward has an aggregated feature vector of length 4096 associated with it.

2.2.3 Visualizing Features

At this point it may not be apparent what these features represent. One way to visualise what the CNN regarded as important when encoding information into the feature vectors is by visualizing the activation maps at a given convolutional layer. Following the approach by Jean et al. [1], the activation maps of the fifth convolutional layer will be investigated.

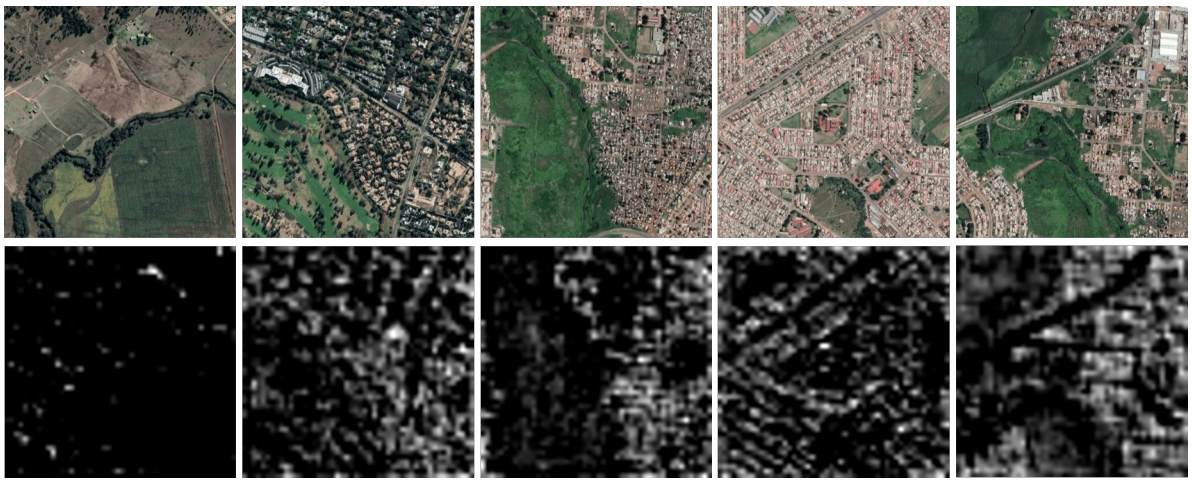

Figure 2.8: A visualization of a subset of the activation maps in the fifth convolutional layer. When comparing the activation maps to the daytime satellite images, it appears that the activations correspond to urban areas or where there are buildings.

Referring to Figure 2.8, it appears that urban areas or buildings are activating these particular filters. This gives evidence to suggest that the presence of buildings was regarded as an important feature by the CNN when accounting for the variations in nightlight

intensity. Referring to the left-most column of images, the CNN was even able to identify the house present near the top-right of the image despite it being very small and in isolation. Regarding the presence of buildings as an important feature seems logical as buildings generally produce light of some sort (from indoor or outdoor lights), and thus they would be regarded as a distinguishing factor when determining nightlight intensity.

Referring to Figure 2.9, it appears the the CNN could have also viewed rural areas, bodies of water, major roads, urban green spaces and tree lines as important features. However, since the above assertions are made from generalizing the results in the visual investigation, which is undertaken on a very small subset of the images, it cannot be said for certain that these are in fact the important features. For example, in the second column, the dam seems to have been identified. However, without examining more images with dams, it cannot be inferred with certainty that the dam is an important feature, and not simply any large dark-coloured area.

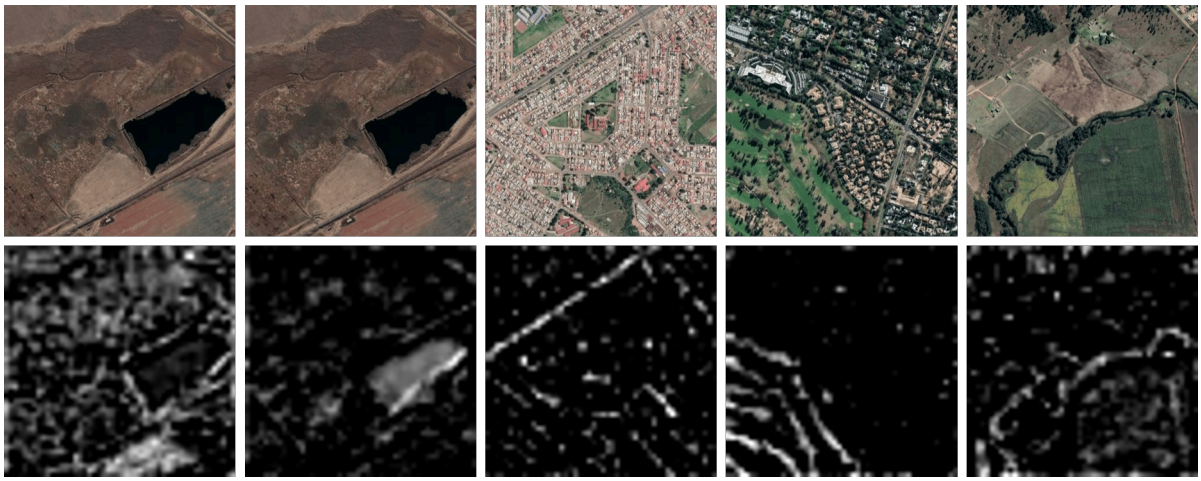

Figure 2.9: A visualization of a subset of the activation maps in the fifth convolutional layer. From left-to-right, the activations seems to correspond to the rural area, the dam, the main road, the golf course and the tree line.

It is interesting to note that the important features identified by Jean et al.'s CNN were urban areas, roads, bodies of water, and agricultural areas, which roughly corresponds to what was found above [1]. Whether this is a result of the researchers "seeing what they wanted to see" would require further analysis. This could involve using one of the numerous advanced techniques for visualizing the features extracted by CNNs, such as saliency maps [25]. However, this is regarded as out of the scope of the study and thus will not be pursued further.

The final feature set comprised of a feature vector of length 4096 for each ward will now be used for further analysis. This will firstly involve reducing the dimensions of the data, before examining the extent of clustering among the wards.

Chapter 3

Dimension Reduction

The dimension of the data refers to the number of attributes or features per observation. As mentioned previously, the data has dimension 4096 corresponding to the 4096 features (or measurements) extracted by the CNN for each of the 508 Gauteng wards. This would be considered a high-dimensional data set. In the context of clustering algorithms, this high-dimensionality is problematic since it has been shown that the concept of proximity, distance and nearest neighbours might not even be "qualitatively meaningful" in this high-dimensional space [26]. The two clustering algorithms explored in Chapter 4, namely K-means and HDBSCAN, rely heavily on the computation of distances and nearest neighbours, respectively, to distinguish observations. Thus, the dimensionality of the data needs to be reduced. One popular method that can be applied to give a low-dimensional linear approximation of the data is Principal Component Analysis (PCA).

3.1 Principal Component Analysis (PCA)

PCA is a method that makes use of linear combinations of features to capture the information in the covariance (or in the case of standardised features, the correlation) matrix of the features [27].

Let $\boldsymbol{X}$ denote the 508×4096 feature set. Standardisation of $\boldsymbol{X}$ is necessary whenever the measurement scale of features differ significantly, which could lead to these features dominating the PCA[1]. Since the images were normalized before being passed through the CNN, the features in $\boldsymbol{X}$ all have similar scales. Thus, it is not necessary to standardize $\boldsymbol{X}$ before performing PCA and the covariance matrix, $\boldsymbol{\Sigma}$, can be used.

The vectors used for the coefficients of the linear combinations are known to be the eigenvectors of $\boldsymbol{\Sigma}$ [27]. To be clear, $\boldsymbol{\Sigma} = \text{cov}(\boldsymbol{X})$, thus $\boldsymbol{\Sigma}$ is a 4096×4096 matrix. The eigenvectors and eigenvalues of $\boldsymbol{\Sigma}$ are then calculated. Let $\boldsymbol{e}_i$ denote the eigenvectors with $i = 1, \ldots, 4096$ and λ_i denote the respective eigenvalues. Thus, there exists 4096 principal components (PCs), defined as P_i where

$$P_i = \boldsymbol{X}\boldsymbol{e}_i$$

[1] "Dominating" here refers to when some features are given a much higher weight or level of importance such that other variables are regarded as being negligible or unimportant.

for each $i = 1, \ldots, 4096$.

It is known that the proportion of variance in $\boldsymbol{X}$ captured by the i^{th} PC, P_i, is given by

$$\frac{\lambda_i}{\sum_{i=1}^{4096} \lambda_i}$$

The aim of PCA is to reduce the dimensionality to say $p << 4096$ such that the total variance in $\boldsymbol{X}$ captured by the first p PCs is sufficiently large [27]. The cumulative variance captured by the first p PCs is calculated as

$$\frac{\sum_{i=1}^{p} \lambda_i}{\sum_{i=1}^{4096} \lambda_i}$$

The application of PCA for dimension reduction is especially helpful for K-means clustering [28]. It can be proved that PCA provides continuous solutions to the discrete indicators used by K-means to indicate membership to clusters [28]. The discrete indicators simply refer to the group or cluster an observation is assigned to. Once the PCs are computed, these can be back transformed to find the K discrete cluster indicators. This derivation is outside of the scope of this paper. For details, see [28].

For the reason stated above, PCA will be implemented on $\boldsymbol{X}$ after which a sufficient number of PCs will be used to cluster the Gauteng wards into K clusters. This will be done by considering the cumulative variance captured by the components in the implementation section.

3.1.1 PCA Implementation

In the PCA implementation, only the first 508 PCs are considered to avoid the case where the number of variables exceed the number of observations[2]. This seems like a reasonable simplifications, since the increase in cumulative variance contributed by each PC beyond the first 508 components is negligible (<0.001%).

Thus, the first 508 PCs will be used. The proportion of the variance in $\boldsymbol{X}$ explained by the i^{th} PC (left) and the cumulative variance captured up to and including the i^{th}

[2]This was found to be an artifact of the PCA-function used where the number of components are not allowed to exceed the number of observations. However, as shown later in this section, there is no need to consider more than the first 508 components.

component (right) is plotted in Figure 3.1.

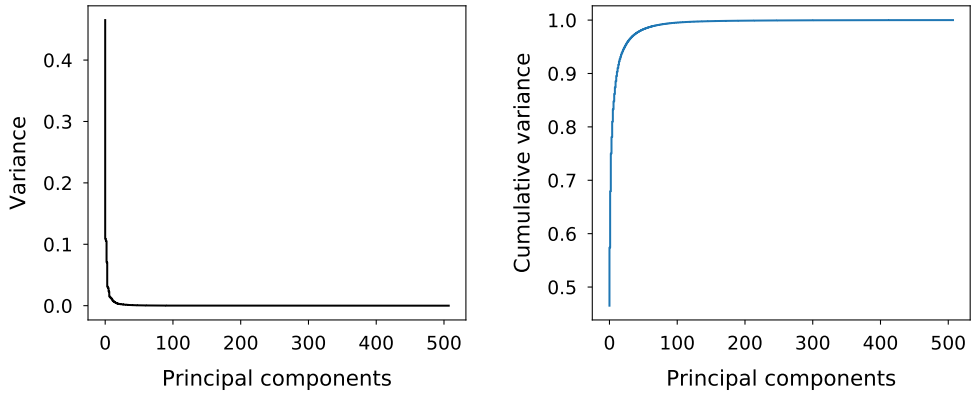

Figure 3.1: The proportion of variance in $\boldsymbol{X}$ captured by each of the first 508 PCs (left) and the corresponding cumulative variance (right)

The variance captured by the i^{th} component flattens out quite quickly. Although the cumulative variance increases drastically at first, it flattens out after approximately 30 PCs. This relates to the fact the the variance captured by the PCs after the 10^{th} component, although relatively small, are all approximately equal. Thus the cumulative variance gradually tends to 1 after the 10^{th} component. To see this effect more clearly, the first 150 (left) and 50 (right) PCs are each plotted in Figure 3.2.

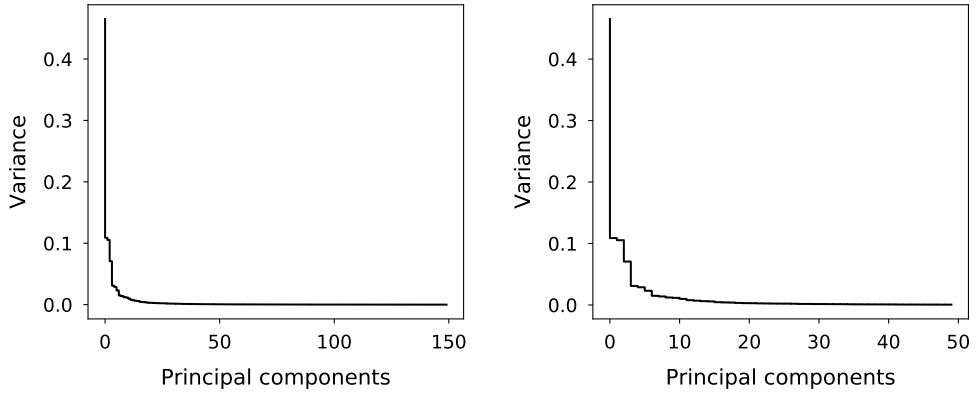

Figure 3.2: The proportion of variance in $\boldsymbol{X}$ captured by each of the first 150 (left) and 50 (right) PCs

From Figure 3.2 it is clear that the proportion of variance captured by the components flattens out after 20 components, after which it falls to almost zero. However, this alone is not sufficient to decide how many components to use for the clustering. To better inform this decision, the cumulative variance is superimposed on the graph that shows the variance explained by the first 6, 18, 26 and 40 PCs. These are shown in Figure

3.3. These specific points were chosen where there was a slight elbow in the graph of the variance explained by each component.

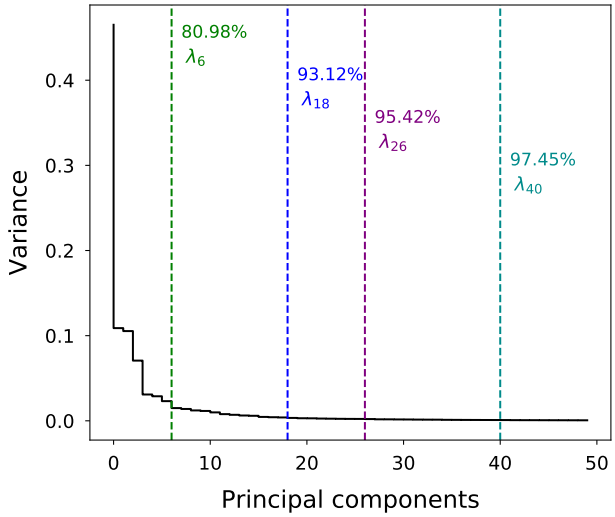

Figure 3.3: The cumulative variance explained by the first 6, 18, 26 and 40 PCs. Each λ_i indicates the cumulative variance calculated at the i^{th} PC.

The most noticeable increase is moving from 6 to 18 components, where the cumulative variance explained increases by 12.14%. Each next elbow leads to roughly a 2% increase in variance explained. Any of the above cut-off points are a vast improvement from the original dimensionality of 4096. However, when considering that there are only 508 observations, the choice of using the first 6 and 18 components appears to be a better balance between the dimensionality, number of observations and cumulative variance explained. For the remainder of the paper, let

- 6PCs denote the first 6 PCs and
- 18PCs denote the first 18 PCs

In the sections to follow, the K-means algorithm will be implemented on both 6PCs and 18PCs for a sequence of K-values. A choice of K will then be made based internal validation metrics and the results for both will be compared.

Chapter 4

Clustering

The PCs derived in the *Dimension Reduction* Chapter above contains useful information regarding the distinguishing characteristics of each ward. In the absence of a response variable or an up-to-date measurement of poverty for each ward, the information captured by these features is exploited by grouping similar wards together, identifying "clusters" in the data. This paper will apply two clustering algorithms: (1) K-means and (2) HDBSCAN.

This chapter begins by broadly defining Clustering and the main categories it is divided into. It then specifically defines the K-means clustering algorithm, taking care when determining the initialization method. Thereafter, the HDBSCAN algorithm will be explained and implemented. For both algorithms, the optimal number of clusters will be determined using internal validation metrics. The spatial distribution of these optimal cluster-solutions will then be examined, before using poverty data to determine whether the clusters are in fact related to poverty.

4.1 Introduction to Clustering

When working with multivariate data (data with multiple measurements or attributes for each observation), exploratory data analysis can be a powerful tool to understand complex patterns in data. The method of finding groupings or clusters of similar observations within data is known as Cluster Analysis. It differs from classification methods in that it does not require data where true group labels[1] is known [27], instead grouping is based on some measurement of similarities between data points. A cluster centroid is then defined as the average of the observations within a cluster.

One of the most common measurements used is the Euclidean distance between observations and cluster centroids. This is equivalent to the **Sum of Squared Error (SSE)** formulation given as:

$$SSE = \sum_{i=1}^{K} \sum_{\boldsymbol{x}_j \in \boldsymbol{C}_i} ||\boldsymbol{x}_j - \boldsymbol{c}_i||_2^2 \tag{4.1}$$

where $||\cdot||_2$ denotes the Euclidean ($\mathcal{L}_2$) norm, $\boldsymbol{c}_i$ is the centroid for cluster $\boldsymbol{C}_i$ and $\boldsymbol{x}_j$ is the j^{th} data vector in the i^{th} cluster [29]. In the above equation there exists K such

[1]Group labels refer to an identifying factor assigned to an observation, for example "Smoker" or "Non-smoker".

clusters. Minimising Eqn. (4.1) is equivalent to minimising the distance between each observation in a cluster and its centroid, summed over all K clusters. SSE is often also referred to as Inertia, but in some cases the formulation differs slightly.

However, when considering all possible partitions of N data points into K non-empty clusters, there exists

$$\mathcal{S}(N,K) = \frac{1}{K!}\sum_{i=0}^{K}(-1)^{K-i}\binom{K}{i}i^N$$

number of partitions, known as Stirling numbers of the second kind [29]. This is approximately $\frac{K^N}{K!}$, which explodes as N increases. Thus, considering all possible groupings and choosing the single combination that minimises some measurement, such as Eqn. (4.1), can be computationally expensive and infeasible. This has lead to the development of a number of algorithms that consider the most suitable clusters whilst avoiding the iteration between all configurations. Such algorithms can broadly be divided into two categories: hierarchical and non-hierarchical methods.

4.1.1 Hierarchical and Non-Hierarchical Methods

Hierarchical clustering algorithms consist of successively merging similar observations or dividing clusters, known as Agglomerative hierarchical and Divisive hierarchical methods, respectively. The former begins with as many groups as observations and ends with one single group. The latter starts with a single group of all observations and iteratively separates the observations until each group consists of a single observation. The merging and division are based on similarities of observations within a group and dissimilarities between groups [27].

Non-hierarchical methods do not follow a tree-like structure (see Figure 4.1). The number of clusters, K, can either be set in advance or determined as part of the implementation procedure. The method determines cluster allocations simultaneously for each observation and does not implement a hierarchical structure (see Figure 4.1). It is more suited to large datasets since it does not require the calculation of a matrix of all pair-wise distances between observations to sequentially assign the next best observation to a cluster or deciding on the next best division of clusters [27].

The starting point for non-hierarchical methods can either be an initial partition of observations into K groups, or an initial set of cluster centroids [27]. The method used to define centroids and the allocation rule to assign observations to clusters are defined through different algorithms. One of the most common algorithms is known as the K-means algorithm.

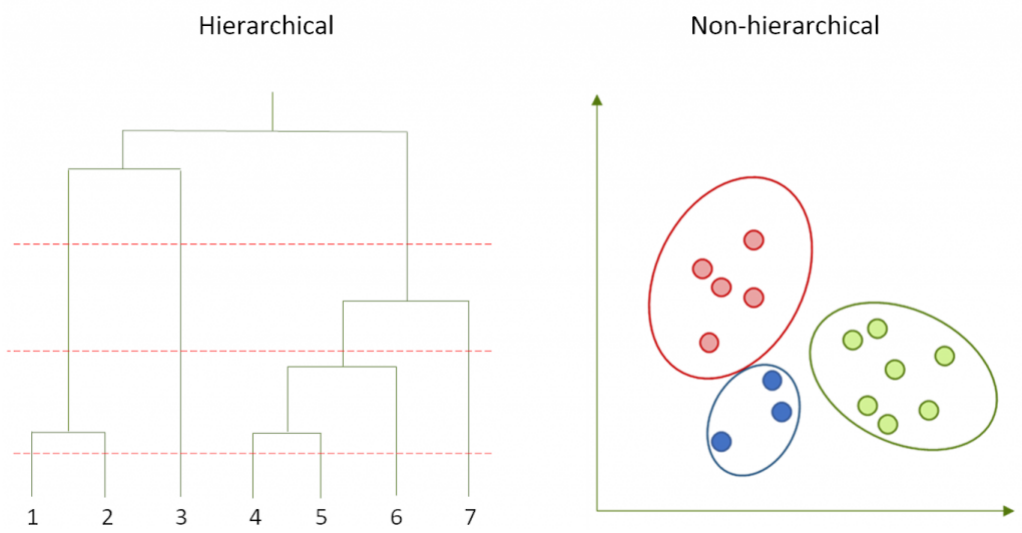

Figure 4.1: An example of Hierarchical clustering (left) where observations sequentially pass through a decision tree structure to get allocated to one of 7 clusters. An example of non-hierarchical clustering (right) where observations are simultaneously assigned to a cluster within which observations are similar. (Example sourced from [4])

4.2 K-means Algorithm

4.2.1 Introduction

K-means is one of the most widely used clustering algorithms due to its computational ease and simple interpretation [29]. The basic idea is that the algorithm assigns each observation to the cluster with the nearest centroid, or mean. The steps are as follows:

0. Determine the number of clusters, K, to be used
1. Starting with K initial centroids, assign observations to the cluster with the smallest Euclidean distance to the centroid
2. Recalculate the cluster centroid as the average of all observations within each cluster
3. Reassign observations to clusters with the nearest centroid
4. Repeat step 2 and 3 until no more reassignment of observations take place, in other words, when the algorithm converges

From Step 4 above, a hyperparameter for convergence can be defined. Instead of reassigning observations to a different cluster whenever the distance between the observation and the new cluster centroid is less than the current centroid, the hyperparameter can specify a threshold that this difference needs to exceed before reassignment is justified.

Changing this parameter influences the time to convergence, or in some cases, whether convergence would take place at all.

In the algorithm above this parameter is set to zero, which means a reassignment will take place even if a different centroid is only marginally closer to an observation than the cluster it is currently assigned to.

K-means cannot, however, be applied to 6PCs and 18PCs without careful consideration of some key factors. Specifically, two factors are considered: the choice of initial cluster centroids and the number of clusters, K. When an initialisation method has been chosen, a sequence of values for K can be used to implement the K-means algorithm and the most suitable choice of K determined. This choice will be informed by a few metrics, including a silhouette score, Inertia and the Davies-Bouldin Index. The sections below defines a theoretical background to the initialisation methods and each internal metric and thereafter their application to 6PCs and 18PCs.

4.2.2 Initialisation

K-means uses a back-propagation method in an effort to find global minima. The issue of initialisation refers to the algorithm's susceptibility to converge to local minima based on the choice of initial centroids [29]. Numerous studies have focused on the choice of initialisation method to address this potential issue, which will briefly be outlined in the following section whereafter the most suitable initialisation method will be indicated.

The K-means++ Algorithm

One of the more recent initialisation methods is the K-means++ algorithm. This method was introduced by David Arthur and Sergei Vassilvitskii in 2007 [30]. It augments step 1 in the K-means algorithm above by carefully choosing K initial cluster centroids.

To describe this method, first define some notation. Let $\mathbf{X}$ be an $n \times p$ matrix where n is the number of observations and p the number of features. Each row, $\mathbf{x}_j$, represents a single observation j, thus $\mathbf{x}_j$ is an $1 \times p$ row vector and $j = 1, \ldots, n$. As above the aim is to find K centroids, $\boldsymbol{c}_i$ with $i = 1, \ldots, K$. Let $\mathbf{S}_i$ be the set of centroids selected after the i^{th} iteration. Clearly, after K iterations, $\mathbf{S}_K = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_K\}$. Let $\sim \mathbf{S}_i$ be the set of observations from $\mathbf{X}$ that have not been selected as centroids after the i^{th} iteration.

Now define $D^i(\mathbf{x}_j)$ as the shortest distance between observation $\mathbf{x}_j \in \sim \mathbf{S}_i$ and all centroids in the set $\mathbf{S}_i$. Thus at each iteration $i = 1, \ldots, K$ the quantity $D^i(\mathbf{x}_j)$ is recalculated

for each $\mathbf{x}_j \in\sim \mathbf{S}_i$. In other words, at the $(i+1)^{th}$ iteration, consider each $\mathbf{x}_j \in\sim \mathbf{S}_i$, i.e. all the observations that have not yet been selected as a centroid in one of the previous iterations. For each of these observations, assign the value $D^i(\mathbf{x}_j)$, representing the shortest distance from that observation to any of the centroids selected in the previous iterations.

Finally, define the following probability of choosing observation $\mathbf{x}_j$ to be centroid $\boldsymbol{c}_{i+1}$ at iteration $(i+1)$ as

$$\frac{D^i(\mathbf{x}_j)^2}{\sum\limits_{\mathbf{x}_j \in\sim \mathbf{S}_i} D^i(\mathbf{x}_j)^2} \tag{4.2}$$

Then the K-means++ algorithm extends step 1 of the K-means algorithm above as follows:

Step 1: (a) Choose $\boldsymbol{c}_1$ uniformly at random from $\mathbf{X}$

(b) Choose the next centroid, $\boldsymbol{c}_i$, $i = 2, \ldots, K$, by drawing $\mathbf{x}_j$ from $\sim \mathbf{S}_i$ with probability as defined in Eqn. (4.2)

(c) Repeat step (b) until K such centroids have been selected

The algorithm then proceeds in the same way as the K-means algorithm outlined above. By weighting each subsequent observation by Eqn. (4.2), observations remaining in the set $\sim \mathbf{S}_i$ (before the $(i+1)^{th}$ centroid, $\boldsymbol{c}_{i+1}$, is selected) no longer have an equal probability of being selected as the next centroid. Instead, observations further apart from centroids in the set $\mathbf{S}_i$ have greater probabilities of being selected. This means that there is a greater probability of choosing initial centroids that are further apart.

This augmentation leads to faster convergence of the algorithm than the traditional K-means algorithm with random initialisation or seeding [30]. This is especially true for large datasets, where the K-means++ algorithm far outperforms the K-means algorithm [30].

The more modern K-means++ algorithm and seven other initialisation methods were compared by Celebi, Kingravi and Vela (2013) [29]. Amongst them were Forgy's method [31], which randomly assigns observations to each of the K clusters, and MacQueen's second method [32], that chooses observations at random to serve as the initial centroids. MacQueen argued that randomly choosing observations as centroids would mean that points are likely to be chosen from dense regions that would then implicitly serve well as initial centroids [29].

The study included two sets of experiments run on both well-known and synthetically created datasets of varying sizes and complexity. The aim was to compare each initialisation method on the basis of quality, or effectiveness, and speed, or efficiency, and make recommendations to future researchers in accordance with their specific problem.

It is recommended that in general Forgy and MacQueen's methods should be avoided as they were found to be unreliable and unstable (the results varied greatly when applied to different datasets). With regards to the K-means++ method, it is found to produce comparable results with the greedy K-means++ algorithm. This algorithm only slightly modifies the K-means++ method in that at each iteration the observation that decreases the SSE most is chosen as the next centroid. For datasets with fewer than 10 000 observations, these methods are recommended.

Furthermore, the scikit-learn [33] library provides well-documented and user-friendly implementation of the K-means++ algorithm. Due to the relatively small dataset size of this paper (only a total of 508 observations corresponding to the 508 Gauteng wards) and the easy implementation by the packages provided by scikit-learn, the K-means algorithm is implemented using the K-means++ initialisation method. The application to 6PCs and 18PCs is outlined in the implementation section.

4.2.3 Choice of K: the Number of Clusters

Once the dimensionality of the data has been reduced, the number of clusters, K, has to be determined. This is done by creating a sequence of K-values and implementing the K-means algorithm using the K-means++ initialisation method for each K. Thereafter, the quality of the clustering can be determined by considering a number of measures, namely Inertia, silhouette score and the Davies-Bouldin (DB) Index. Inertia is identical to SSE in the context of K-means (see Eqn. (4.1)). These are particularly useful metrics in this case since the data does not have "ground truth"-labels, i.e. the wards do not have a true classification of poverty levels that is know at the onset of the analysis.The silhouette scores, DB Index and application of Inertia in the Elbow method are defined below. All three methods are then applied to 6PCs and 18PCs and the results compared.

Silhouette Scores

The silhouette score is a metric that combines a measure of the similarity of an observation to its cluster and dissimilarity with observations from other clusters [34].

Formally, the silhouette coefficient [34] of observation $\boldsymbol{x}_i$ can be defined as

$$s(\boldsymbol{x}_i) = \frac{b(\boldsymbol{x}_i) - a(\boldsymbol{x}_i)}{\max\{a(\boldsymbol{x}_i), b(\boldsymbol{x}_i)\}} \tag{4.3}$$

where

- $a(\boldsymbol{x}_i)$ is the average distance between $\boldsymbol{x}_i$ and all observations within the same cluster
- $b(\boldsymbol{x}_i)$ is the average distance between $\boldsymbol{x}_i$ and all observations in the nearest cluster to the cluster $\boldsymbol{x}_i$ is assigned to

From Eqn. (4.3) it is clear that the aim is for $a(\boldsymbol{x}_i) << b(\boldsymbol{x}_i)$ [34]. This would mean that observations in the same cluster are very similar and observations in different clusters are dissimilar. When $a(\boldsymbol{x}_i) \approx 0$ and $b(\boldsymbol{x}_i)$ tends to infinity $\forall\ i$, i.e. observations in the same cluster are nearly identical and very far away from the next nearest cluster, $s(\boldsymbol{x}_i) \approx 1$, corresponding to the best possible silhouette coefficient. Similarly, when observations are misclassified[2], $a(\boldsymbol{x}_i)$ will be large and $b(\boldsymbol{x}_i)$ will tend to zero. This will lead to $s(\boldsymbol{x}_i)$ tending to -1. When the data is not easily separable and clusters are very similar, $a(\boldsymbol{x}_i) \approx b(\boldsymbol{x}_i)$ and $s(\boldsymbol{x}_i) \approx 0$.

The silhouette coefficients can easily be visualised through horizontal bar plots [34]. As an example, see Figure 4.2. The example is taken from an article by Orkphol and Yang [5] where observations are clustered using $K = 4$ clusters. Each bar corresponds to the silhouette coefficient of a single observation with the colour representing the cluster an observation is assigned to.

In this case, $K = 4$ would not be a good choice, since for two of the clusters all the observations' silhouette coefficients fall below the silhouette score (indicated by the dotted red line). Also note that for the blue cluster, some of the observations have a negative silhouette coefficient. From the scatter plot to the right it is clear that Cluster 1 and 3 lie very close together (in fact, they could easily be regarded as one cluster), meaning that the (small) negative silhouette score of observations in the blue cluster is understandable.

This visual aid in determining the quality of the clustering will become very helpful later on. These coefficients can be combined into a single measure, the silhouette score,

[2] "Misclassification" is defined differently than the classic definition in classification-type problems. Here, "misclassification" is less black-and-white, since the ground-truth labels of cluster allocations are not known. In this case an observation is deemed as being "misclassified" to a cluster if one of the internal validation metrics (such as the silhouette score) exceeds some threshold. Thus, it serves more as a probability indicator that suggests the probability that the observation should belong to a different cluster is greater than the probability of it simply being dissimilar to observations in the cluster it has been assigned to.

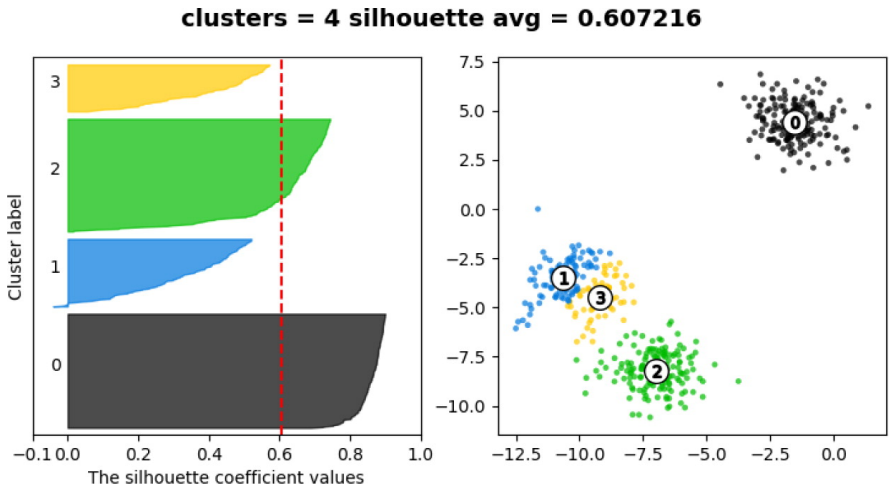

Figure 4.2: Left: A visualisation of silhouette coefficients. Each horizontal bar represents the silhouette coefficient of a single observation. The 4 different colours represent the 4 clusters. The dotted red line indicates the average silhouette score across all observations. Right: A scatter plot of all observations coloured by the cluster they are assigned to [5].

by taking the average of silhouette coefficients across all observations. It follows that a silhouette score close to 1 relates to an ideal clustering solution. Both the silhouette coefficients and the silhouette scores are employed below to determine a suitable choice of K, the number of clusters.

Davies-Bouldin Index

Similar to the silhouette score defined above, the Davies-Bouldin (DB) Index is a measure of the degree of separation between clusters [35]. To define this index mathematically, the following notation is introduced: Let

- K: total number of clusters
- $\boldsymbol{c}_i$: cluster centroid for the i^{th} cluster, with $i = 1, \ldots, K$
- s_i: average distance between all observations in cluster i and the centroid $\boldsymbol{c}_i$,with $i = 1, \ldots, K$,
- d_{ij}: distance between cluster centroid $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$

From the above, a measure, R_{ij}, is constructed as

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}$$

for $i \neq j$, such that R_{ij} is non-negative and symmetric.

The DB Index is then defined as

$$\text{DB} = \frac{1}{K}\sum_{i=1}^{K}\max_{i\neq j} R_{ij} \tag{4.4}$$

Thus for each cluster combination, i and j, R_{ij} is calculated. Then for each cluster i, the maximal value for R_{ij} over the remaining $K-1$ clusters is found. These maximal values are then averaged to obtain the DB Index. The index has a minimum of zero and the aim is to minimise this index over a range of K-values.

To utilise the DB Index, the K-means algorithm is implemented for a sequence of K. For each K, the index is calculated and 1/DB Index is plotted. The reason for plotting the inverse is for visual comparison with the silhouette score. The optimal number of clusters is taken to be where 1/DB Index reaches a maximum, or the index value reaches a minimum.

Elbow Method

The elbow method is another way to inform the choice of K. In this method, the K-means algorithm is implemented over a sequence of values for K. For each K, the SSE (from Eqn. (4.1)) is calculated and plotted.

An elbow is then identified on the graph as the point beyond which the marginal decrease in the metric diminishes or tends to zero. This elbow represents the best choice of K. The implementation section below utilises this method along with the silhouette score and DB Index to determine K.

4.2.4 Implementation and Results

This section implements the K-means algorithm using the K-means++ initialisation method on 6PCs and 18PCs. It begins by employing the silhouette scores, DB Index and elbow method to determine K, after which the data is clustered into K groups.

Choice of K, the Number of Clusters

Using 6PCs and 18PCs, the number of clusters to use in the K-means algorithm needs to be determined. To achieve this, a sequence is created for $K = 2, \ldots, 10$. The choice

of the sequence range seems reasonable when considering more than 10 different levels of poverty in one province.

At each iteration of K, the Inertia, silhouette score and DB Index are calculated and plotted[3]. Thereafter, an appropriate K is determined and the silhouette coefficient of each observation is plotted. The section below give the results for both 6PCs and 18PCs.

Results: 6PCs

Figure 4.3 shows the Inertia ($\times 10^{-3}$) for each $K = 2, \ldots, 10$. Ideally, in accordance with the elbow method mentioned above, the Inertia would have a clear elbow. Unfortunately, the figure shows the Inertia steadily decreasing over the whole range of K with no clear elbow. There does, however, seem to be a slight elbow at $K = 6$, but this alone is not sufficient to provide insight into an appropriate choice for K.

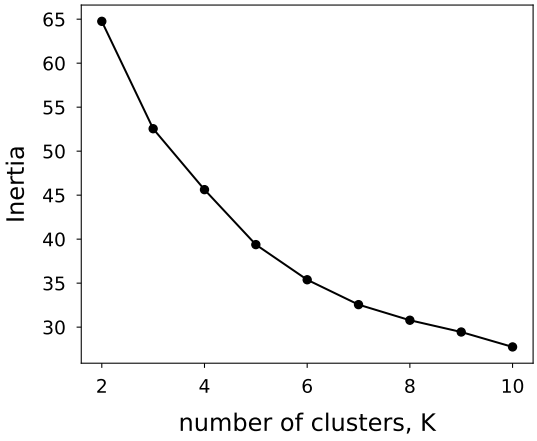

Figure 4.3: Inertia ($\times 10^{-3}$) calculated when applying the K-means algorithm to 6PCs for each $K = 2, \ldots, 10$.

Figure 4.4 shows the silhouette score (left) and 1/DB Index (right) for each $K = 2, \ldots, 10$. Both the silhouette score and 1/DB Index reach a maximum at $K = 2$ with a silhouette score of 0.31 and DB Index of 1.22. The next maximum occurs at $K = 6$ for both the silhouette score and DB Index, with a silhouette score of 0.24 and DB Index of 1.28. Both the silhouette score and 1/DB Index never exceed these maxima for $K > 6$. The silhouette coefficients are plotted in Figure 4.5 for $K = 2$ and $K = 6$.

For $K = 2$ (left) in Figure 4.5, it is clear that some observations seem to either be incorrectly assigned to Cluster 1, or that they are not easily separable from observations in clusters 0, indicated by the (small) negative silhouette coefficients for those observations. The silhouette score is indicated by the dotted red line and is found to be 0.31. Similarly

[3]Note again for visual comparison with the silhouette score, 1/DB Index is plotted and the aim is to maximise both 1/DB Index and the silhouette score.

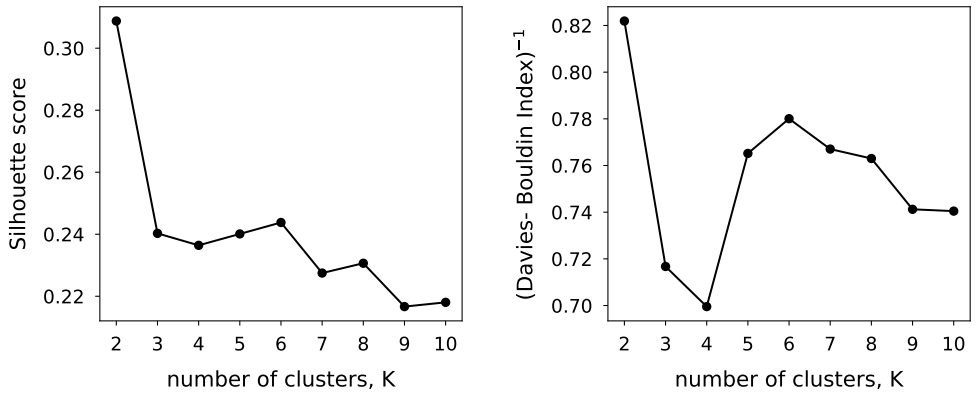

Figure 4.4: Silhouette score (left) and 1/DB Index (right) where the K-means algorithm is applied to 6PCs for each $K = 2, \ldots, 10$.

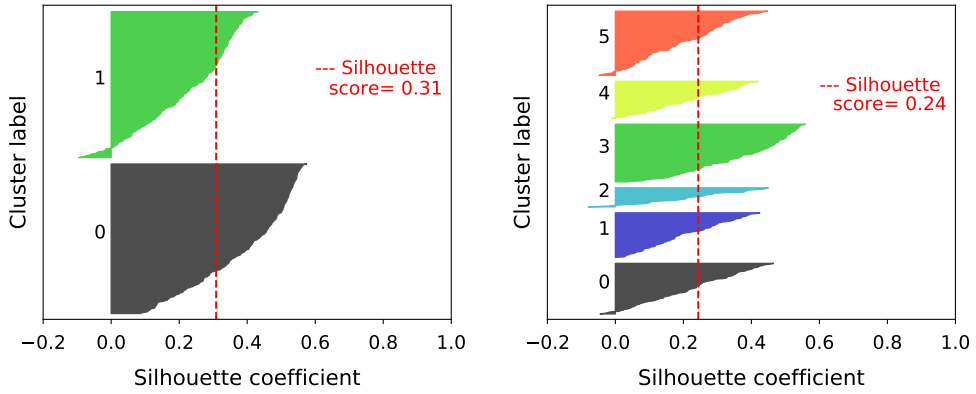

Figure 4.5: The silhouette coefficients for each observation for $K = 2$ (left) and $K = 6$ (right) using 6PCs.

for $K = 6$ (right), the silhouette score is 0.24. Again, the (small) negative silhouette coefficients for some observations in Cluster 0, 2 and 5 indicate that these observations are either incorrectly assigned to those clusters, or that they are very similar to observations in other clusters, i.e. clusters are not well separated.

In conclusion, the reasonable choice for K when using 6PCs seems to be $K = 2$ and $K = 6$ as indicated by both the silhouette score and DB Index. It is noted, however, that the magnitude of the silhouette score is concerningly low. Further, the DB Indices are far from the ideal value of 0[4]. Given the current method of initialisation and dimension reduction, it seems unreasonable to conclude that any clear clustering has been achieved. This notion will be referred to in more detail in a later section.

[4]A DB Index close to 0 would correspond to a very large number for 1/DB Index on the y-axis of Figure 4.4.

Results: 18PCs

Figure 4.6 shows the Inertia ($\times 10^{-3}$) for each $K = 2, \ldots, 10$ when using 18PCs. Again, the figure does not show a clear elbow in the Inertia, but only a slight bend around $K = 6$, indicated by the change in slope of the curve.

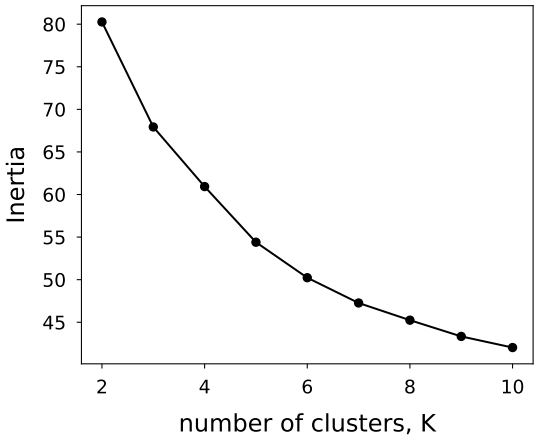

Figure 4.6: Inertia ($\times 10^{-3}$) calculated when applying the K-means algorithm to 18PCs for each $K = 1, \ldots, 10$.

Figure 4.7 shows the silhouette scores (left) and 1/DB Index (right) for $K = 2, \ldots, 10$ when using 18PCs for the clustering procedure. 1/DB Index reaches its first maximum at $K = 2$, with a DB Index-value of 1.22, thereafter at 3 and finally at 6. The index values for $K = 3$ and 6 are nearly identical, with values of 1.28 and 1.29, respectively. The silhouette score again reaches a maximum at $K = 2$ with a score of 0.27, then the next highest scores are at $K = 3$ with 0.23 and finally at $K = 6$ with 0.20.

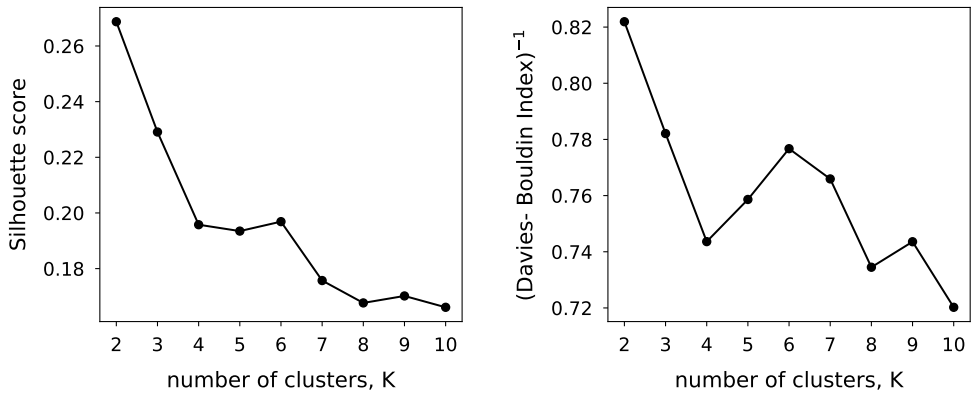

Figure 4.7: Silhouette score (left) and 1/DB Index (right) where the K-means algorithm is applied to 18PCs for each $K = 2, \ldots, 10$

Figure 4.8 shows the silhouette coefficients of all observations for $K = 2$ (top-left), $K = 3$ (top-right) and $K = 6$ (bottom).

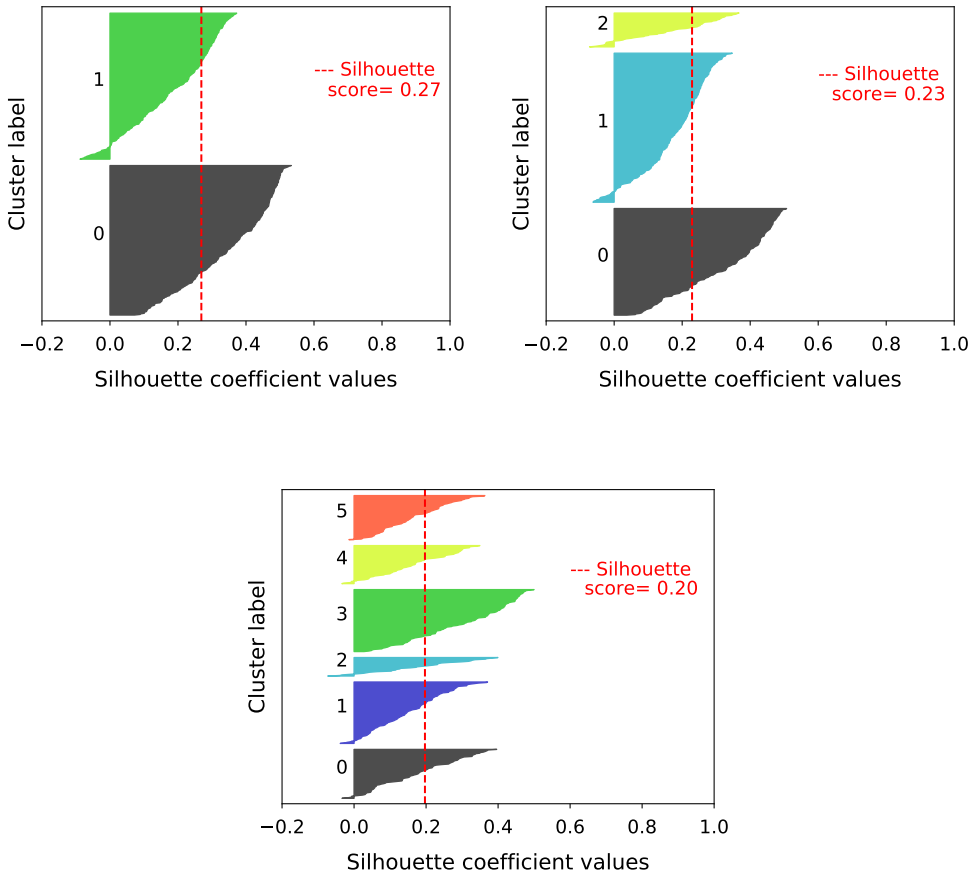

Figure 4.8: The silhouette coefficients for each observation for the case where $K = 2$ (top-left), $K = 3$ (top-right) and $K = 6$ (bottom) using 18PCs.

Similar to above when using 6PCs, some observations have (small) negative silhouette coefficients, indicating misclassification or observations that are not easily separable for $K = 2,\ 3$ and 6. Overall, similar silhouette scores are observed when using 6PCs and 18PCs.

In conclusion, from the silhouette scores and DB Indices, the most suitable choice for K seems to be 2, 3 and 6 clusters. In order to compare 6PCs and 18PCs, $K = 2$ and $K = 6$ will be used for the remainder of the paper. Discarding $K = 3$ for 18PCs seems reasonable since it has a very similar silhouette score and DB Index compared to the $K = 6$ result. It is again noted that the magnitude of the silhouette scores are still very low and the DB Indices are far from zero, even when considering $K = 2$ which is the best choice indicated by the internal validation techniques. Again, this will be referred to later in the *External Validation* section.

Table 4.1 summarises the silhouette scores and DB Indices for 6PCs and 18PCs. The silhouette scores and DB Indices shows that both 6PCs and 18PCs deliver clustering

results that have similar qualities. The differences in both the silhouette scores and DB Indices for each K when moving from 6PCs to 18PCs is so small that it cannot be said the one outperforms the other. There does not seem to be a valuable gain in clustering quality for the additional variance explained when moving from 6PCs to 18PCs. Thus, is seems reasonable to discard the use of 18PCs and focus on the results of 6PCs for the remaining sections.

Table 4.1: A summary of silhouette coefficients and DB Indices for 6PCs and 18PCs at different values of K. The brackets next to the principal components indicate the cumulative variance explained.

Principal components	K	Silhouette score	Davies-Bouldin Index
6 (80.98%)	2	0.31	1.22
	6	0.24	1.28
18 (93.12%)	2	0.27	1.22
	6	0.20	1.29

This section has outlined the implementation of the K-means algorithm on 6PCs and 18PCs for a range of values of $K = 2, \ldots, 10$. The elbow method, silhouette score and DB Index were used to determine that $K = 2$ and $K = 6$ for 6PCs seemed to be the best choice for the given initialisation methods. The section to follow examines the spatial distribution of the clusters by visualising the cluster results on a map of Gauteng for $K = 2$ and $K = 6$ using 6PCs.

4.2.5 Visualisation of Clustering

Using the resulting cluster allocations for 6PCs, the spatial distribution of clusters can be investigated by using Figure 4.9 (for $K = 2$) and Figure 4.10 (for $K = 6$). The colours correspond to different cluster labels.

From Figure 4.9, Cluster 1 allocations are mainly situated around the center and top-middle of the province, with a few more wards to the bottom-middle. City centers, namely Pretoria, Centurion and Johannesburg, are found near the center, where the Cluster 1 areas to the top-middle include Soshanguve, Mabopane and Hammanskraal. The few areas belonging to Cluster 1 to the bottom-middle correspond to Vanderbijlpark and Vereeniging. All of these areas consist mainly of built-up areas, i.e. areas where there are buildings of some sort.

From this, it seems that the K-means algorithm generally seems to cluster wards that consist of built-up areas together. Thus, the biggest discrepancy seems to be between urban and rural areas. Recall that the CNN used to extract the feature set was trained on nightlight intensity. It is expected that areas that are more built-up would correspond

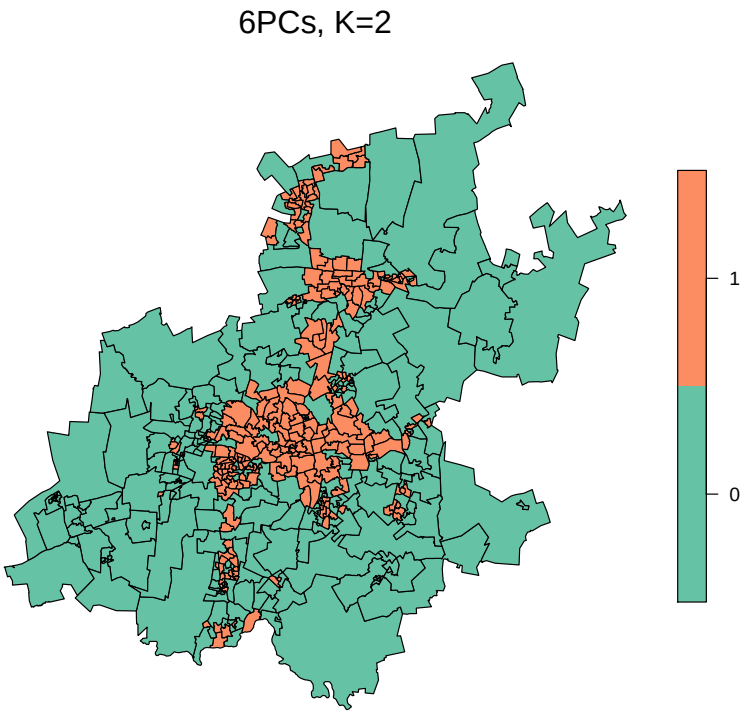

Figure 4.9: Clustering of Gauteng wards using 6PCs and $K = 2$ clusters

with areas with higher nightlight intensities and vice-versa for non-built-up areas. Therefore, the fact that the clustering algorithm is finding the biggest dispersion between these built-up (urban) and non-built-up (rural) areas is hardly surprising.

Figure 4.10 considers the spatial distribution of clusters for $K = 6$ using 6PCs. Again, the rural areas on the border of Gauteng are mostly assigned to the same cluster. In this case, most of the built-up areas mentioned above are now divided between the 5 remaining clusters. From this figure alone it is very difficult to draw inference on whether or not the different clusters in the built-up areas do, in fact, correspond with different levels of poverty.

In general, with unsupervised methods it is very hard to validate whether the algorithm is clustering well. In this context, "clustering well" refers to two things in particular: (1) the quality of the clusters with respect to similarity of wards within a cluster and dissimilarity with wards of a different cluster and (2) whether these similarities or dissimilarities are with regards to poverty or some other, unknown factor.

In the case of this paper, it is clear that K-means is not clustering the wards well[5] with regards to point (1). However, more insight can be drawn from Figure 4.11 that shows the nightlight intensities for Gauteng. The areas with the brightest nightlight intensities

[5]Recall the low silhouette scores and high DB Indices in Table 4.1.

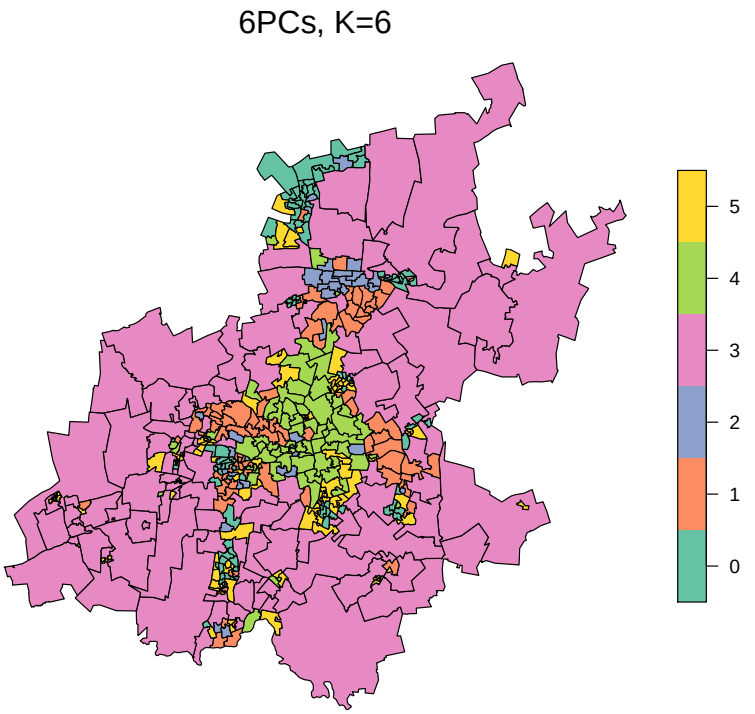

Figure 4.10: Clustering of Gauteng wards using 6PCs and $K = 6$ clusters

roughly correspond to the wards assigned to Cluster 1 in the $K = 2$ case. The Cluster 1 wards were also found to be mostly built-up areas. Similarly, Cluster 0 wards were areas with lowest nightlight intensities.

Thus, it seems possible that the algorithm is simply clustering built-up and non-built-up areas together in each case, since these are areas that exhibit the biggest difference in nightlight intensities. The low silhouette scores and high DB Indices found for these cluster solutions could correspond to the fact that the nightlight intensities vary continuously over the areas, i.e. the intensities gradually increase or decrease and do not change drastically from one ward to the next. This could result in the wards not being well-separated. Whether or not the wards are similar or dissimilar specifically with respect to poverty, as mentioned in point (2) above, is still unclear.

4.2.6 External Validation of K-means

This section critically evaluates the effectiveness of the K-means algorithm in grouping wards into clusters of similar poverty levels.

As mentioned above, the low silhouette scores and high DB Indices obtained when im-

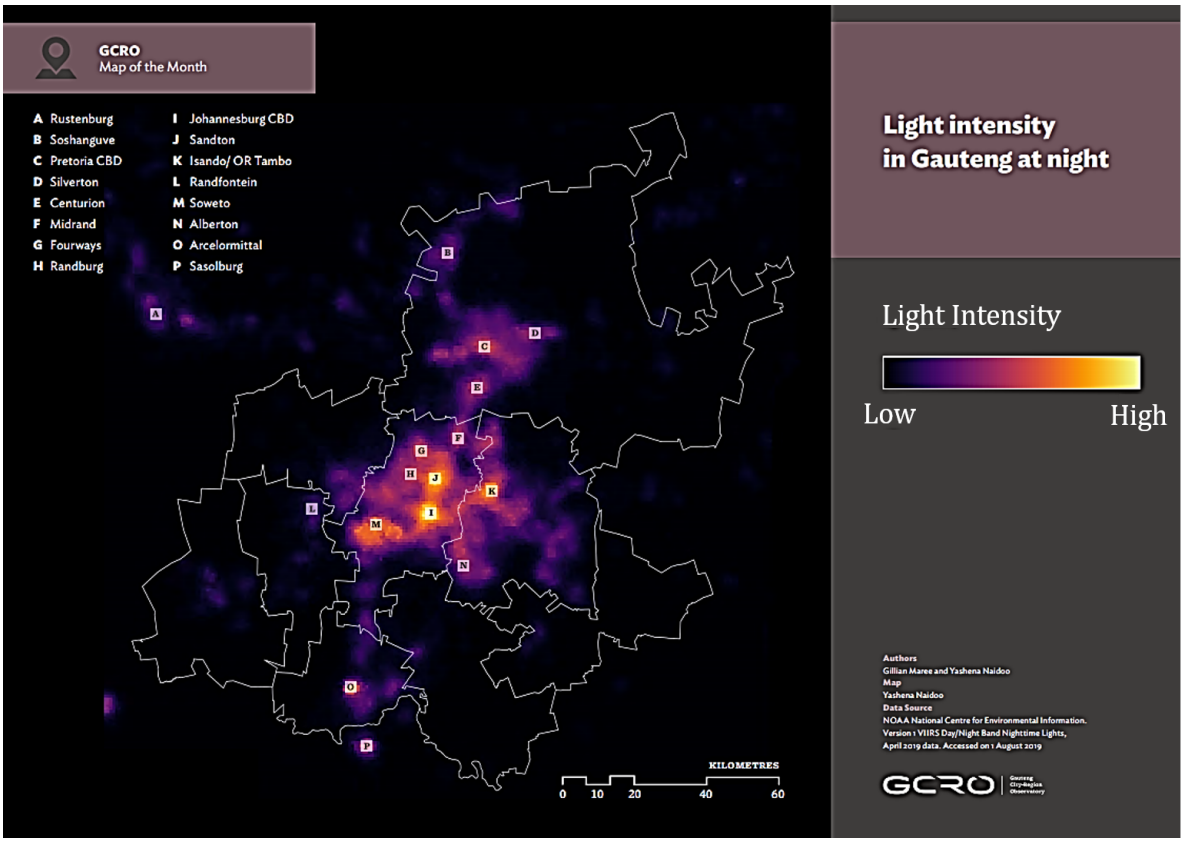

Figure 4.11: Nightlight intensity of Gauteng. The colours range from black, to purple and finally light yellow, with black being the most dim and light yellow the brightest nightlight intensities [6].

plementing K-means makes the validation of the effectiveness of the clustering algorithm tricky. Furthermore, even if a high silhouette score and low DB Index were obtained, it is difficult to determine what the algorithm is clustering. Since the methods here are unsupervised (i.e. true cluster labels are not available), the algorithm simply finds observations most similar to cluster together, but it is not clear yet if these observations are specifically similar with regards to poverty.

In order to investigate this, the 2015 SAMPI values[6] for Gauteng wards was sourced. Even though the 2015 values are a bit outdated, poverty is not something that changes drastically over short periods of time, especially at high levels of aggregation such as ward-levels. Thus the 2015 SAMPI values are assumed to be a relatively accurate measure of current poverty levels.

When investigating the SAMPI data it became clear that some of the wards had missing values. Of the 508 SAMPI values corresponding to the 508 Gauteng wards, 13 values were missing. Since only a few of the values were missing and given the time-constraint of this paper, a crude estimate of the missing SAMPI values seemed appropriate.

[6]SAMPI is the South African Multidimensional Poverty Index. See the Introduction section for a description of how the measure is created.

In order to fill in the missing values, the Inverse Distance Weighted-method of spatial interpolation was used. This method simply replaces a missing value by a weighted average of the surrounding wards' SAMPI values. Wards closer to the ward with a missing value are given more weight than those further away. This relates to Tobler's first law of geography [36] that states: "everything is related to everything else, but near things are more related than distant things".

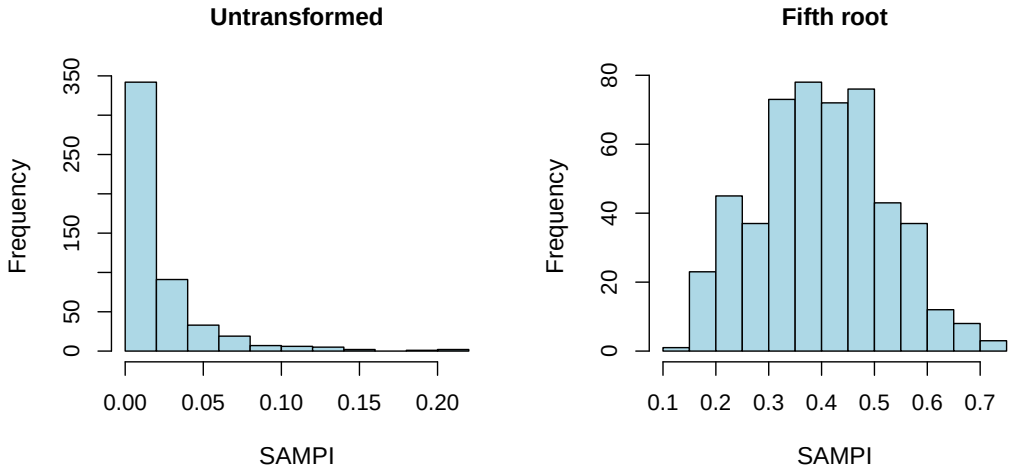

Figure 4.12: Histogram of original SAMPI values (left) and the fifth root of SAMPI values (right).

Finally, the distribution of the SAMPI values was investigated. The histogram to the left of Figure 4.12 shows that the SAMPI values are highly skewed to the right. Having a scewed distribution would cloud subtler nuances when performing the validation analysis, thus a transformation was applied. A few transformation were considered, namely taking the natural logarithm and taking the second, third, fourth and fifth roots. Of these, the best distribution[7] was found to be the fifth root, as shown in the histogram to the right of Figure 4.12. This distribution was the least skewed of all the distributions considered and showed a vast improvement from the untransformed SAMPI values with far fewer outlying values. Note that for the remainder of this section any reference to the SAMPI values refers to the fifth root of the SAMPI values.

Once the missing SAMPI values had been interpolated, an external validation analysis could be conducted. This was done by assigning the SAMPI values of each region to the corresponding cluster label as obtained by the K-means algorithm above and creating boxplots for each cluster label. This is repeated for $K = 2$ and $K = 6$ using 6PCs. These are visualised below in Figure 4.13.

Figure 4.13 shows the boxplots of SAMPI values in the $K = 2$ case (left) and $K = 6$

[7] "Best" here refers to the distribution being centered around its mean with small tails

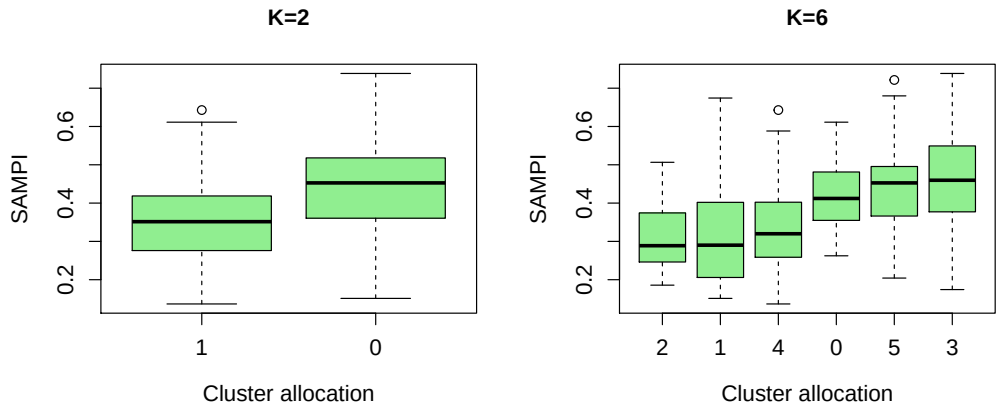

Figure 4.13: Boxplot of SAMPI values for the $K = 2$ case (left) and $K = 6$ case (right) with cluster labels obtained from applying the K-means algorithm to 6PCs.

case (right). In an ideal situation, the boxes[8] for each cluster would not overlap and the within-cluster variation would be very small. This would correspond to observations within a cluster being very similar to one another and very dissimilar to observations in other clusters (indicated by the boxes being disjoint). However, that is not the case here. For $K = 2$, it is noted that although the median value for Cluster 0 is higher than that of Cluster 1, there is, unfortunately, a substantial amount of overlap between the two boxes. This implies that the SAMPI values are not well distinguished between different clusters.

This idea ties back to the concern brought up at the end of the implementation section regarding the low silhouette score and high DB Indices. It is more clear now that the data is not clustering well specifically with regards to poverty levels.

Similarly, for the $K = 6$ case (as seen to the right of Figure 4.13), some of the clusters seem to be more disjoint, for example Cluster 2 and 3. However, there is still a lot of overlap between the different clusters which again means that the SAMPI values between clusters are not well separated.

In conclusion, this section has confirmed that the K-means algorithm to cluster the principal components does not cluster the wards well in terms of poverty levels. The following sections attempt to improve the clustering by considering a density-based clustering algorithm, HDBSCAN.

[8]Including the tails and whiskers

4.3 HDBSCAN

K-means suffers from three problems: it requires the selection of the number of clusters; it partitions the data, and hence assigns noise[9] to clusters; and there is an implicit assumption that clusters have Gaussian distributions [37]. Since the shape of the clusters trying to be identified are not discernible, it seems necessary to consider density-based techniques that can handle non-globular clusters. Two potential candidates are DBSCAN and HDBSCAN. DBSCAN resolves the problems above but suffers from the difficulty of parameter selection [37]. HDBSCAN resolves many of the difficulties in parameter selection by requiring only a small set of intuitive and fairly robust parameters [37].

Thus, in this section, HDBSCAN will be explored further. This will entail describing the HDBSCAN algorithm and how the resulting cluster solutions can be validated, thereafter applying it to the features set. But before the exploration can begin, some terminology needs to be elaborated on.

4.3.1 Introduction

Hierarchical density-based spatial clustering of applications with noise (HDBSCAN) is, as the name suggests, a density-based, hierarchical clustering algorithm developed by Campello, Moulavi and Sander [38]. It extends DBSCAN [39] by converting it into a hierarchical clustering algorithm, and then uses a technique to extract a "flat" (i.e. non-hierarchical) clustering based on the stability of the clusters [38]. The meaning of each of these terms will become clearer as the algorithm is explored further.

Density-based clustering algorithms identify distinctive groups/clusters in the data based on the idea that a cluster in a data space is a connected region of high point density, separated from other such clusters by connected regions of low point density [40]. The data points in the regions of low point density (i.e. the sparse regions) are generally considered noise [40].

A simple 1D example helps to cement this terminology. Consider the histogram and stripplot of the simulated 1D data in Figure 4.14. From the stripplot, it is evident that there are certain intervals in which many observations are in very close proximity (i.e. dense intervals). Furthermore, there are regions between these intervals in which the data points are far less frequent (i.e. sparse intervals). Viewing the empirical distribution of the data allows for more intuition: the dense and sparse regions can be viewed as "mountains"

[9] In a general sense, observations are classified as noise if the algorithm deems them too "different" from any of the proposed clusters.

and “valleys”, respectively, where the mountains are candidates for potential clusters.

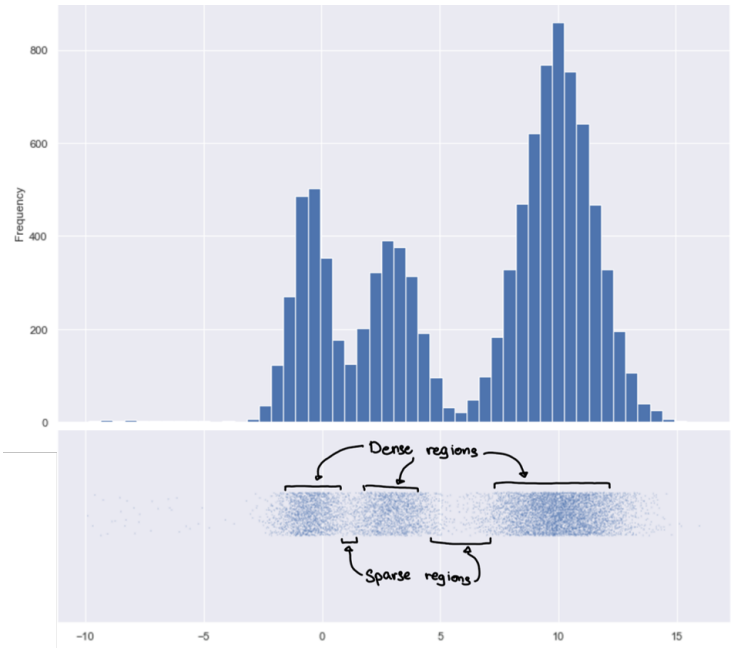

Figure 4.14: Bottom: the stripplot for the simulated 1D data. Top: the empirical distribution of the data (Example sourced from [7]).

From this intuition of density-based clustering, the HDBSCAN algorithm can be outlined.

4.3.2 Algorithm

The HDBSCAN algorithm is considerably more complex than K-means. Describing the algorithm in detail would require a substantial number of definitions before any aspects of the algorithm can be elaborated on. Thus, rather than focusing on the technical details of the algorithm, an emphasis will be placed on providing a high-level overview of the main steps of the algorithm. These steps are as follows:

1. Transform the data space according to the density/sparsity
2. Build the minimum spanning tree (MST) of the distance-weighted graph
3. Construct a cluster hierarchy of connected components
4. Condense the cluster hierarchy based on minimum cluster size
5. Extract the stable clusters from the condensed tree

Each of these will be expanded on further in detail:

Step 1. Transform the data space according to the density/sparsity.

As mentioned previously, density-based algorithms require the identification of dense and sparse regions. In Figure 4.14, the density and sparsity of a point was estimated using a histogram. However, this is only useful for one-dimensional data and becomes computationally intractable as the the number of dimensions increase [8]. Thus, an inexpensive estimate of density needs to be devised.

The simplest method is to consider the distance from a point to its t^{th} nearest neighbour (where the point itself is included in the neighbourhood). For a point $\boldsymbol{x}_i$ and parameter t, this distance is known as the **core distance** and is denoted $\text{core}_t(\boldsymbol{x}_i)$. The core distance of a point with $t = 6$ has been visualised in Figure 4.15. It follows logically that points with low core distances are in high-density regions, and points with high core distances are in sparser regions.

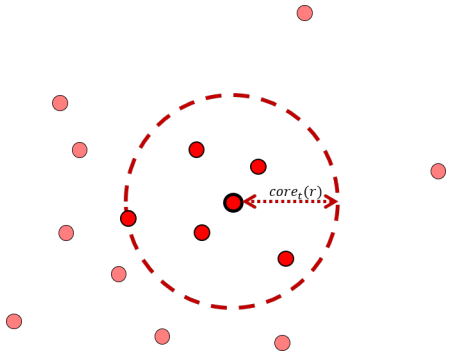

Figure 4.15: An illustration of the core distance of a point, r, with $t = 6$.

From this relationship between the core distance and density, it is evident that the inverse of the core distance can be used to approximate the local density at a point. Thus, observations can be distinguished using a threshold, ϵ, for their core distance or a threshold, $\lambda = \frac{1}{\epsilon}$, for their density.

The algorithm then makes sparse and dense regions more distinguishable by defining a new distance metric between all points, called the **mutual reachability distance**. For points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, this is defined w.r.t. t as

$$d_t^*(\boldsymbol{x}_i, \boldsymbol{x}_j) = \max\{\text{core}_t(\boldsymbol{x}_i), \text{core}_t(\boldsymbol{x}_j), d(\boldsymbol{x}_i, \boldsymbol{x}_j)\}$$

where $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the original metric distance (e.g. euclidean) between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

Under this new metric, points in dense regions remain the same distance from each other but sparser points are pushed away to be at least their core distance away from any other point. This "repelling effect" has been visualised in Figure 4.16, using the euclidean distance to define $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Even though the red and blue points are in very

close proximity, because they are sparse (as seen by their relatively large core distances), the points are pushed apart under the mutual reachability distance.

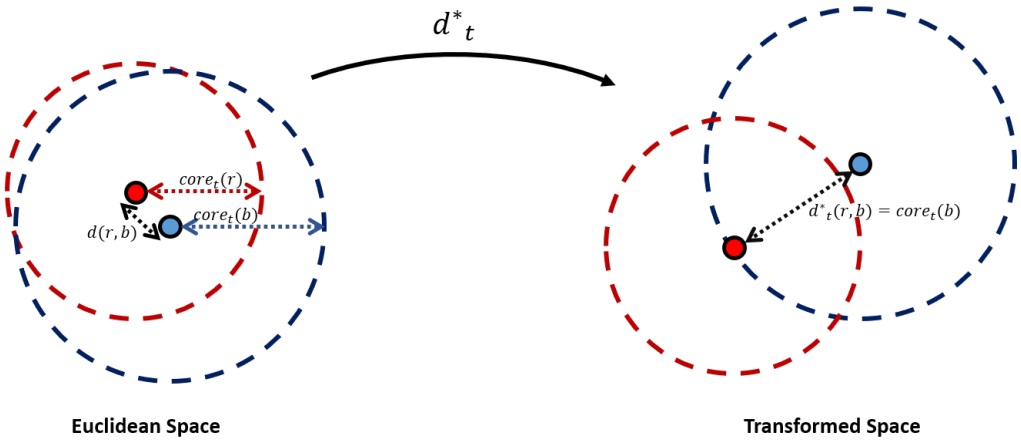

Figure 4.16: The transformation of two points, r (red) and b (blue), from the euclidean space to the transformed space using the mutual reachability distance. The radius of the dashed circles represent the core distance for each point.

Step 2. Build the MST of the distance-weighted graph

Once the points have been embedded into the transformed space, the algorithm then constructs a minimum spanning tree (MST) with the data-points as vertices and an edge between any two points with weight equal to the mutual reachability distance of those points. For intuition, this is equivalent to performing single-linkage agglomerative clustering in the transformed space. Although the implementation makes use of efficient graph theoretical algorithms to obtain the MST (such as Prim's), this explanation will focus on the parallels with single-linkage clustering. Thus, the process of obtaining the MST, and equivalently, the hierarchy of single-linkage clusters can be summarized as follows:

1. Consider each point as a separate cluster
2. Find the shortest distance (under $d^*_t(\cdot,\cdot)$) between two clusters, where the distance between clusters is the shortest distance between any two points in the respective clusters
3. Merge the two clusters
4. Go back to step (2) until there is only one cluster that contains all the points

The question then begs: why not perform single-linkage agglomerative clustering in euclidean space? Performing single-linkage clustering in euclidean space can be sensitive to noise since noisy points can form spurious bridges across clusters [37]. By embedding

the points in the transformed space, the "repelling effect" makes the clustering much more robust to noise [37]. An example of an MST generated in the space of transformed distances can be seen in Figure 4.17. Note that the points have been projected into 2D by t-SNE [41]. Thus, it is not the proximity of the points in the plot that are important, but the mutual reachability of the points (as seen by the coloured edges).

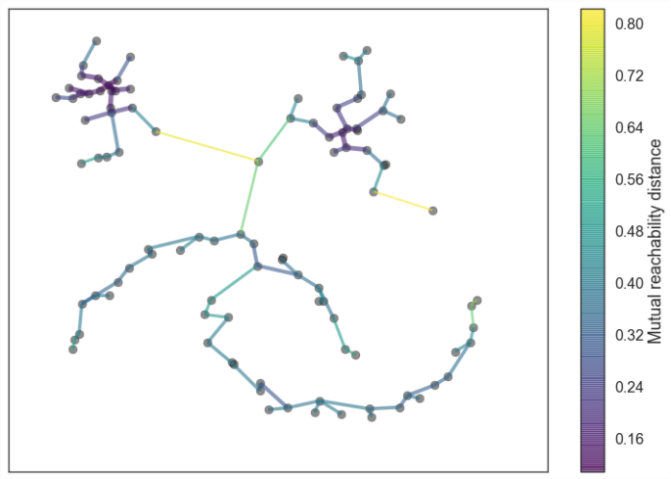

Figure 4.17: An MST generated by the HDBSCAN *scikit-learn-contrib* implementation [8].

Step 3. Construct a cluster hierarchy of connected components

As the previous step of the algorithm was equivalent to performing single-linkage agglomerative clustering on the transformed space, the merging of clusters can be viewed with a dendrogram (see Figure 4.18). It seems logical to then use this dendrogram to obtain "flat" clusters by cutting the dendrogram at a certain mutual reachability threshold, ϵ. In fact, the clusters at level ϵ of this hierarchical clustering are precisely the clusters obtained by DBSCAN*[10] for the parameter choices t and ϵ (declaring any singleton clusters at the cut level with $\text{core}_t(\boldsymbol{x}_i) > \epsilon$ as noise); in this sense a hierarchical DBSCAN* clustering has been obtained [37].

However, HDBSCAN would like to be able to obtain clusters with variable density, and choosing a fixed mutual reachability distance cut-off effectively sets a single, global density level for the resulting clusters.

Thus, the algorithm would ideally need to be able to cut the tree at different places to obtain clusters. Furthermore, as the number of points increases, a dendrogram becomes increasingly difficult to interpret as the plot becomes overloaded with lines. In response to both of the above, the condensed cluster tree is developed.

[10]DBSCAN* is an adaptation of standard DBSCAN which removes the notion of border points.

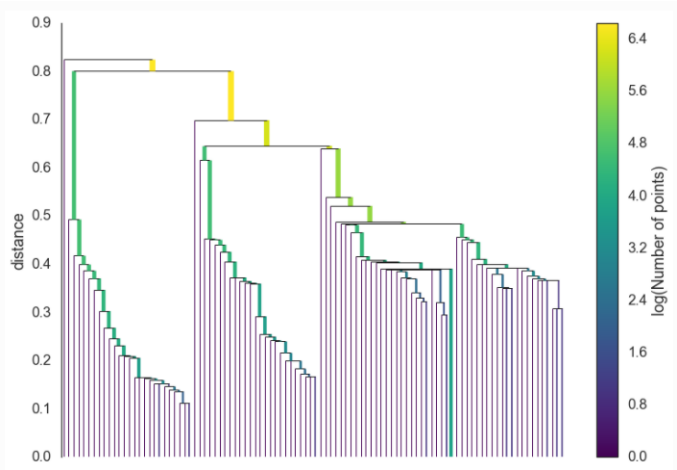

Figure 4.18: A single-linkage dendrogram generated by the HDBSCAN *scikit-learn-contrib* implementation [8]

Step 4. Condense the cluster hierarchy based on minimum cluster size

In order to develop the condensed cluster tree, a new parameter, **minimum cluster size** (m_{size}), needs to be introduced. Starting with all the observations in one cluster, whenever there is a split in the dendrogram above there are four cases that need to be considered:

Case 1: One of the new clusters created by the split has fewer than m_{size} points.
The points in this new cluster are regarded to have "fallen out of the parent cluster" as opposed to started a new one. As a result, they are marked as noise.

Case 2: One of the new clusters created by the split has at least m_{size} points.
This cluster is considered a continuation of the parent cluster, and will be assigned the parent's identity. Thus, the parent cluster will persist further as opposed to the creation of a new cluster.

Case 3: Both clusters have at least m_{size} points.
This is regarded as a "true" cluster split and the two new clusters are allowed to persist in the tree.

Case 4: Both clusters have less than m_{size} points.
The parent cluster is effectively "stopped", with no new clusters being formed. All of the observations in the parent cluster are marked as noise.

Thus, for decreasing mutual reachability thresholds, ϵ, the dendrogram is traversed from top to bottom, with true cluster splits creating new clusters and with some points falling out of the clusters. Equivalently, the y-axis of the dendrogram can be changed to reflect the density level, $\lambda = \frac{1}{\epsilon}$, at which splits and the falling out of points occurs.

Using these ideas, the condensed dendrogram can be constructed. An example of such a dendrogram is displayed in Figure 4.19, where the width of the coloured lines represent the number of points in the cluster. It is evident that there are instances of true cluster splits (e.g. the first split) and instances where points have "fallen out of clusters" (as seen by the diminishing width of the lines).

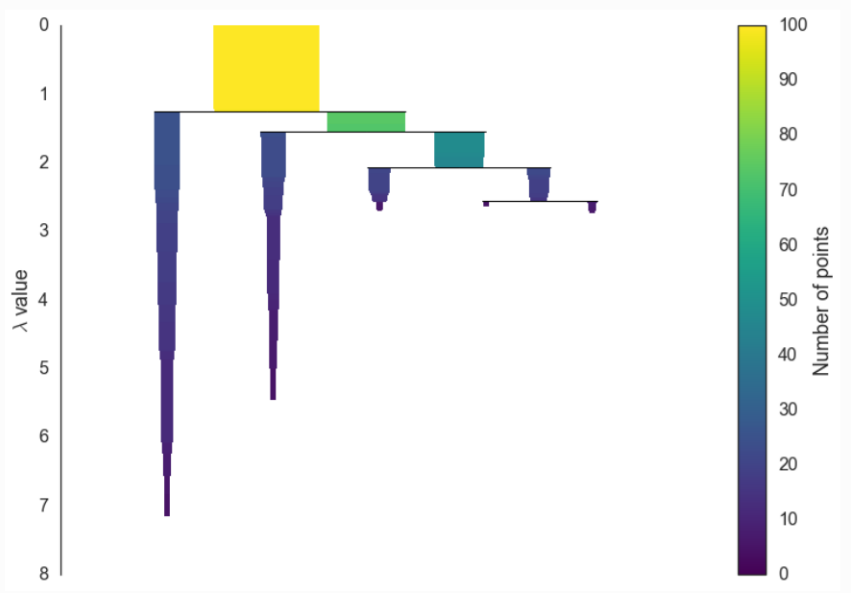

Figure 4.19: A condensed dendrogram generated by the HDBSCAN *scikit-learn-contrib* implementation [8].

Step 5. Extract the stable clusters from the condensed tree

There are two core ideas that govern the selection of clusters from the condensed cluster tree: (1) short-lived clusters are probably merely artifacts of the single-linkage approach, hence clusters that persist for longer-lifetimes are preferable; (2) If a cluster is selected, then a descendant cluster cannot also be selected.

In order to formalize (1), the notion of stability needs to be developed. Given a cluster $\boldsymbol{C}_k$, define $\lambda_{min}(\boldsymbol{C}_k)$ to be the minimum density level at which $\boldsymbol{C}_k$ exists and $\lambda_{max}(\boldsymbol{x}_i, \boldsymbol{C}_k)$ to be the density level beyond which object $\boldsymbol{x}_i$ no longer belongs to cluster $\boldsymbol{C}_k$ (either because the point "fell out" of the cluster, or because the cluster split). The **stability** of cluster $\boldsymbol{C}_k$ is then defined as

$$S(\boldsymbol{C}_k) = \sum_{\boldsymbol{x}_i \in \boldsymbol{C}_k} \left(\lambda_{max}(\boldsymbol{x}_i, \boldsymbol{C}_k) - \lambda_{min}(\boldsymbol{C}_k) \right)$$

Now, the selection of clusters can be formulated as an optimization problem with the objective of maximizing the sum of stabilities of the extracted clusters, subject to the constraint that clusters must not overlap [38].

In order to achieve this, set all leaf nodes from the condensed tree to be the selected clusters. Now, iteratively select clusters from the tree from bottom to top: if the sum of the stabilities of the child clusters is greater than the stability of the cluster, then set the cluster's stability to be the sum of the child stabilities. If, on the other hand, the cluster's stability is greater than the sum of its children then we declare the cluster to be a selected cluster and deselect all its descendants. Once we reach the root node, the current set of selected clusters are the solution. These are the clusters that the HDBSCAN algorithm then returns.

Using the condensed dendrogram in Figure 4.19, the selected clusters have been circled in Figure 4.20. More intuitively, these are the clusters with the largest total ink area in the condensed dendrogram, subject to the descendent constraints.

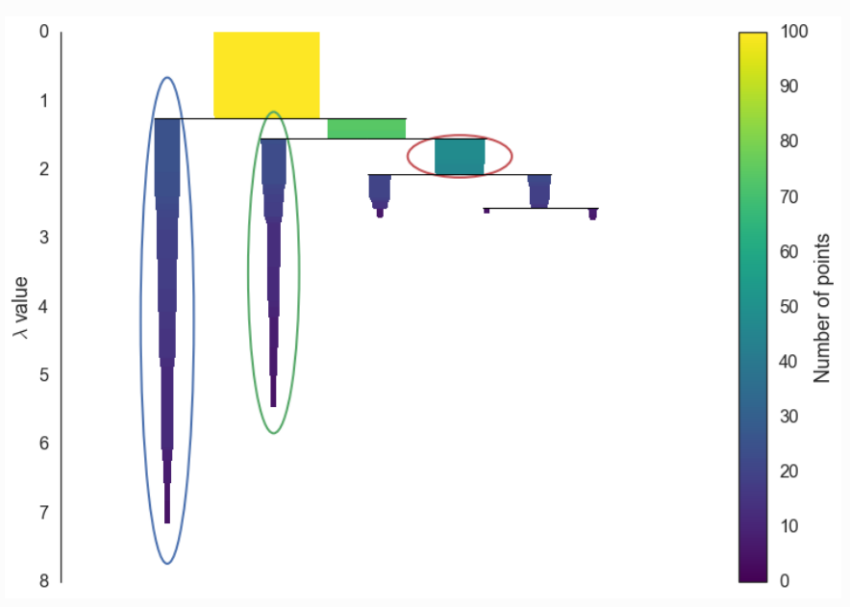

Figure 4.20: A condensed dendrogram (with the selected clusters circled) generated by the HDBSCAN *scikit-learn-contrib* implementation [8].

Thus, given two parameters, t and m_{size}, the HDBSCAN algorithm is able to extract clusters using optimal local cuts through the cluster tree. However, in order to determine the optimal choice of t and m_{size}, an internal method for evaluating the performance of the clusters needs to be obtained.

4.3.3 Evaluating Density-Based Clusters

The silhouette score, inertia and Davies-Bouldin (DB) Index were the internal validation metrics used for K-means. However, these measures are generally not suitable for density-based algorithms since: (1) they have been designed for the evaluation of convex shaped

clusters (e.g. globular clusters) and fail when applied to validate arbitrarily shaped, non-convex clusters; (2) they are also not defined for noise objects [42].

An example of the failings of the silhouette score when applied to non-convex clusters can be seen in Figure 4.21. It appears that HDBSCAN has been able to effectively identify the two most plausible clusters in the data, whereas the K-means algorithm has not been able to do so. However, when referring to the silhouette scores for the clusters obtained, two issues arise:

1. The K-means clusters have a substantially larger silhouette score (0.49 vs 0.17) despite the fact that HDBSCAN clearly performed better.

2. The HDBSCAN silhouette score is very low. A score of 0.17 would generally suggest that the clusters obtained could merely be artefacts of noise in the data. From the visualization in Figure 4.21, it is very clear that this is not the case.

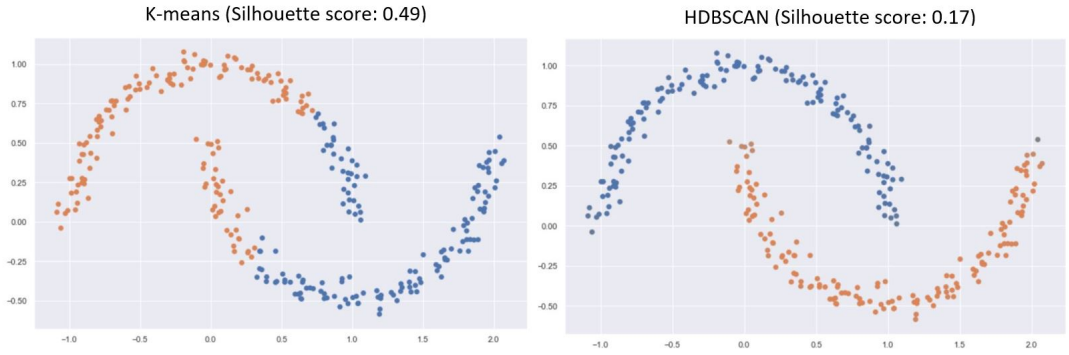

Figure 4.21: The clusters obtained and the associated silhouette score when using K-means (left) and HDBSCAN (right) (Example sourced from [7]).

To overcome the lack of appropriate measures for the validation of density-based clusters, Moulavi et al. (2014) developed a measure called the Density-Based Clustering Validation index (DBCV).

Since density-based clusters are regions of high density separated by regions of low density, a good density-based clustering solution is expected to have clusters in which the lowest-density area inside each cluster is still denser than the highest-density area surrounding clusters [42]. Thus, DBCV defines the quality of clusters obtained based on densities rather than distance [42].

Interestingly, the theory underpinning DBCV is very similar to HDBSCAN - particularly the notion of the core distance of a point, the mutual reachability distance between two points and the use of these mutual reachability distances to develop an MST.

As mentioned previously, for a given point, HDBSCAN uses the inverse of its core distance to estimate the point's density. As this density is based on the distance to a single point (the t^{th} nearest-neighbour), it is not as robust as density estimates that consider more objects from the neighborhood [42]. Furthermore, this definition of density introduces a parameter t, which is not desirable in validation [42]. Thus, DBCV introduces a new definition for the core distance that is more robust, and parameterless - while maintaining its interpretation as the inverse of a density estimate [42].

This new definition of the core distance of a point, $\boldsymbol{x}_i$, belonging to a cluster, $\boldsymbol{C}_k$, w.r.t. the remaining $n_k - 1$ points in $\boldsymbol{C}_k$ is defined as:

$$\text{core}(\boldsymbol{x}_i) = \left(\frac{\sum_{t=2}^{n_k} \left(\frac{1}{\text{core}_t(\boldsymbol{x}_i)} \right)^p}{n_k - 1} \right)^{-\frac{1}{p}}$$

where $\text{core}_t(\boldsymbol{x}_i)$ is the distance from $\boldsymbol{x}_i$ to its t^{th} nearest neighbour in $\boldsymbol{C}_k$ (note the difference from the HDBSCAN definition of $\text{core}_t(\cdot)$) and p is the number of features in the data. The neighbourhood of $\boldsymbol{x}_i$ is defined to include $\boldsymbol{x}_i$. As a result, the 1^{st} nearest neighbour of $\boldsymbol{x}_i$ is itself, and hence its core distance would be zero. Thus, to avoid a zero-division, the summation is over $t = 2$ to n_k. Evidently, $\text{core}(\boldsymbol{x}_i)$ takes into account the distance to all of the other points in $\boldsymbol{C}_k$ to estimate the density of $\boldsymbol{x}_i$, with closer observations contributing more to the density than farther observations.

The mutual reachability distance between any two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is then adapted to use the new core distance definition, such that

$$d^*(\boldsymbol{x}_i, \boldsymbol{x}_j) = \max\{\text{core}(\boldsymbol{x}_i), \text{core}(\boldsymbol{x}_j), d(\boldsymbol{x}_i, \boldsymbol{x}_j)\}$$

where $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the original metric distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

Equivalent to HDBSCAN, an MST is then constructed with the data points as vertices and the edges between any two points with weight equal to the mutual reachability distance of those points. However, the main difference is that these MSTs are constructed per cluster, and not for all observations. Thus, for each of the K clusters an MST is created.

Before the DBCV index can be constructed, two more definitions are needed to capture the before-mentioned statement that "a good density-based clustering solution is expected to have clusters in which the lowest-density area inside each cluster is still denser than the highest-density area surrounding clusters" [42].

The **Density Sparseness of a Cluster** (DSC) $\boldsymbol{C}_k$ is defined as the maximum edge

weight of the internal edges[11] in the MST for $\boldsymbol{C}_k$. Finally, the **Density Separation of a Pair of Clusters** (DSPC) $\boldsymbol{C}_k$ and $\boldsymbol{C}_s$, $k \neq s$, is defined as the minimum reachability distance between the internal nodes[12] of the MSTs for $\boldsymbol{C}_k$ and $\boldsymbol{C}_s$.

Using these definitions, the density-based quality of a cluster, $\boldsymbol{C}_k$ ($k \in \{1, \ldots, K\}$), is given by:

$$V_C(\boldsymbol{C}_k) = \frac{\min\limits_{s \in \{1,\ldots,K\}, s \neq k} \big(DSPC(\boldsymbol{C}_k, \boldsymbol{C}_s)\big) - DSC(\boldsymbol{C}_k)}{\max\left(\min\limits_{s \in \{1,\ldots,K\}, s \neq k} (DSPC(\boldsymbol{C}_k, \boldsymbol{C}_s)), DSC(\boldsymbol{C}_k)\right)}$$

Note that if a cluster has better density compactness (lower DSC) than density separation, the validity index is positive. Alternatively, if the density inside a cluster is lower than the density that separates it from other clusters, the index is negative.

Finally, the DBCV is then defined as a weighted average of the Validity Index for all the clusters:

$$DBCV = \sum_{k=1}^{K} \frac{|\boldsymbol{C}_k|}{N} V_C(\boldsymbol{C}_k)$$

where N is the total number of observations being clustered and $|\boldsymbol{C}_k|$ is the number of observations assigned to cluster $\boldsymbol{C}_k$. The DBCV produces values between -1 and $+1$, with larger values indicating better density-based clustering solutions.

4.3.4 Implementation and Results

Now that both the HDBSCAN algorithm and the method for evaluating the quality of the resulting clusters has been determined, the clustering can proceed. As mentioned previously, the two parameters that need to be determined are t and m_{size}. Unfortunately, because the meaning of the features is unknown (which is amplified further by the use of dimension reduction on these unknown features), the resulting intuition on suitable values for these parameters is limited. Thus, a grid search over a range of plausible parameter values will be undertaken, using the DBCV to determine the optimal parameter combination. However, these "plausible" parameter combinations need to be determined.

The minimum cluster size, m_{size}, should be rather intuitive - what is the minimum size grouping that should be considered a cluster? It seems reasonable to discard clusters that have less than 5 wards. On the other hand, there does not seem to be a clear upper bound that should be considered for m_{size}, thus a somewhat large value of 50 is used.

[11] The internal edges of an MST are all the edges except those with one ending vertex of degree one

[12] The internal nodes are all the objects except those with degree one.

It can be shown that t controls how conservative the clustering should be. As t increases, the clustering becomes more conservative and, as a result, more points are declared as noise [8]. Determining a lower and upper bound for this parameter is again very difficult. By experimentation, it was identified that many observations were being classified as noise - hence, small values of t are more suitable. Thus, an appropriate range for t is deemed to be between 2 and 20.

Thus, for the data comprised of 6PCs and 18PCs, HDBSCAN clusters are going to be obtained for every combination[13] of $m_{size} \in \{1, \ldots, 50\}$ and $t \in \{1, \ldots, 20\}$.

Results: 6PCs

The DBCV scores for each of the parameter combinations when using 6PCs can be seen in Figure 4.22. There are 9 different parameter configurations with a DBCV score above 0.08. When examining the corresponding cluster solutions further, it is apparent that only 5 of them are unique. Despite these being the solutions with the largest DBCV scores, the actual DBCV values attained seem very low (when considering that good density-based cluster solutions have DBCV scores closer to 1). This suggests that the density-based clusters obtained are potentially not very distinguishable.

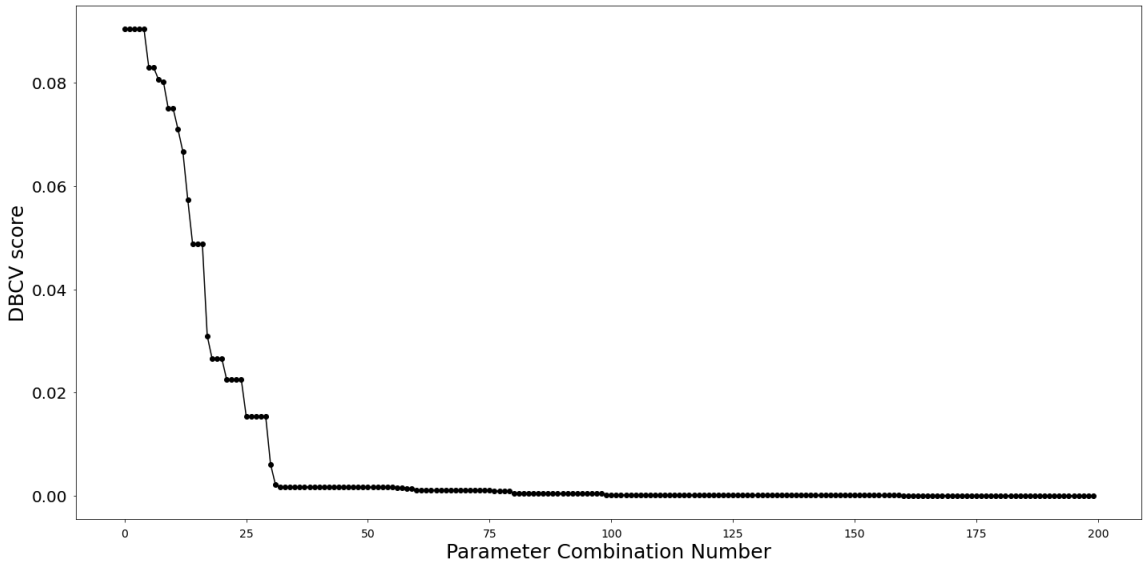

Figure 4.22: The DBCV score for the clusters obtained using the grid search over $m_{size} \in \{1, \ldots, 50\}$ and $t \in \{1, \ldots, 20\}$ when clustering 6PCs. The parameter configurations have been sorted in descending order of DBCV score.

Additional information on these unique cluster solutions with a DBCV score above 0.08 can be seen in Table 4.2. It is concerning that less than half of the wards are assigned

[13] Above a certain threshold of m_{size} and t, the clusters obtained mark all observations as noise. These cluster solutions have not been included in the proceeding plots for the sake of readability.

to clusters in each of the respective "best" solutions. In theory, a method to increase the number of wards assigned to clusters is to consider smaller values of the the parameter t. However, when referring to Figure 4.23, it is evident that the cluster solutions with the largest number of wards assigned to clusters have substantially lower DBCV values than the clusters in Table 4.2. Thus, this approach does not seem suitable. However, if too many wards are marked as noise, then the objective of this paper is defeated.

Table 4.2: The number of clusters, the parameter configuration for m_{size} and t, the number of observations not marked as noise and the DBCV of the best five cluster solutions when clustering 6PCs. The selected cluster solution is highlighted in blue.

Clusters	m_{size}	t	Not Noise	DBCV
8	5/6	3	177	0.090
7	7/8/9	3	171	0.090
8	6/7	4	129	0.083
8	11	2	**241**	0.081
6	12	2	219	0.080

With this in mind, the cluster solution that has the highest number of non-noise observations, while maintaining a relatively large DBCV will be selected. The cluster solution with $m_{size} = 11$ and $t = 2$ has 241 non-noise wards which is 22 more than any of the other "optimal" solutions identified. Thus, this solution will be investigated further, despite it not having the largest DBCV value. It must be noted that having only 241 wards assigned to the clusters and the extremely low DBCV value is still problematic. Thus, this selected solution will only be investigated further if the optimal 18PCs cluster solution is worse.

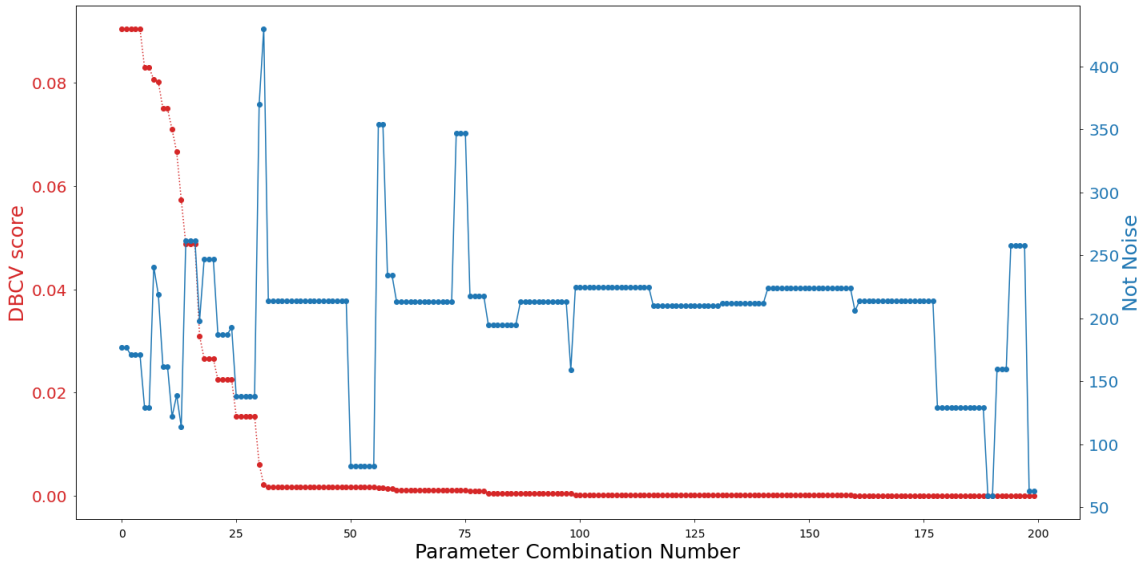

Figure 4.23: Left axis: the DBCV score for the clusters obtained using the grid search when clustering 6PCs. Right axis: the number of wards assigned to clusters.

Results: 18PCs

The DBCV scores for each of the parameter combinations when using 18PCs can be seen in Figure 4.24. Evidently, there are four parameter combinations that achieved substantially higher DBCV scores than the others (of which two are unique). Due to the relatively low DBCV values obtained by the other combinations, they will not be investigated further. Additional information pertaining to the top two cluster solutions can be seen in Table 4.3.

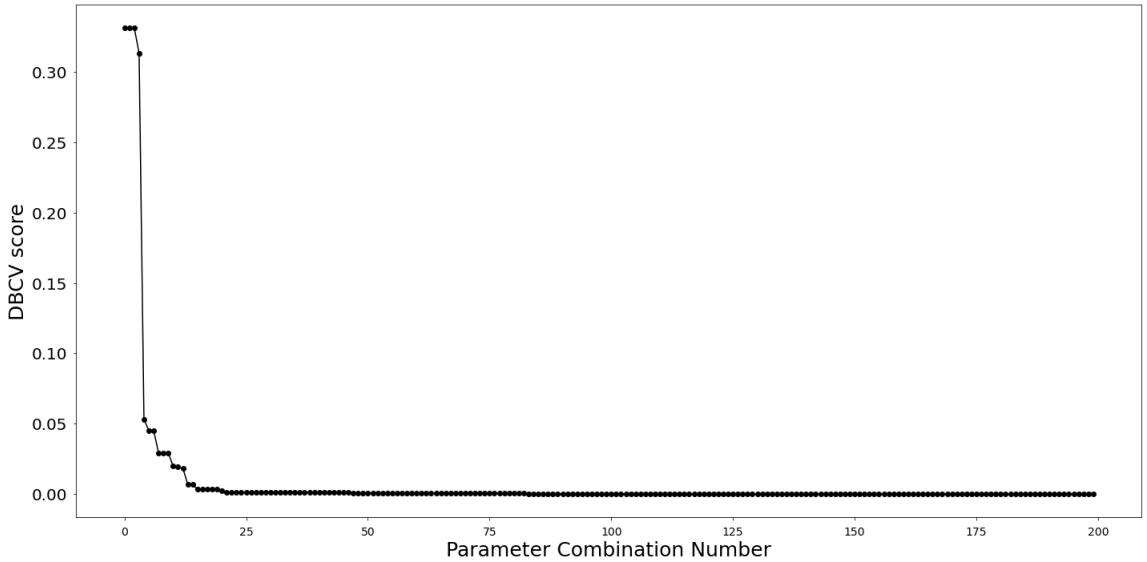

Figure 4.24: The DBCV score for the clusters obtained using the grid search over $m_{size} \in \{1, \ldots, 50\}$ and $t \in \{1, \ldots, 20\}$ when clustering 18PCs. The parameter configurations have been sorted in descending order of DBCV score.

Table 4.3: The number of clusters, the parameter configuration for m_{size} and t, the number of observations not marked as noise and the DBCV of the best two cluster solutions when clustering 18PCs. The selected cluster solution is highlighted in blue.

Clusters	m_{size}	t	Not Noise	DBCV
4	5	5	210	0.313
4	7/8/9	4	**233**	0.331

From Table 4.3, it is evident that the DBCV scores obtained by these clustering solutions are substantially higher than the solutions on 6PCs. However, it is again very concerning that, despite these being the best cluster solutions, the number of observations that were assigned to clusters (and not marked as noise) was only 233 and 210, respectively. This means that more than half of the wards were not assigned to a cluster. Examining Figure 4.25, it appears that there are solutions with 300 and more non-noise observations. However, once again, these solutions have a significantly lower DBCV value. Thus, as before, the cluster solution that has the highest number of non-noise observations, while

maintaining a relatively large DBCV will be selected. This appears to be the solution corresponding to $m_{size} = 7/8/9$ and $t = 4$, where there are 233 wards assigned to clusters.

It is evident that both selected clustering solutions for 6PCs and 18PCs marked a concerning amount of wards as noise. However, the DBCV for the 18PCs solutions was substantially higher than the DBCV for the 6PCs solution. Subsequently, due to its extremely low DBCV value, the 6PCs solution will be discarded.

Thus, despite the large number of wards classified as noise in the 18PCs solution, there might still be some information that can be derived through examining the SAMPI within each of these clusters (and "noisy" observations) and investigating the spatial distribution of these clusters (and "noisy" observations). Thus, despite its limitations, this cluster solution will be investigated further.

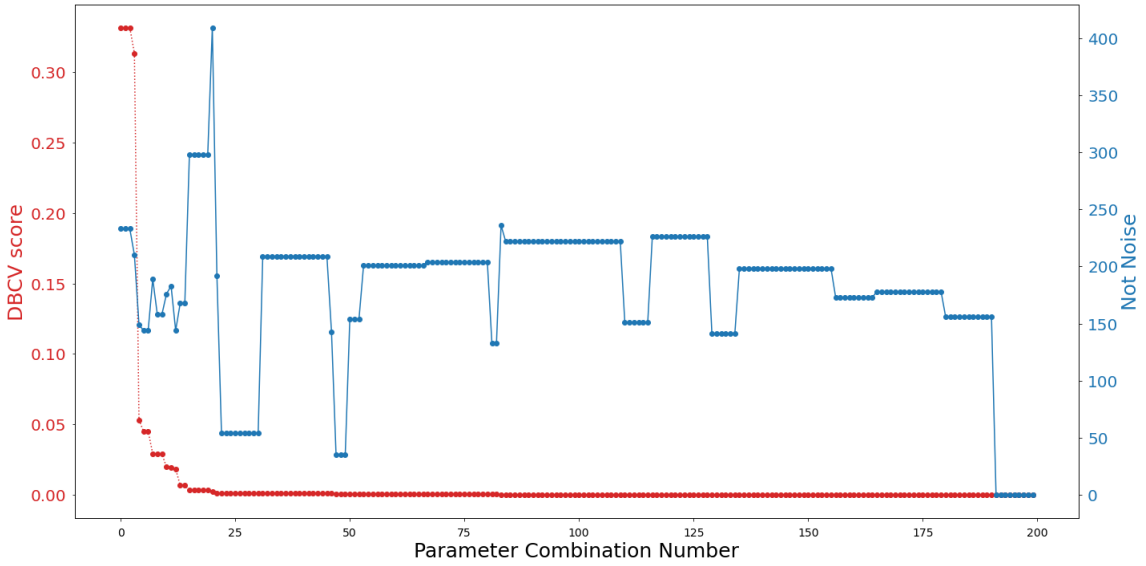

Figure 4.25: Left axis: the DBCV score for the clusters obtained using the grid search when clustering 18PCs. Right axis: the number of wards assigned to clusters.

The compositions of the selected cluster solutions for 18PCs can be seen in Table 4.4. Evidently, Cluster 1 contains most of the observations. The remainder of the clusters in each solution have less than 20 observations. Without further information, it is very hard to determine whether these smaller clusters are finding "random" groupings in the data or whether these small clusters of wards are in fact very similar.

Table 4.4: The number of wards in each cluster in the selected cluster solution for 6 and 18 principal components. C_{noise} is in fact not a cluster but a grouping of all of the wards marked as noise.

C_0	C_1	C_2	C_3	C_{noise}
10	203	10	10	275

Now that an optimal cluster solution has been obtained (i.e the selected cluster solution

for 18PCs), it will be investigated further by examining the spatial distribution of the clusters, and the distribution of the SAMPI values within each cluster.

4.3.5 Visualization of Clusters

The spatial distribution of the 4 clusters when using 18PCs is displayed in Figure 4.26. It appears that the center of the map exhibits the largest number of different clusters. This roughly corresponds to Johannesburg. This could be a result of the variations in building sizes, presence or absence of large roads, extent of greenery etc. in the Johannesburg wards, but this is very difficult to say for certain.

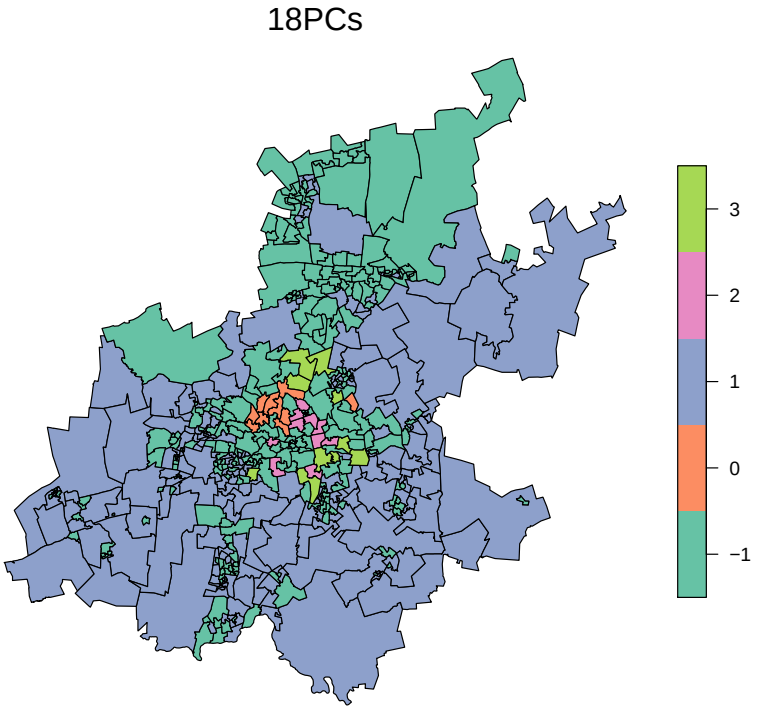

Figure 4.26: The Gauteng wards coloured by cluster using the selected HDBSCAN cluster solution for 18PCs. Cluster -1 corresponds to $\boldsymbol{C}_{noise}$.

The wards in $\boldsymbol{C}_{noise}$ seem to be randomly scattered between the rural (low nightlight intensity) and urban (high nightlight intensity) areas, which provides little insight into what characterizes the "noisy" wards. However, in the areas with low nightlight intensity, there seems to be a trend that the "noisy" wards are those that are significantly built-up (i.e. the built-up areas cover a large proportion of the ward). An example of such a ward can be seen in Figure 4.27.

This might suggest that the algorithm is able to effectively group the wards with little-

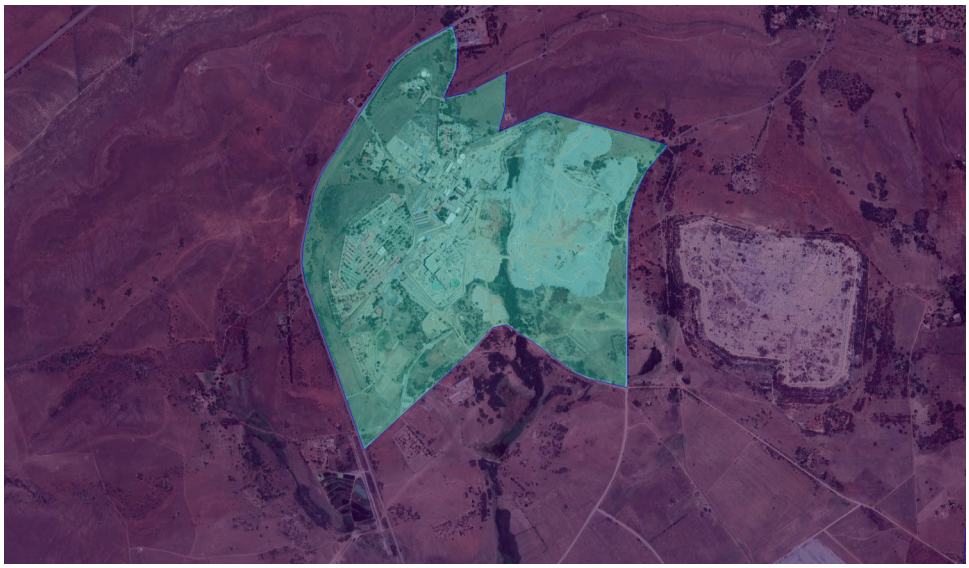

Figure 4.27: A significantly built-up ward (green) in an area where the surrounding wards (purple) are rural. The green ward was classified as noise, whereas the purple wards were not.

to-no built-up areas, but struggles more when distinguishing wards with larger built-up areas. Potentially, the presence of built-up areas might be increasing the variability in the extracted features (as there are more roads, buildings etc. that might be identified by the CNN), making it harder for HDBSCAN to find similar wards. Alternatively, the wards that are primarily rural might have similar features due to the absence of built-up areas, making them easier to cluster together. This is purely speculative, and far more rigorous testing is required to confirm this.

In and around Johannesburg (the center of the map), it appears that there is some evidence of spatial clustering, where 9 of the orange wards and 7 of the pink wards are neighbours (either directly or transitively). It seems reasonable to expect there to be a certain degree of spatial clustering as there are certainly regions where the wards would have similar poverty levels. For example, the poverty levels would be expected to be higher around some of the larger townships, such as Soweto, and lower around affluent business hubs, such as Sandton.

Interestingly, it appears that the purple wards ($\boldsymbol{C}_1$) roughly correspond to the non-central, rural areas (or the areas with low nightlight intensity, as seen in Figure 4.11). The grouping of these rural areas was also seen in the K-means cluster solutions. This gives more support for the idea that the wards are clustered by nightlight intensity as opposed to poverty. However, it would be a stretch to infer anything more about this since many of the wards with low nightlight intensity were also marked as noise.

The information that can be gained from the map pertaining to the ability of the clusters to discern wards based on their level of poverty is very limited. Thus, the distribution of

the SAMPI values within each cluster will be used for external validation of the cluster solutions.

4.3.6 External Validation of HDBSCAN

The boxplots for the SAMPI values within the 18PCs cluster solution can be seen in Figure 4.28. Evidently, $\boldsymbol{C}_1$ has a very narrow distribution of SAMPI values within it. Although it only contains 10 wards, it seems to have identified wards with very similar (small) SAMPI values. Referring back to Figure 4.26, these wards are found in a small region in the center of Gauteng. Upon further investigation, these wards are in close proximity to Sandton, an extremely wealthy area of Johannesburg - explaining the low SAMPI values.

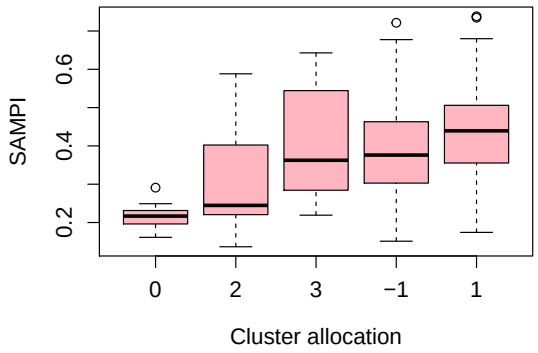

Figure 4.28: Boxplots of the SAMPI values for the HDBSCAN cluster solutions obtained for 18PCs. Cluster -1 corresponds to $\boldsymbol{C}_{noise}$.

It appears that $\boldsymbol{C}_0$ and $\boldsymbol{C}_2$ have a lower median SAMPI value than $\boldsymbol{C}_1$ and $\boldsymbol{C}_3$. This gives some support that the algorithm was able to segregate the wards with low and high SAMPI values. However, due to the long, overlapping tails of most of the clusters, this support is very minimal. It also appears that the SAMPI values for the wards classified as noise effectively spans the whole SAMPI domain. Thus, there does not seem to be any obvious pattern that characterizes these "noisy" wards.

Unfortunately, it seems rather conclusive that the HDBSCAN cluster solutions were not able to effectively distinguish the wards based on their level of poverty (as proxied by the SAMPI). Thus, the hypothesis that a density-based method would improve on the unsatisfactory K-means results, due to its ability to identify non-convex clusters, seems to have been proven incorrect.

Chapter 5

Discussion

It is evident that the K-means and HDBSCAN cluster solutions do not effectively distinguish the wards based on their level of poverty (as proxied by the SAMPI). For both algorithms, the SAMPI boxplots showed not only that most of the clusters had a wide spread of SAMPI values, but that the SAMPI values between clusters also overlapped substantially. This is equivalent to a lack of similarity between the wards within a cluster, and dissimilarity with other clusters.

Instead, the clusters seemed far more effective at identifying wards with similar nightlight intensity. In particular, it appears that the clusters were able to distinguish between areas with very low nightlight intensity, and the remainder of the wards. This was identified by comparing the coloured map of Gauteng with the map of the light intensity of Gauteng at night in Figure 4.11.

This seems to suggest that the extracted features are far more associated with nightlight intensity than poverty. This is not surprising, since the CNN was trained to predict nightlight intensity and not poverty. However, this falsifies the initial hypothesis that these features (or some lower dimensional approximation of them) could be used to cluster the wards according to poverty levels.

As mentioned in the *Related Work: Poverty Estimation* section, nightlights have been shown to be an imperfect proxy for poverty. Thus, it is not surprising that the apparent effectiveness of the clustering algorithms to identify nightlights-based clusters does not correspond to an effective poverty-based clustering. To explore this further, the SAMPI values and nightlight intensities for Gauteng are plotted side-by-side in Figure 5.1.

It is evident that the areas with the most "wealthy" (lowest SAMPI) wards correspond to the regions with the highest nightlight intensity. Alternatively, the poorest (highest SAMPI) wards are generally situated in regions with low nightlight intensity. This would suggest that nightlight intensity is a reasonable proxy for poverty among the wealthiest and poorest wards. However, when examining Figure 5.1, various inconsistencies between the SAMPI values and nightlight intensities can be found: (1) it is evident that in areas with little-to-no nightlights, the SAMPI values among the corresponding wards differ, despite them having similar (very low) nightlight intensities; (2) the wards in two of Gauteng's townships, namely Soweto (M) and Shoshanguve (B), have very high and moderate nightlight intensities, respectively, despite containing wards with high SAMPI values.

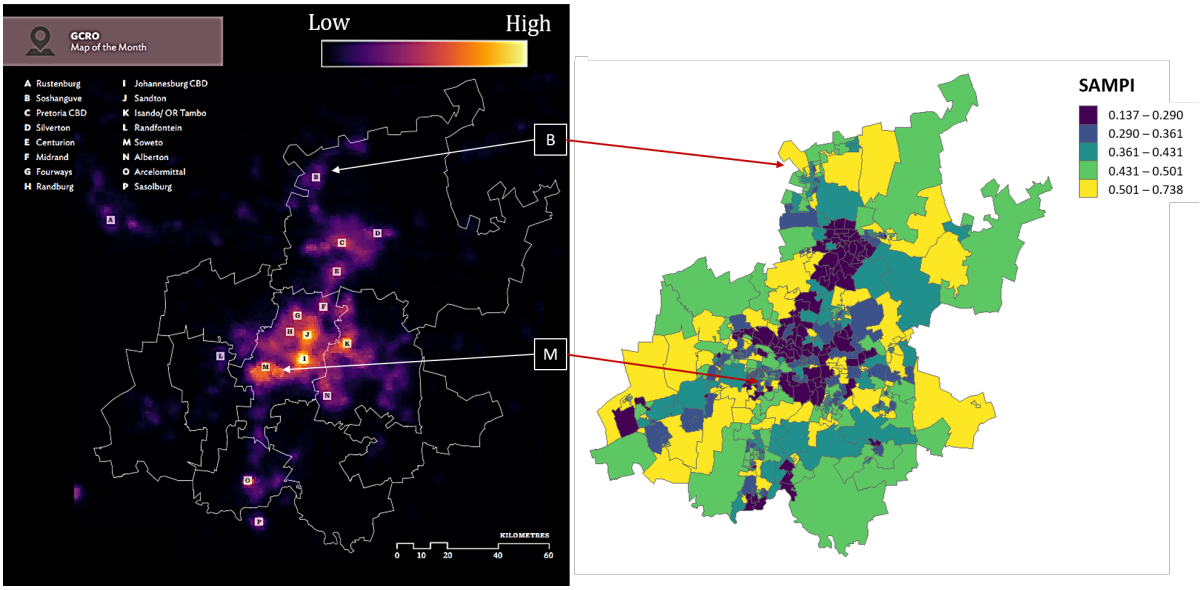

Figure 5.1: The nightlight intensities (left) [6] and SAMPI values (right) for Gauteng. The letters B and M show where Shoshanguve and Soweto are, respectively.

Thus, given the inconsistencies in the relationship between the ward-level SAMPI values and nightlight intensities, a cluster solution that effectively clusters the wards based on their nightlight intensity cannot cluster their SAMPI values effectively too. Hence, the extracted feature set is inadequate when attempting to create poverty-based clusters.

However, this does not mean that the features are entirely unrelated to poverty. There could exist a more complex linear or non-linear mapping between the features and poverty that the clustering algorithms cannot identify. This is due to the nature of clustering algorithms implemented in that they use the proximity of the wards' features and do not use combinations of the features, unlike regression models.

In fact, it has been shown by Jean et al. [1] that a ridge regression model trained on a similar[1] feature set can explain more than half of the variation in indicators of poverty in selected areas of African countries (as outlined in the *Related Work: Poverty Estimation* section). Thus, for the purposes of mapping poverty in Gauteng, a regression model might be more suitable.

It is important to note the apparent contradiction between the poor internal validation scores obtained for the clustering solutions and the notion that the wards are clustered "well" with regards to nightlight intensity. Particularly for K-means when $K = 2$ and 6PCs are used, when comparing Figure 4.9 and Figure 4.11, there seems to be an uncanny correlation between the wards that are assigned to Cluster 0 and Cluster 1, and the areas of low and high nightlight intensities, respectively. Despite this visual evidence, the silhouette score is only 0.31 and the DB Index is 1.22 - which are far from the optimal

[1]They also used a CNN trained on nightlight intensity as a feature extractor.

values of 1 and 0, respectively.

A possible explanation for the low silhouette scores and high DB Indices found could be that the light intensities vary continuously over the areas, i.e. the intensities gradually increase or decrease and do not change drastically from one ward to the next. Thus, it would be expected that the clusters would not be far apart in the data space[2], and hence, not easily separable. However, due to the high-dimensionality of the data, and the resulting inability to visualise the clusters[3] (in the data space), it is very hard to confirm this.

Another possible reason for the mediocre internal validation metrics is the use of PCA. PCA is a linear dimension-reduction technique in that it reduces the dimensions by taking *linear* combinations of the features. However, it may fail when the features have non-linear relationships or, put alternatively, lie on a non-linear manifold. In these cases, a non-linear dimension-reduction technique is more suited. It has been shown that non-linear techniques can improve the quality of the cluster solutions obtained when using HDBSCAN and K-means [43]. However, it is uncertain whether the potential improvements would lead to the clusters merely being able to better distinguish areas with differing nightlight intensities, as opposed to poverty.

[2]The data space refers to the space in which the data is found. For example, a 2D data space can be visualised with a scatterplot.

[3]For example, if the data were 2D, a scatter plot of the data could be used to visualise the separability of the clusters. However, even when using only 6PCs, a graphical visualization is not possible (unless a further dimension reduction technique is applied).

Chapter 6

Conclusions

This paper has explored the application of machine-learning techniques to satellite imagery in an attempt to create a ward-level poverty-map for Gauteng, without the use of survey data. However, the cluster solutions from K-means and HDBSCAN used to achieve this showed unsatisfactory results.

It was evident that the clusters performed poorly when distinguishing the wards based on their level of poverty. This was seen through the large variance and substantial overlap of the SAMPI values within each cluster. Instead, the clusters appeared to be far more effective at identifying wards with similar nightlight intensity. This seems to suggest that the extracted features are substantially more associated with variations in nightlight intensity than poverty. However, due to the mediocre internal validation scores for the cluster solutions, this requires further investigation. As a result, the creation of an accurate poverty map from the cluster solutions is not possible.

However, there is no suggestion that the features are entirely unrelated to poverty. There could exist a more complex linear or non-linear mapping between the features and poverty that the clustering algorithms cannot identify. Thus, a natural extension of the paper is to fit a regression model to either the full feature set, or a set of reduced features to examine the predictive power of the features.

Chapter 7

Recommendations for Future Work

Improving clustering: different dimension reduction.

Improving poverty estimation: regression model.

Bibliography

[1] N. Jean, M. Burke, M. Xie, W. M. Davis, D. B. Lobell, and S. Ermon, "Combining satellite imagery and machine learning to predict poverty," *Science*, vol. 353, no. 6301, pp. 790–794, 2016. [Online]. Available: https://science.sciencemag.org/content/353/6301/790

[2] M. ul Hassan, "Vgg16 – convolutional network for classification and detection," 2018. [Online]. Available: https://neurohive.io/en/popular-networks/vgg16/

[3] R. Girshick, "Fast r-cnn," 2015. [Online]. Available: https://dl.dropboxusercontent.com/s/vlyrkgd8nz8gy5l/fast-rcnn.pdf?dl=0

[4] "Clustering methods for large molecular library screening." [Online]. Available: https://new.pharmacelera.com/science/clustering-methods-big-library-screening/

[5] K. Orkphol and W. Yang, "Sentiment analysis on microblogging with k-means clustering and artificial bee colony," *Int. J. Comput. Intell. Appl.*, vol. 18, pp. 1 950 017:1–1 950 017:22, 2019.

[6] GCRO, "Gauteng at night: light intensity at 2am," https://www.gcro.ac.za/outputs/map-of-the-month/detail/gauteng-night-light-intensity-2am/, 2019.

[7] P. Berba, "Understanding density-based clustering," Feb 2020. [Online]. Available: https://www.kdnuggets.com/

[8] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical density based clustering," *The Journal of Open Source Software*, vol. 2, no. 11, p. 205, 2017.

[9] I. M. Fund, *SOUTH AFRICA*, ser. STAFF REPORT FOR THE 2019 ARTICLE IV CONSULTATION. IMF, 2019. [Online]. Available: https://www.imf.org/~/media/Files/Publications/CR/2020/English/1ZAFEA2020001.ashx

[10] S. S. Africa, "The south african mpi," 2014. [Online]. Available: http://www.statssa.gov.za/publications/Report-03-10-08/Report-03-10-082014.pdf

[11] J. V. Henderson, A. Storeygard, and D. N. Weil, "Measuring economic growth from outer space," *American Economic Review*, vol. 102, no. 2, pp. 994–1028, April 2012. [Online]. Available: https://www.aeaweb.org/articles?id=10.1257/aer.102.2.994

[12] X. Chen and W. D. Nordhaus, "Using luminosity data as a proxy for economic statistics," *Proceedings of the National Academy of Sciences*, vol. 108, no. 21, pp. 8589–8594, 2011. [Online]. Available: https://www.pnas.org/content/108/21/8589

[13] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" 2014.

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[15] S. Piaggesi, L. Gauvin, M. Tizzoni, C. Cattuto, N. Adler, S. Verhulst, A. Young, R. Price, L. Ferres, and A. Panisson, "Predicting city poverty using satellite imagery," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[16] J. Mathur. predicting-poverty-replication. [Online]. Available: https://github.com/jmather625/predicting-poverty-replication

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[18] C. C. Olisah and L. Smith, "Understanding unconventional preprocessors in deep convolutional neural networks for face identification," *SN Applied Sciences*, vol. 1, no. 11, p. 1511, Oct 2019. [Online]. Available: https://doi.org/10.1007/s42452-019-1538-5

[19] R. Girshick, "Fast r-cnn," 2015.

[20] J. Uijlings, K. Sande, T. Gevers, and A. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, pp. 154–171, 09 2013.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 346–361.

[22] J. Dai, K. He, and J. Sun, "Convolutional feature masking for joint object and stuff segmentation," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2015. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2015.7299025

[23] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[24] F. Hu, G.-S. Xia, J. Hu, and L. Zhang, "Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery," *Remote Sensing*, vol. 7, no. 11, p. 14680–14707, Nov 2015. [Online]. Available: http://dx.doi.org/10.3390/rs71114680

[25] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," 2013.

[26] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *Database Theory — ICDT 2001*, J. Van den Bussche and V. Vianu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 420–434.

[27] R. Johnson and D. Wichern, *Applied multivariate statistical analysis*, 5th ed. Upper Saddle River, NJ: Prentice Hall, 2002. [Online]. Available: http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+330798693&sourceid=fbw_bibsonomy

[28] C. Ding and X. He, "K-means clustering via principal component analysis," in *ICML '04: Proceedings of the twenty-first international conference on Machine learning*. New York, NY, USA: ACM, 2004, p. 29.

[29] M. E. Celebi, H. A. Kingravi, and P. A. Vela, "A comparative study of efficient initialization methods for the k-means clustering algorithm," *Expert Systems with Applications*, vol. 40, no. 1, pp. 200 – 210, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417412008767

[30] D. Arthur and S. Vassilsitskii, "k-means++: the advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, N. Bansal, K. Pruhs, and C. Stein, Eds. SIAM, 2007, pp. 1027–1035. [Online]. Available: http://dl.acm.org/citation.cfm?id=1283383.1283494

[31] E. Forgy, "Cluster analysis of multivariate data: Efficiency versus interpretability of classification," *Biometrics*, vol. 21, no. 3, pp. 768–769, 1965.

[32] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, L. M. L. Cam and J. Neyman, Eds., vol. 1. University of California Press, 1967, pp. 281–297.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[34] P. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *J. Comput. Appl. Math.*, vol. 20, no. 1, p. 53–65, Nov. 1987. [Online]. Available: https://doi.org/10.1016/0377-0427(87)90125-7

[35] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, 1979.

[36] W. R. Tobler, "A computer movie simulating urban growth in the detroit region," *Economic Geography*, vol. 46, no. sup1, pp. 234–240, 1970. [Online]. Available: https://www.tandfonline.com/doi/abs/10.2307/143141

[37] L. McInnes and J. Healy, "Accelerated hierarchical density based clustering," *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, Nov 2017. [Online]. Available: http://dx.doi.org/10.1109/ICDMW.2017.12

[38] R. J. G. B. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Advances in Knowledge Discovery and Data Mining*, J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 160–172.

[39] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD'96. AAAI Press, 1996, p. 226–231.

[40] J. Sander, *Density-Based Clustering*. Boston, MA: Springer US, 2010, pp. 270–273. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_211

[41] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008. [Online]. Available: http://www.jmlr.org/papers/v9/vandermaaten08a.html

[42] D. Moulavi, P. A Jaskowiak, R. Campello, A. Zimek, and J. Sander, "Density-based clustering validation," in *Proceedings of the 14th SIAM International Conference on Data Mining (SDM), Philadelphia, PA, 2014*, 2014.

[43] M. Allaoui, M. L. Kherfi, and A. Cheriet, "Considerably improving clustering algorithms using umap dimensionality reduction technique: A comparative study," in *Image and Signal Processing*, A. El Moataz, D. Mammass, A. Mansouri, and F. Nouboud, Eds. Cham: Springer International Publishing, 2020, pp. 317–325.

[44] Google. Map and tile coordinates. [Online]. Available: https://developers.google.com/maps/documentation/javascript/coordinates

Appendix A

Image Extraction Algorithm

A.1 Explanation of the Image-Extraction Algorithm

Before any clustering and predictive models can be trained, a set of features for each ward in the respective Gauteng municipalities needs to be extracted. The process to develop this feature set can be divided up as follows: (1) image location determination, (2) image extraction, (3) image cropping and mask creation, and (4) feature extraction. This appendix chapter describes each of the respective steps.

A.2 Image Location Determination

In the first stage of the feature extraction process, the "unadjusted" centers for the images needed to cover the wards are obtained. The reason that they are termed "unadjusted" centers is that they will not correspond to the actual, "adjusted" centers of the satellite images used, but are rather used to simplify the image-location-determination algorithm. A minor adjustment or mapping can then be used to convert from the "unadjusted" to "adjusted" centers when actually extracting the satellite images. However, this will become clearer when it is elaborated on further below.

For each municipality and each ward within the municipality, the following procedure is used to extract the before-mentioned "unadjusted" centers:

1. Use the municipality's shapefile to extract the coordinates for the ward. See Figure A.1.
2. Place a bounding box around the ward. This is done by first calculating the maximum and minimum latitude and longitude for the ward. These values are then used as the corners of the box. Referring to Figure A.2, the reason that the bottom and left side of the bounding box are dashed is that they will form as a stopping criteria for the image location determination algorithm. This means that the coverage of images sampled from the bounding boxes can exceed these boundaries. The same is not true for the top and left boundaries.
3. Before being able to sample images from this box, the area of coverage of a single Google Static Maps satellite image relative to the bounding box needs to be deter-

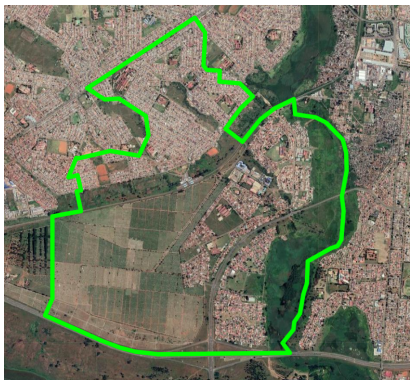

Figure A.1: A visualization of step 1 of the algorithm. A ward is overlaid in green to demonstrate the use of the shapefile to obtain the ward boundaries.

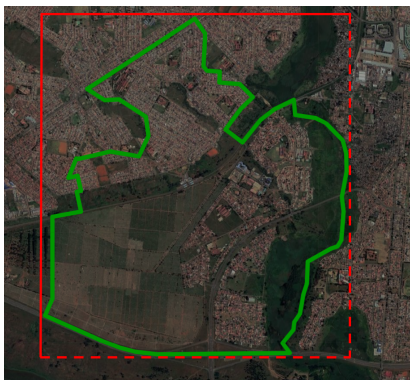

Figure A.2: A visualization of step 2 of the algorithm. The bounding box is represented by the solid and dashed red line around the ward.

mined. In order to do this, the size of the images must be converted from a pixel space to a latitude and longitude space. Thus, the approximate degrees per pixel with respect to the latitude and longitude needs to be calculated.

Let D^{lng} and D^{lat} denote the degrees per pixel with respect to the longitude and latitude, respectively. Furthermore, let z denote the value of the *zoom* parameter used in the API call and, finally, let ℓ denote the latitude associated with the center of the image of interest. Then the degrees per pixel can be approximated as follows:

$$D^{lng} = \frac{360}{2^{(z+8)}} \quad \text{and} \quad D^{lat} = \frac{360}{2^{(z+8)}} m \tag{A.1}$$

where $m = \cos(\ell \frac{\pi}{180})$. The derivation of these formulae are based on the Mercator Projection in conjunction with the specific way that the Google Static Maps API was developed. See the API documentation for more information [44].

The effect of m on D^{lat} is very small when considering the relatively small changes in latitude across the municipalities. Thus, the values for D^{lng} and D^{lat} are regarded as constant across the wards and municipalities. They are calculated with $z = 16$ and with ℓ corresponding to the latitude of the center of ward 1 in the City of

Johannesburg (arguably the most central municipality).

Finally, let h and ω denote the height and width of the satellite images (in latitude and longitude terms), respectively. Then, it follows that $\omega = 400D^{lng}$ and $h = 400D^{lat}$, where 400 is the width and height of the images in pixels. This result is visualised in Figure A.3.

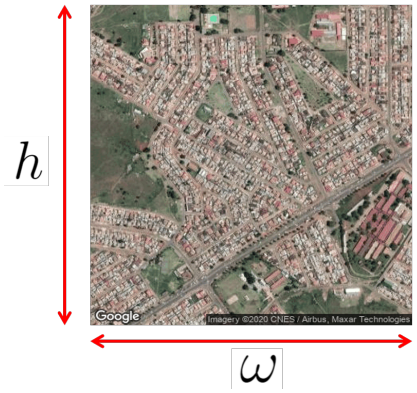

Figure A.3: A satellite image with its height, h, and width, ω, overlaid.

4. The approximate coverage of each satellite image is then used to convert the bounding box into a grid-like structure as follows:

   (a) Starting at the top-left corner of the bounding box, move $0.5h$ degrees south and 0.5ω degrees east. Define this new location, or center, as an "unadjusted" center, as stated above. The rectangle centered at this location fits in the top-left corner of the bounding box. Note, however, that this will not be the center of the final image that will be sourced, but the area of coverage of this rectangle will correspond exactly to the corresponding final images used. Hence, images of this type will be referred to as "placeholder" images. These "placeholder" images will be used to determine the location of an image to be extracted. See Figure A.4.

   With the notion of these "unadjusted" centers and their corresponding "placeholder" images in mind, an algorithm is then defined to extract all of the "unadjusted" centers for a given ward. This algorithm is displayed in Algorithm 1.

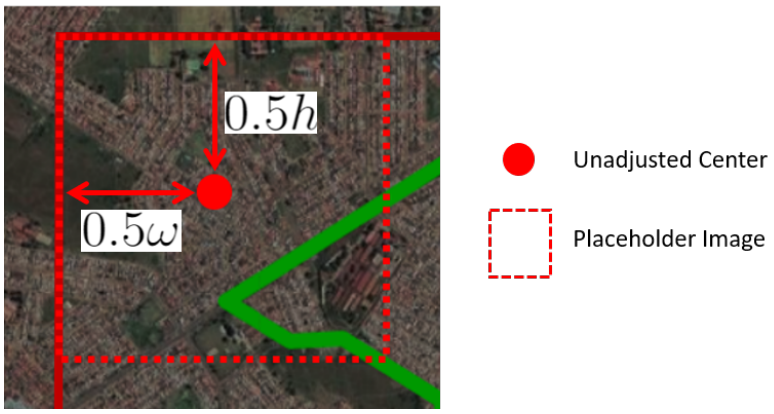

Figure A.4: A visualization of step 4(a) of the algorithm, whereby the top-left 'unadjusted' center is obtained (represented by the point). The corresponding 'placeholder' image is the rectangle enclosing the point.

Algorithm 1: Image location determination

```
/* each location/center is comprised of (lat, lng)        */
/* and is associated with a ``placeholder'' image         */
/* (max_lat and min_lng) are the top-left of the bounding
   box                                                    */
center_top_left = (max_lat - 0.5h, min_lng + 0.5w)
center = center_top_left
/* for changing to the next row                           */
while bottom of image has not crossed bottom boundary do
    /* for moving across the row                          */
    while right side of image has not crossed right boundary do
        center = (center[0], center[1] + ω)
    end
    center = (center[0] - h, center_top_left[1])
end
```

To summarize the algorithm, starting at the top-left corner of the bounding box, adjacent "placeholder" images are extracted from left-to-right until the next image is entirely outside of the bounding box. This process is repeated row-by-row until the next row of "placeholder" images are all outside of the bounding box. A visualization of the algorithm can be seen in Figure A.5. In the left image, the process has been stopped after the stopping criteria for the first row has been met, whereas, in the right image, the process has been run until termination.

Note that there will be instances where the algorithm extracts images that do

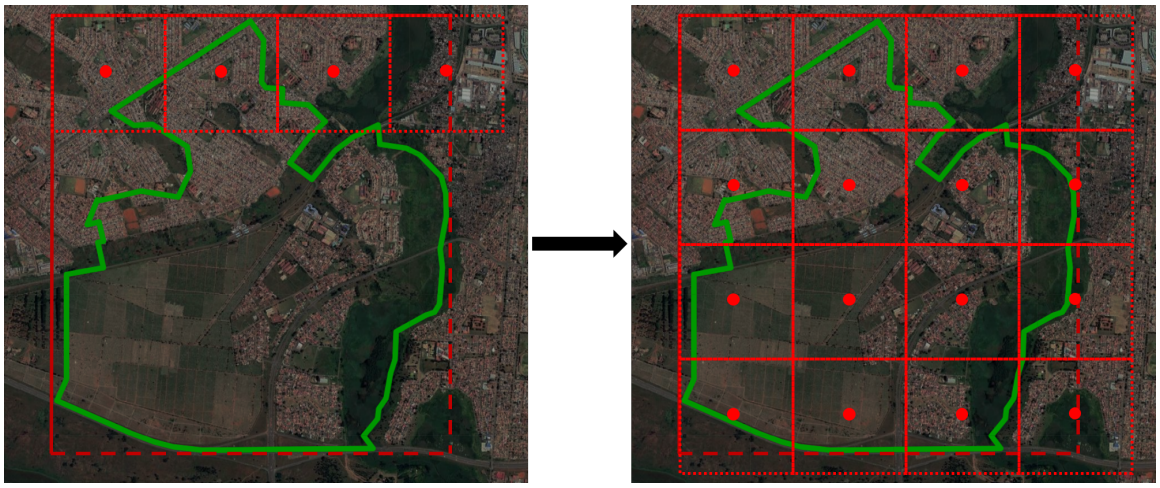

Figure A.5: A visualization of the image location determination algorithm for a given ward. Each point represents an 'unadjusted' center, while the rectangle surrounding the point is the associated 'placeholder' image.

not contain any portion of the ward of interest. For example, see the top-right "placeholder" image in Figure A.5. As these images do not contain any information pertaining to the ward of interest, they will be removed. However, they are only removed in the *image cropping and mask creation* step of the process as there is no obvious way of identifying these images at this stage. This makes the algorithm inefficient when the ward shape is not captured efficiently by the bounding box (for example, if the general ward shape is diagonal in nature) and, for some larger wards, results in over 1000 non-informative images being extracted.

Once the coordinates of the "unadjusted" centers are obtained, the image extraction process can begin.

A.3 Image Extraction

The "unadjusted" centers will now be used to determine the location of the images to be extracted. As mentioned previously, the location of these images is termed the "adjusted" centers and there is a simple mapping from the "unadjusted" centers to the "adjusted" centers.

The reason for the need to adjust the image centers and extract a slightly larger image (400x425 vs 400x400 pixels) is due to the Google watermark overlaid on each image (as seen at the bottom of Figure A.3). To remove the watermark, a simple work-around is devised. This will be explored further in the next step of the process. However, for the

purposes of image extraction, all that is required is to shift the "unadjusted" centers down by 12.5 pixels. This is achieved by simply shifting the "unadjusted" centers $\Delta = 12.5D^{lng}$ degrees south, and thus, the "adjusted" centers are obtained.

Once this is done, three 400x425 pixel images centered at each "adjusted" center are extracted from Google Static Maps API - (1) a plain satellite image, (2) a satellite image with the ward overlaid in black and (3) a satellite image with the ward overlaid in green. The reason for extracting the two images with the ward overlaid is for the purposes of mask creation. This too will be elaborated on further in the next step of the process.

Now that the three images centered at each "adjusted" location have been extracted, the next step of the process can begin.

A.4 Image Cropping and Mask Creation

The CNN used for feature extraction requires two inputs per location - (1) a 400x400 pixel plain satellite image centered at the location and (2) a 400x400 pixel image-mask associated with the plain image. The method to transform the set of three images per location obtained from the image extraction process to these two input images is described now.

A.4.1 Image Cropping

The need to remove the Google watermark is necessitated by the manner in which the CNN converts the input images to feature vectors. The CNN would most likely view the watermark as a prominent feature due to its well-defined edges and corners. Thus, keeping the watermark would have encoded undesired information in the feature vectors.

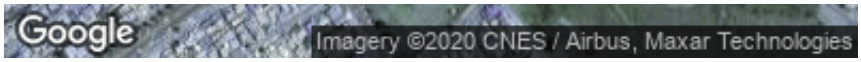

Figure A.6: The bottom 25 pixels of a 400x425 pixel Google Static Maps image with zoom level 16.

Through experimentation, it was noted that roughly the bottom 25 pixels of any 400x425 Google Static Maps image contains the watermark (see Figure A.6). Assuming that enlarging the image height from 400 to 425 pixels adds approximately 12.5 pixels to the top and bottom of the image, an image equivalent to the "placeholder" image, but with the watermark removed can be obtained by shifting the latitude of the image 12.5 pixels down and then removing the bottom 25 pixels. However, the Google Static Maps API does not include the functionality to shift an image that has already been extracted.

Thus, the centers of the images must be shifted down before the images can be extracted. Hence, the use of the "unadjusted" and "adjusted" centers. This has all been visualised in Figure A.7.

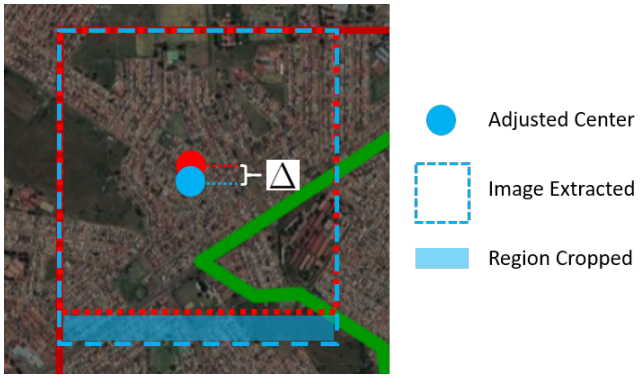

Figure A.7: A visualization of the cropping process. The blue point and the dashed blue rectangle represent the 'adjusted' center and 400x425 pixel extracted image, respectively. The area cropped is shown by the shaded blue region.

Furthermore, in Figure A.8, the effect of this cropping process on a single image can be seen. The flow of images from left-to-right denotes the "placeholder" image whose watermark needs to be removed, the 400x425 image at the "adjusted" center to be cropped and, lastly, the cropped image equivalent to the "placeholder" image but with the watermark removed. As mentioned previously, the "placeholder" image was never extracted and has only been included to demonstrate how the watermark was effectively removed, uncovering the image pixels that it was blocking.

Clearly the watermark has been cropped from the middle image, resulting in an image equivalent to the "placeholder" image centered at the associated "unadjusted" center. Linking Figure A.7 and Figure A.8, the blue rectangle in Figure A.7 represents the middle image in Figure A.8. Similarly, the red rectangle corresponds with the left or right image. All of the images are cropped, including the images with the wards overlaid.

Figure A.8: An example of a satellite image before cropping (middle) and after (right). The middle image corresponds to a 400x425 pixel image extracted from an 'adjusted' center whereas the image on the right corresponds to a 400x400 pixel image equivalent to the 'placeholder' image (left) centered at the associated 'unadjusted' center but with the watermark removed.

There is still one unanswered question that any reader might have - the need for the images

with the ward overlaid in green and black, respectively. The mask creation process will clarify their use.

A.4.2 Mask Creation

The mask creation process is necessary for the purposes of the feature extraction. Although the dynamics of how the masks will be applied will be explored in the *feature extraction* step, the method used to create the masks will be explained now.

In order to reduce repetition and improve readability, let I denote the plain cropped satellite image, and let I^b and I^g denote the cropped satellite images with the ward overlaid in black and green, respectively. An example of these sets of images $\{I, I^g, I^b\}$ can be seen in Figure A.9.

Figure A.9: A set of cropped images for the top-left 'adjusted' center from ward 11 in the City of Johannesburg. From left to right, the images represent I, I^g and I^b for this location.

The purpose of the masks is to identify which portion of a given image I is within the ward and which portion is outside. This is done by setting the pixels of the image that are outside of the ward to black and the pixels inside the ward to white. Referring to Figure A.9, it is evident that this can potentially be achieved by simply looking at the pixels that are black in I^b, setting those to white and then setting the remainder of the pixels to black. It seems reasonable to expect that the resulting image will be the desired mask. However, some of the pixels in I^b outside of the ward are black in their own right. This means that the above process results in a mask with the odd black region being tarnished by a few unwanted white pixels.

In order to work around this, a rather inelegant solution is devised which involves the need for the images I^b and I^g as follows:

(a) Method for isolating pixels in the ward

Let B be the pixels that are black in I^b and let G be the pixels that are green in I^g. The pixels inside the ward are then IN = B $\cap$ G. In other words those pixels that occur as

both green in image I^g and black in image I^b.

(b) Method for isolating pixels out the ward

Let ∼B be the pixels that are not black in I^b and let ∼G be the pixels that are not green in I^g. The pixels outside the ward are then OUT = ∼B ∪ ∼G. In other words, those pixels that are not black in image I^b, not green in image I^g or both.

The sets of pixels obtained can then be used to create the corresponding mask from I^b by setting each of the pixels in IN to white and all of the pixels in OUT to black. An example of this, using the same images as in Figure A.9, can be seen in Figure A.10.

Figure A.10: A demonstration of the use of I^g and I^b to create their corresponding mask.

The cropping and mask creation process is applied to each of the images extracted in the image extraction step. The resulting output is a set of two images for each location: (1) a 400x400 pixel plain satellite image and (2) the 400x400 pixel mask associated with the plain image. For an example of these pairs, see Figure A.11.

Figure A.11: An example of the pair of images obtained at each sampled location. The left image is a plain satellite image whereas the right image is its corresponding mask.

As mentioned previously, there will be images that do not contain any of the ward of interest. Formally, this occurs when IN= ∅. In these instances, the mask and the plain image are not saved. See Figure A.12 for an example this. Discarding these non-informative images will save computational time during the *feature extraction* process.

The extraction and cropping processes need to ensure that the resulting images cover the ward with no overlap and no uncovered regions. In order to determine if this is the

case, a ward can be reconstructed by piecing together the individual images extracted for the ward. For illustration purposes, ward 11 in the City of Johannesburg has been reconstructed using its corresponding masks. The resulting plot can be seen in Figure A.12.

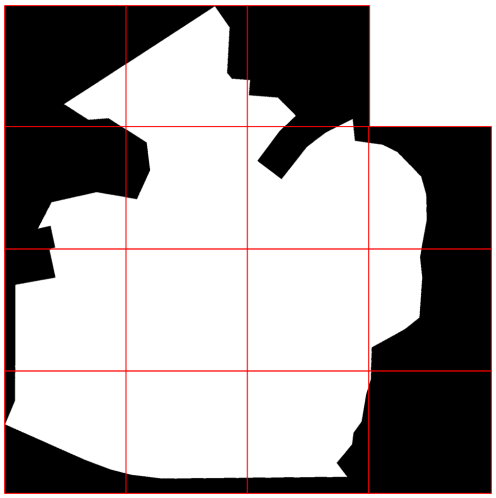

Figure A.12: Ward 11 in the City of Johannesburg reconstructed from the created masks. A red boundary has been imposed on each of the masks for illustration. The top right cell of the grid is missing as the area covered by the image did not contain any of the ward.

Referring to Figure A.12, it appears that the reconstructed ward boundary is equivalent to the original ward boundary seen in Figure A.1. This suggests that the algorithm implemented covered the ward in its entirety with no obvious overlapping regions.

Appendix B

Link to GitHub Repository

The code used for this paper can be found in the following GitHub repository:

`https://github.com/ANDRYA005/Clustering-of-Ward-Level-Deprivation-through-the-use-of-Satellite-Imagery`