Exploring unsupervised clustering models as a basis for recommendation engines

Yutong He

1 Objective

This project focuses on applying unsupervised techniques for principal component analysis and to develop clustering models. We will explore data features, develop several machine learning (ML) models, and arrive at a preliminary choice of one model over the others that fulfills the objective the best. The analysis is makes use of a Kaggle dataset on Stockholm Airbnb Listings. The model development processes are available at this Jupyter Notebook.
Exploring unsupervised ML models could potentially provide data-informed insights to the property rental market by providing relevant recommendations. Some anticipated tasks are as follows:

1. Data cleaning and exploratory analysis. This entails dealing with missing entries or entries with invalid values, encoding non-numeric values to numeric ones, scaling the numeric values such that all features have the same input range for the model. Since this project concerns unsupervised learning, no column will be selected as the target, and all columns will be treated as input features.
2. Non-numeric values will need to be filtered out or encoded such that the dataset going into the ML models will contain only numbers.
3. We will visually inspect the correlations between different input features to gain a first-step intuitive understanding of the dataset.
4. A few ML models, such as K-means, mean-shift, and Density-Based Spatial Clustering of Applications with Noise (DBSCAN), will be used to compare their effects/choices on clustering.

2 Data description and preparation

2.1 Data features

The dataset contains 7854 listings (rows) and 16 features (columns) initially. The features include a mix of object variables (listing name, host name, room type, neighbourhood, etc.), numeric variables (latitude, longitude, price, number of reviews, etc.), and a time stamp (date of last review). The header and the first 5 rows are partially shown in figure 1.

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	reviews_per_month
0	42808	Quiet room right DOWNTOWN Stockholm	186922	Nina	NaN	Norrmalms	59.34342	18.05298	Private room	853	1	64	2019-08-28	0.58
1	53895	Modern Architecture in Stockholm	252075	Eva	NaN	Skarpnäcks	59.27054	18.11231	Private room	1079	3	7	2017-05-28	0.07
2	145320	In the middle of it all - with a view!	703851	Kim	NaN	Södermalms	59.31364	18.05256	Private room	1285	2	72	2019-06-25	2.62
3	155220	Stockholm, new spacoius villa	746396	Madeleine	NaN	Skarpnäcks	59.24667	18.17799	Entire home/apt	1197	3	0	NaN	NaN
4	155685	Hornstull with water view!	748592	Robert	NaN	Södermalms	59.31535	18.03277	Entire home/apt	3247	4	22	2015-12-01	0.23

Figure 1: Partial header and first 5 rows of the dataset. The full dataset has 16 columns.

2.2 Data cleaning

At the preprocessing stage, we performed the following tasks.

1. We dropped the rows with missing or invalid values in four columns indicated in figure 2.

```
(id                                0
 name                             25
 host_id                           0
 host_name                         8
 neighbourhood_group            7854
 neighbourhood                     0
 latitude                          0
 longitude                         0
 room_type                         0
 price                             0
 minimum_nights                    0
 number_of_reviews                 0
 last_review                    1581
 reviews_per_month              1581
 calculated_host_listings_count    0
 availability_365                  0
 dtype: int64,
 (7854, 16))
```

Figure 2: Dataset information. It contains missing or invalid values in four columns.

2. Listing names (name column) were dropped. We converted the time stamp (last review) column into number of days between the last review and the latest recorded date in the column.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6262 entries, 0 to 7832
Data columns (total 12 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   name                            6262 non-null   object
 1   neighbourhood                   6262 non-null   object
 2   latitude                        6262 non-null   float64
 3   longitude                       6262 non-null   float64
 4   room_type                       6262 non-null   object
 5   price                           6262 non-null   int64
 6   minimum_nights                  6262 non-null   int64
 7   number_of_reviews               6262 non-null   int64
 8   last_review                     6262 non-null   object
 9   reviews_per_month               6262 non-null   float64
 10  calculated_host_listings_count  6262 non-null   int64
 11  availability_365                6262 non-null   int64
dtypes: float64(3), int64(5), object(4)
memory usage: 636.0+ KB
```

Figure 3: Dataset information. It is free of invalid values but still contains object types, which need to be encoded to numbers.

3. We encoded neighbourhood object to integers, introducing additional column, such that the final dataset contains only numeric information summarized in figure 4.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6262 entries, 0 to 7832
Data columns (total 27 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   latitude                                6262 non-null   float64
 1   longitude                               6262 non-null   float64
 2   price                                   6262 non-null   float64
 3   minimum_nights                          6262 non-null   float64
 4   number_of_reviews                       6262 non-null   float64
 5   last_review                             6262 non-null   float64
 6   reviews_per_month                       6262 non-null   float64
 7   calculated_host_listings_count          6262 non-null   float64
 8   availability_365                        6262 non-null   float64
 9   neighbourhood_Bromma                    6262 non-null   float64
 10  neighbourhood_Enskede-Årsta-Vantörs     6262 non-null   float64
 11  neighbourhood_Farsta                    6262 non-null   float64
 12  neighbourhood_Hägersten-Liljeholmens    6262 non-null   float64
 13  neighbourhood_Hässelby-Vällingby        6262 non-null   float64
 14  neighbourhood_Kungsholmens              6262 non-null   float64
 15  neighbourhood_Norrmalms                 6262 non-null   float64
 16  neighbourhood_Rinkeby-Tensta            6262 non-null   float64
 17  neighbourhood_Skarpnäcks                6262 non-null   float64
 18  neighbourhood_Skärholmens               6262 non-null   float64
 19  neighbourhood_Spånga-Tensta             6262 non-null   float64
 20  neighbourhood_Södermalms                6262 non-null   float64
 21  neighbourhood_Älvsjö                    6262 non-null   float64
 22  neighbourhood_Östermalms                6262 non-null   float64
 23  room_type_Entire home/apt               6262 non-null   float64
 24  room_type_Hotel room                    6262 non-null   float64
 25  room_type_Private room                  6262 non-null   float64
 26  room_type_Shared room                   6262 non-null   float64
dtypes: float64(27)
memory usage: 1.3 MB
```

Figure 4: Dataset information. Now it contains only numeric values and is ready to be fed into ML models.

2.3 Feature correlation

Upon readying the dataset, we visualize the correlation among the input features via a heatmap in figure 5 and see that the correlation strength varies among different columns. In section 3.1, we will perform principal component analysis to transform these raw features to linear combinations of them.

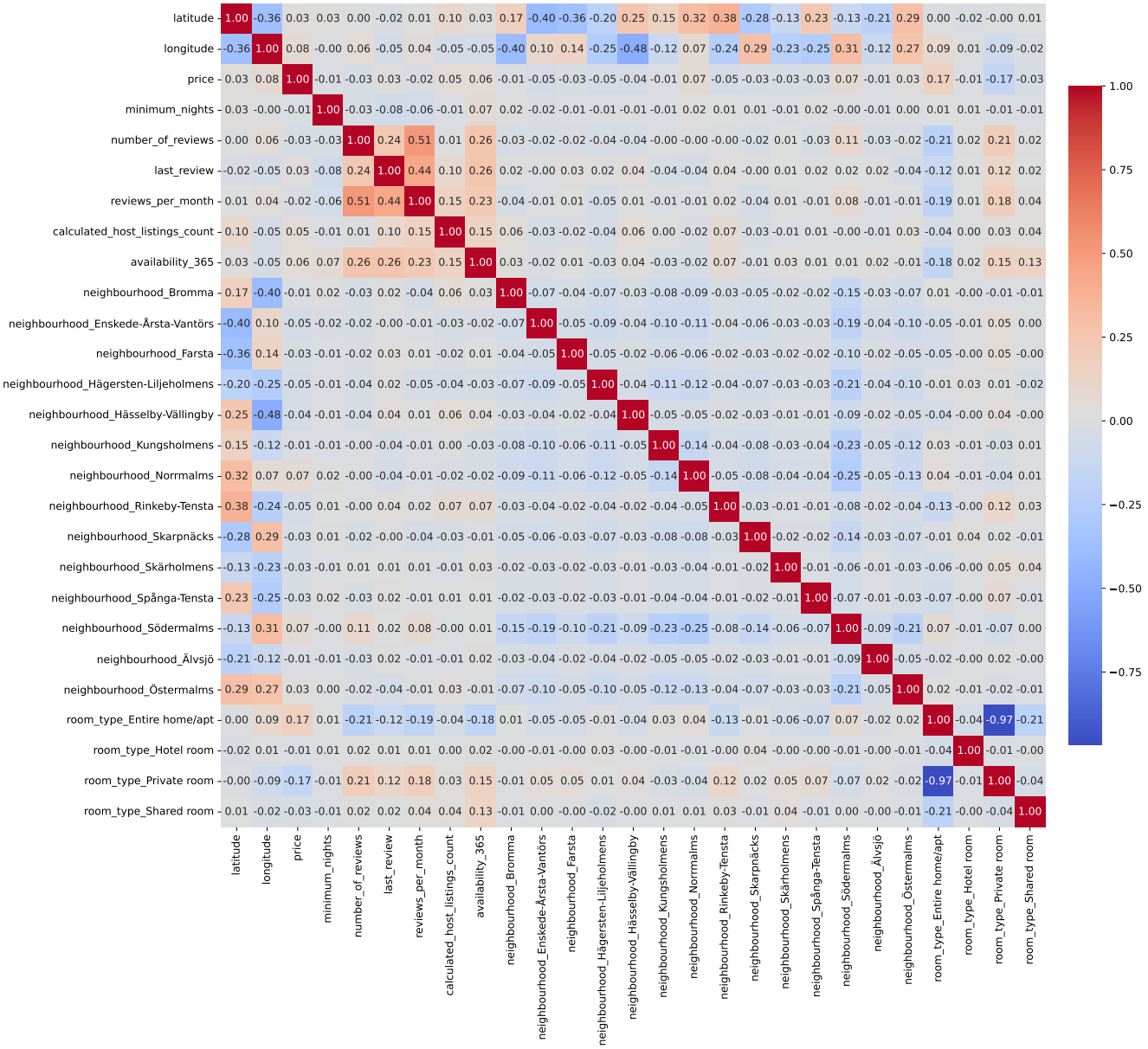

Figure 5: Heatmap showing the dataset features and their correlations, where the color gradient indicates correlation strength.

3 Clustering models

Four clustering models were developed for this project: K-means combined with PCA, mean-shift combined with PCA, DBSCAN in the PCA feature space, and DBSCAN within T-Distributed Stochastic Neighbor Embedding (TSNE). Different sets of model parameters corresponding to the relevant models were searched through for tuning purposes. See the Jupyter Notebook for detailed model development.

3.1 Principal component analysis (PCA)

3.1.1 Visualization of reduced PCA

PCA decomposes the original dataset into a series of components ranked from the most important to the least, i.e., component 1 always outweighs component 2 and so on in determining the data direction in the feature space. For a dataset with many features such as in this project, we provide an intuitive picture of PCA by restricting to its first 1, 2, and 3 components. The visualized projection onto these components are shown in the three panels in figure 6.

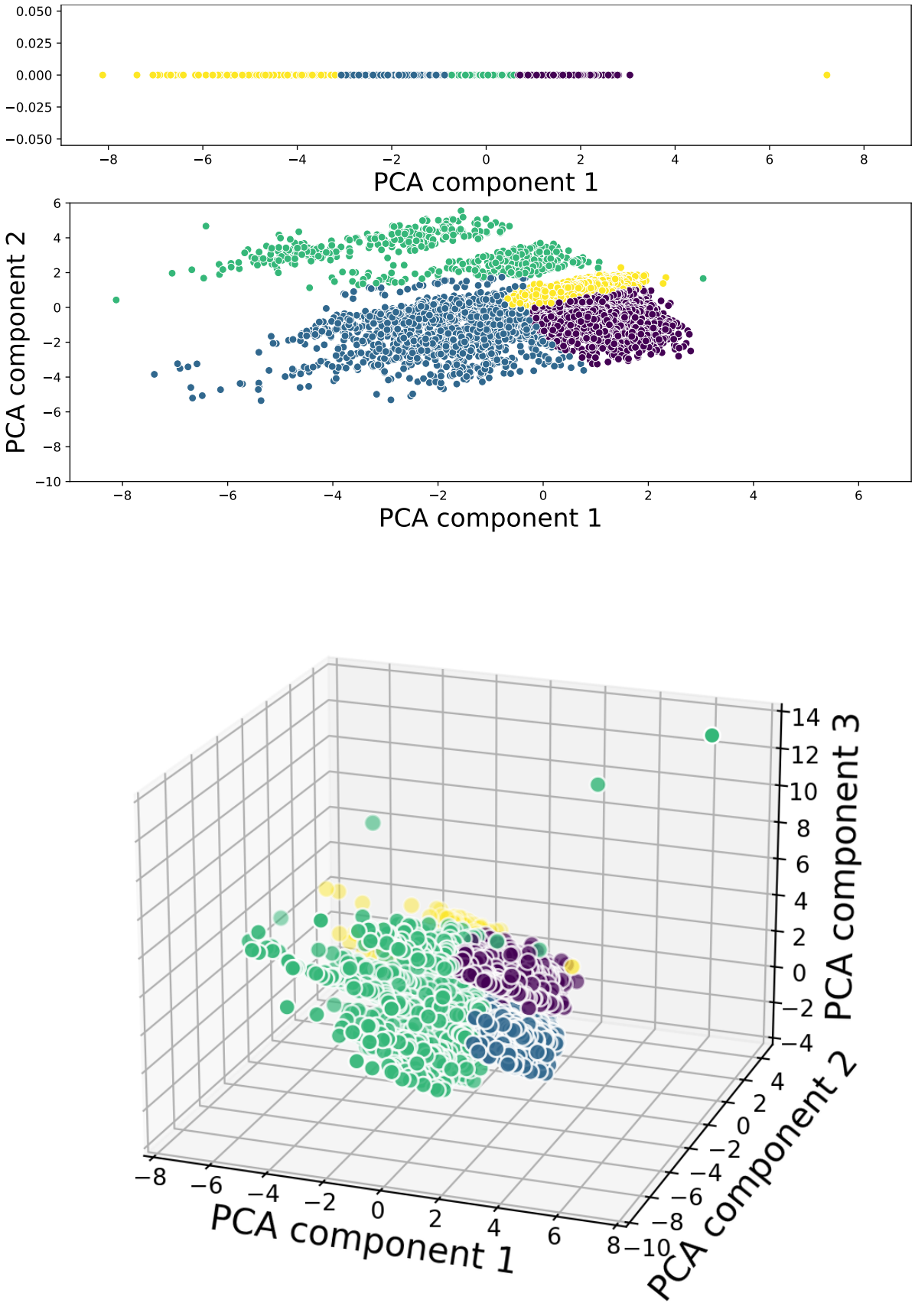

Figure 6: Visualization of projections onto 1 (upper panel), 2 (middle panel), and 3 (lower panel) PCA components.

3.1.2 Before and after PCA

To more rigorously determine the number of PCA components to keep, we present figure 7, which shows the accumulated explained variance as a function of the number of PCA components. For example, in order to recover 95% of the original dataset, we would need to keep 21 components and safely discard the remaining 6. This could reduce the computational complexity compared to operating on the original dataset containing all 27 PCA components. The feature correlation from before and after the removal of 6 tail components are compared in figures 8 and 9. We keep 21 components through the rest of this project.

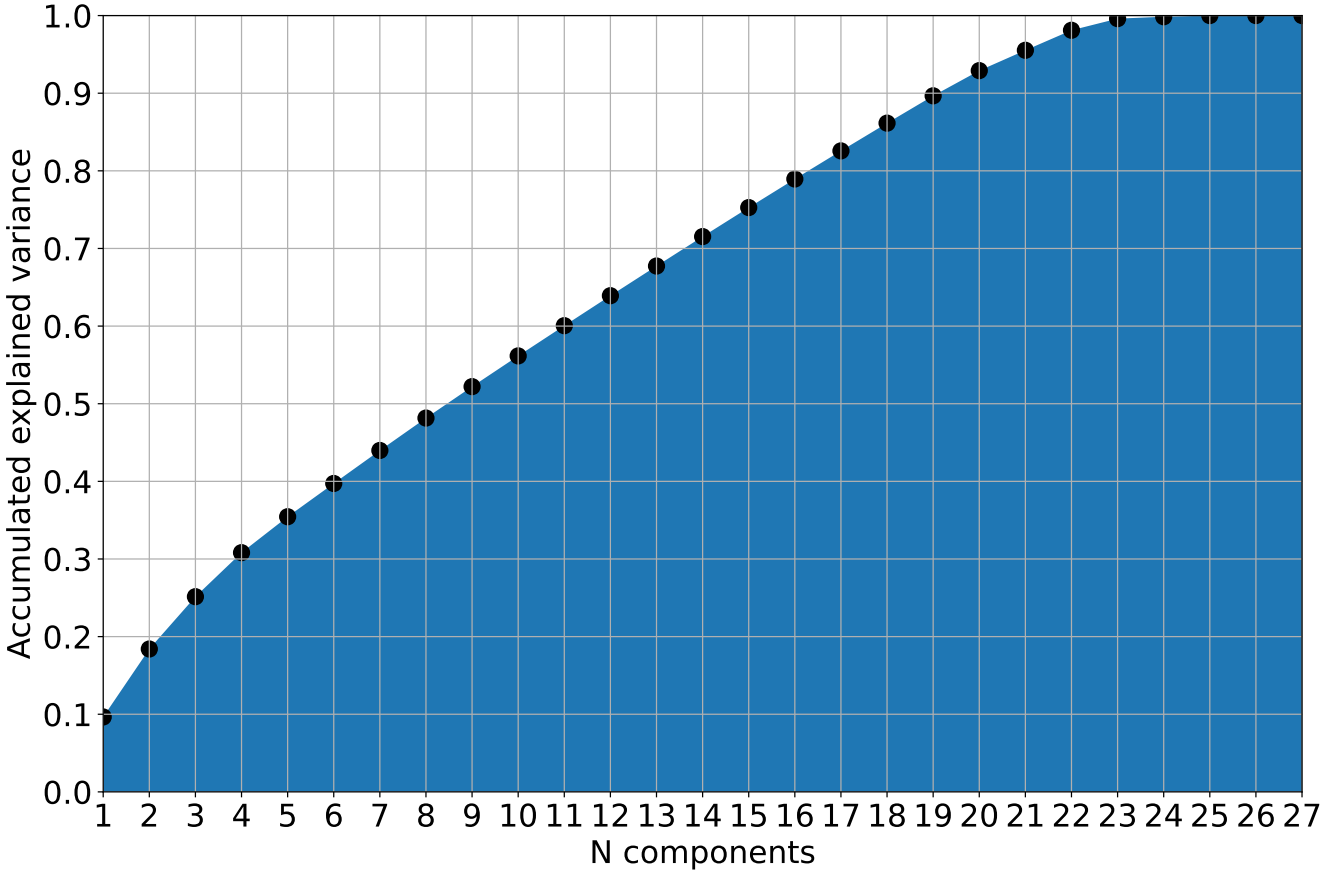

Figure 7: Accumulated recoverable variance increases as the number of PCA components increases.

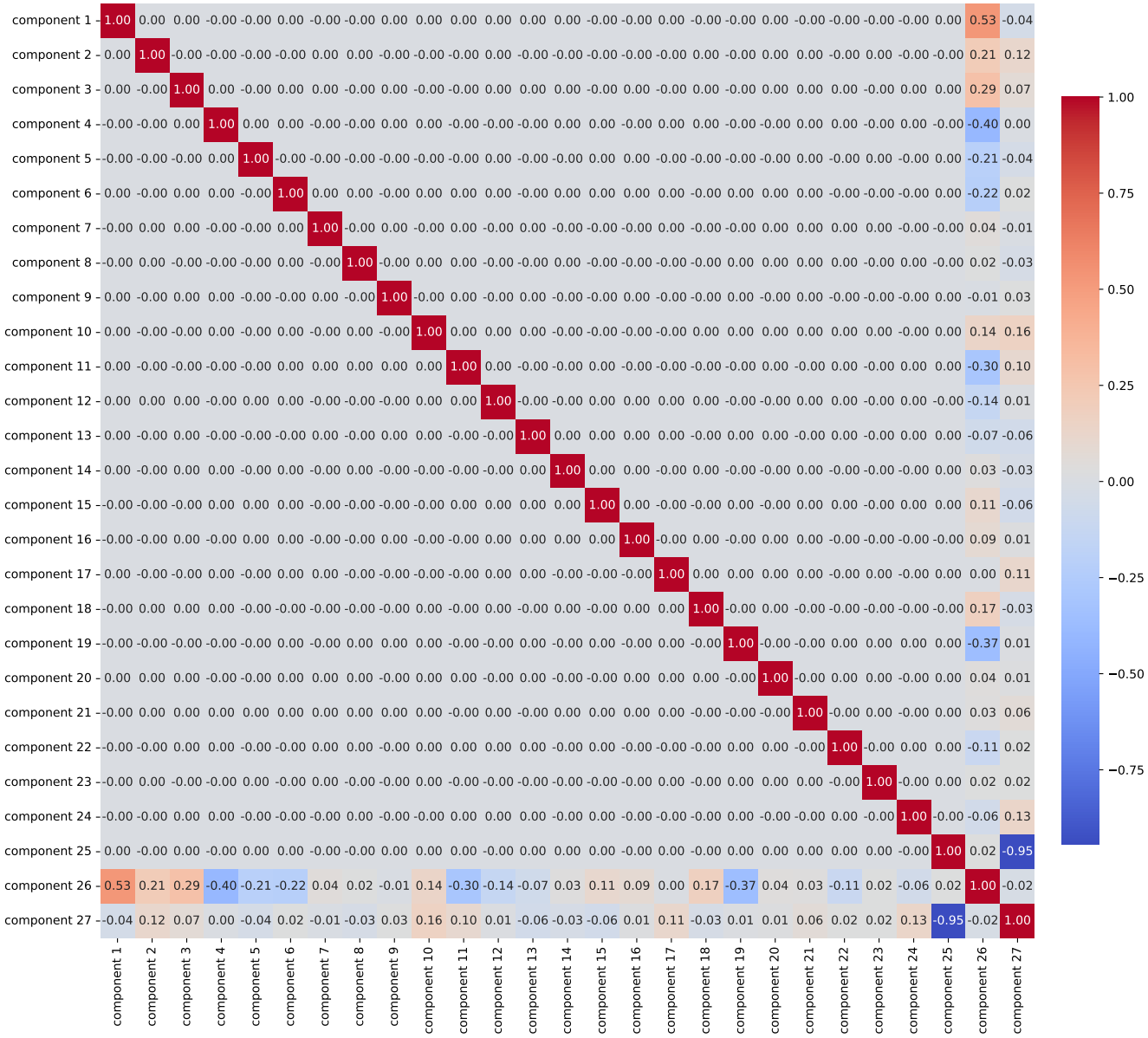

Figure 8: Feature correlation between all 27 PCA components.

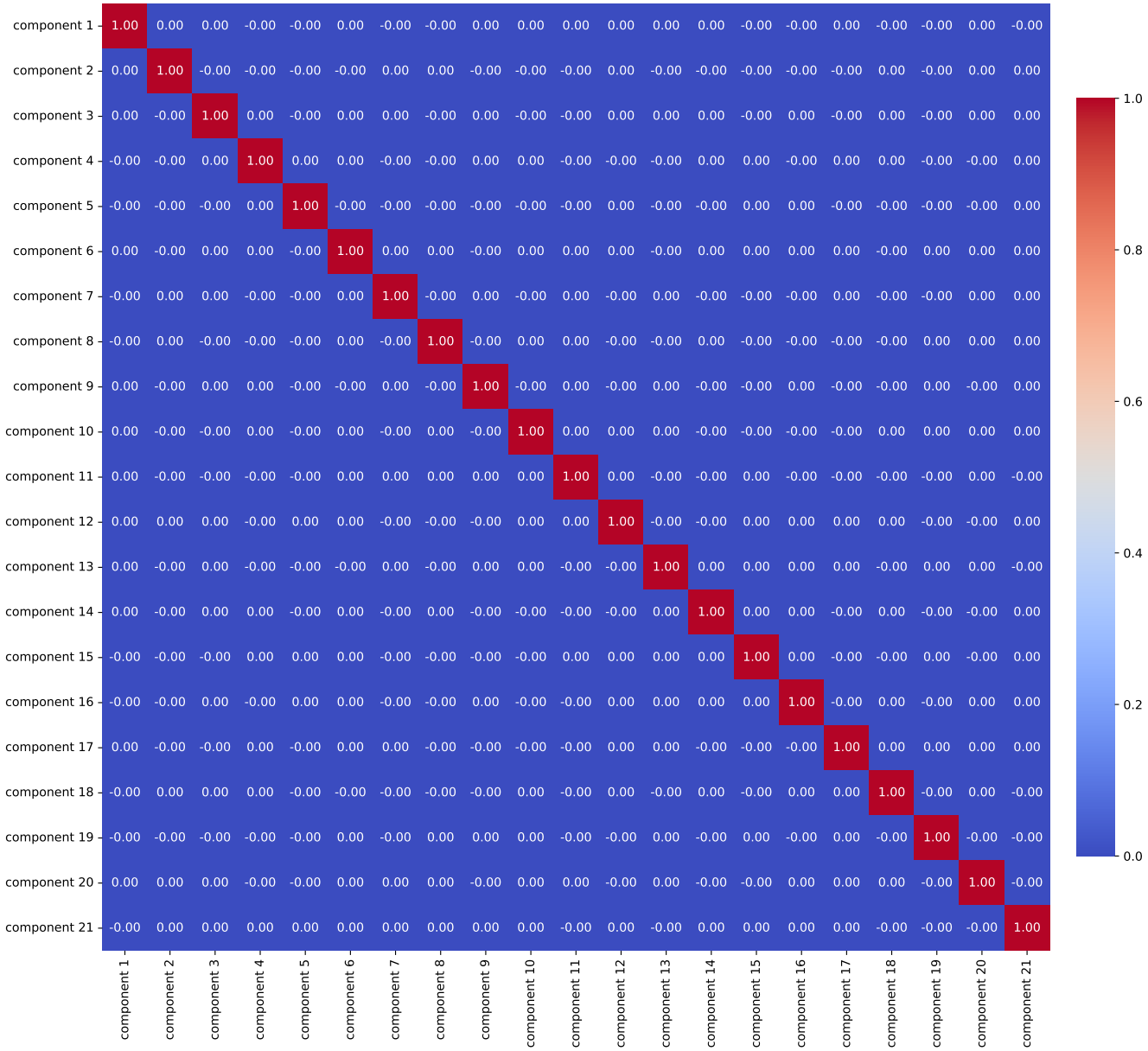

Figure 9: Feature correlation between 21 PCA components after removing the 6 at the tail.

3.2 Clustering model 1: K-means

K-means clustering groups together similar data points. A key input variable is the number of clusters, which can be empirically found by looking at the elbow curve in figure 10. The inflection point around $n = 20$ is optimal.

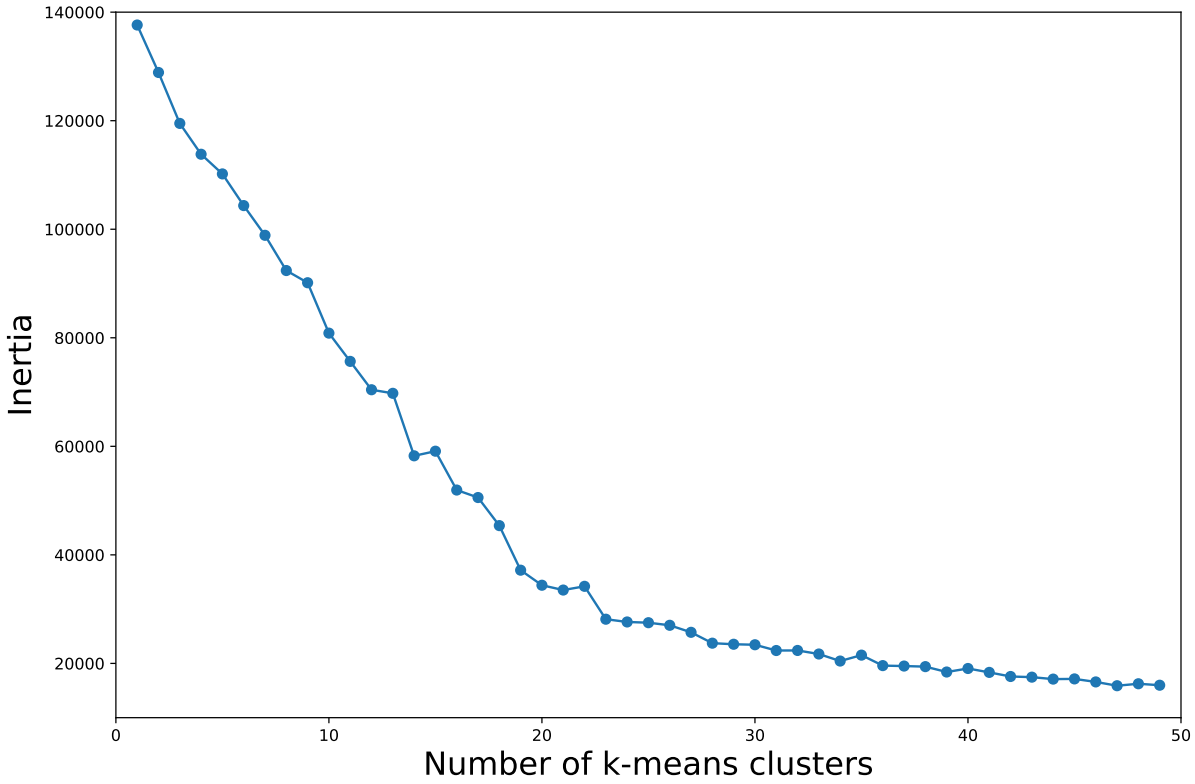

Figure 10: The effect of k-mean cluster numbers on inertia, the sum of squared distance to the centroids.

Therefore we set the input parameters shown in figure 11 for a K-means clustering on the PCA feature space with 21 components.

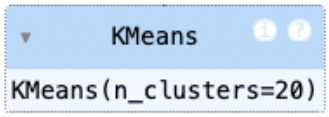

Figure 11: Model parameters for the logistic regression classifier with l1 penalty.

The resulting K-means clusters for the first two components are visualized in figure 12, where 20 clusters are identified. Note that the overlapping between different clusters are likely the result of projected visualization in reduced PCA space. Because in the full PCA component space, all identified clusters are designed to be separated from each other. We will visually examine the outcome from other models in the following sections.

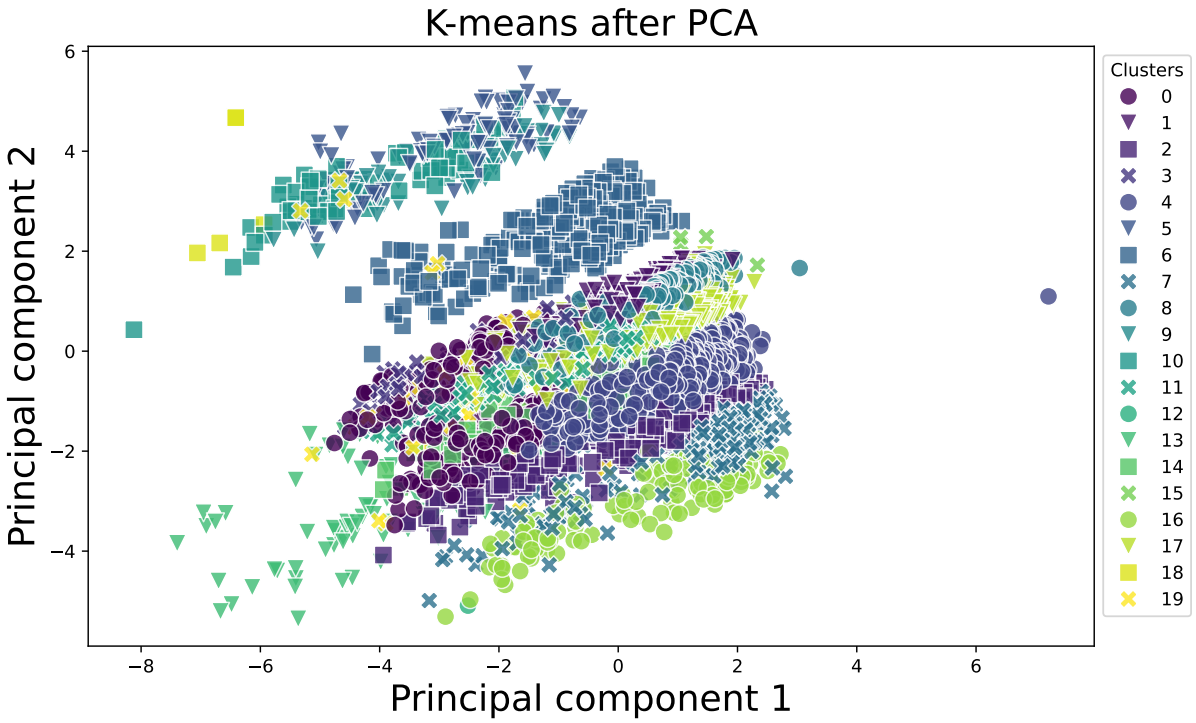

Figure 12: 20 clusters identified via K-means and projected onto two PCA components.

3.3 Clustering model 2: mean-shift

We used the input parameters shown in figure 13 for a mean-shift clustering model and obtained the performance results visualized along two PCA components in figure 14.

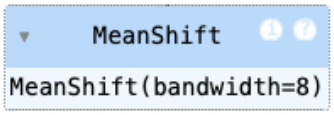

Figure 13: Model parameters for the logistic regression with elastic-net penalty.

As the name suggests, mean-shift model gradually shifts from a random starting point to the mean of the surroundings and eventually settles down to a location where the mean is close to / align with the centroid. This model is similar to K-means as it also relies on an input parameter that indicates the number of clusters, as the bandwidth parameter is, roughly speaking, inverse to the number of clusters. The resulting clusters are also similar to K-means, with 16 clusters identified. From the perspective of only the first two PCA components, mean-shift appears to perform better as there is less overlapping between different clusters. But this could well be invalid in the full PCA component space.

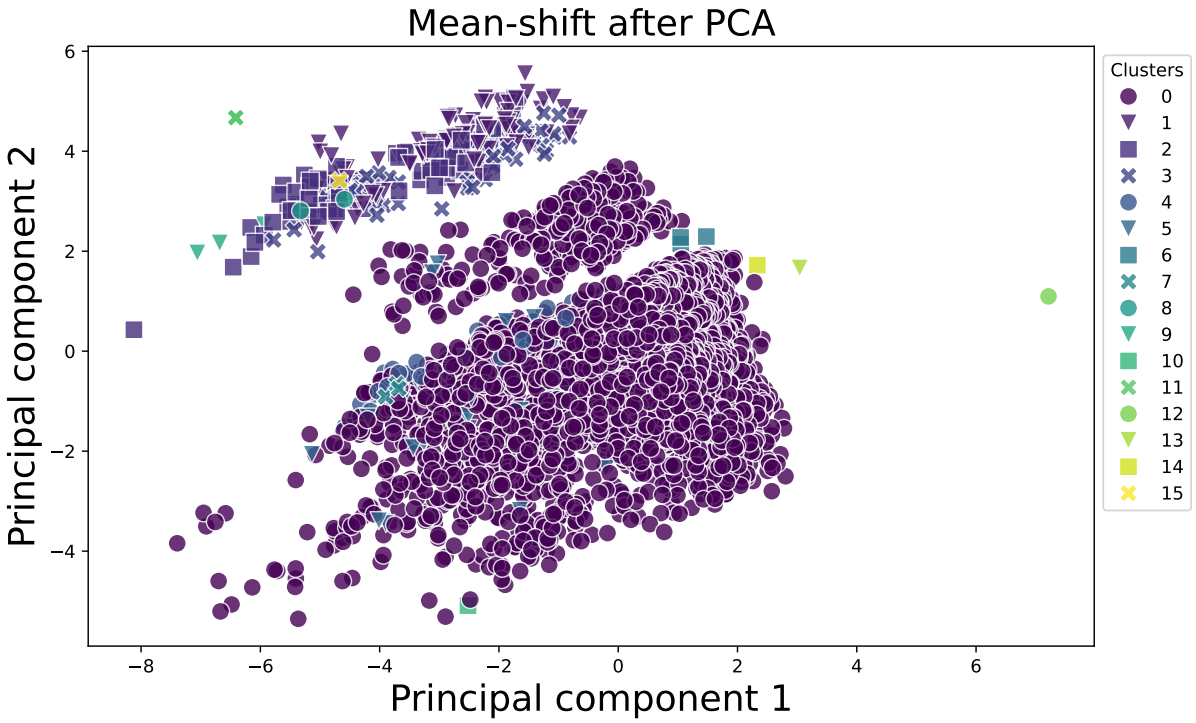

Figure 14: 16 clusters identified via mean-shift and projected onto two PCA components.

3.4 Clustering model 3: Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

We used the input parameters shown in figure 15 for a DBSCAN model and identified the groups visualized along two PCA components in figure 16.

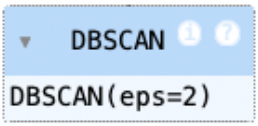

Figure 15: Model parameters for the decision tree classifier.

DBSCAN starts by searching for a predetermined number of points within a predetermined perimeter range from randomly selected centroids. The algorithm then sweeps through all points within the perimeter and successively identifies different clusters depending on their surroundings. This method does not require the number of clusters as an input but rather converges to it automatically based on the density distribution of points. This key feature differs DBSCAN from K-means and mean-shift techniques. The autonomy also allows for identification of noise points that do not contain enough points above the threshold. Overall figure 16 shows that DBSCAN found 37 clusters along with 209 points of noise, which account for 3.34% of the total data points.

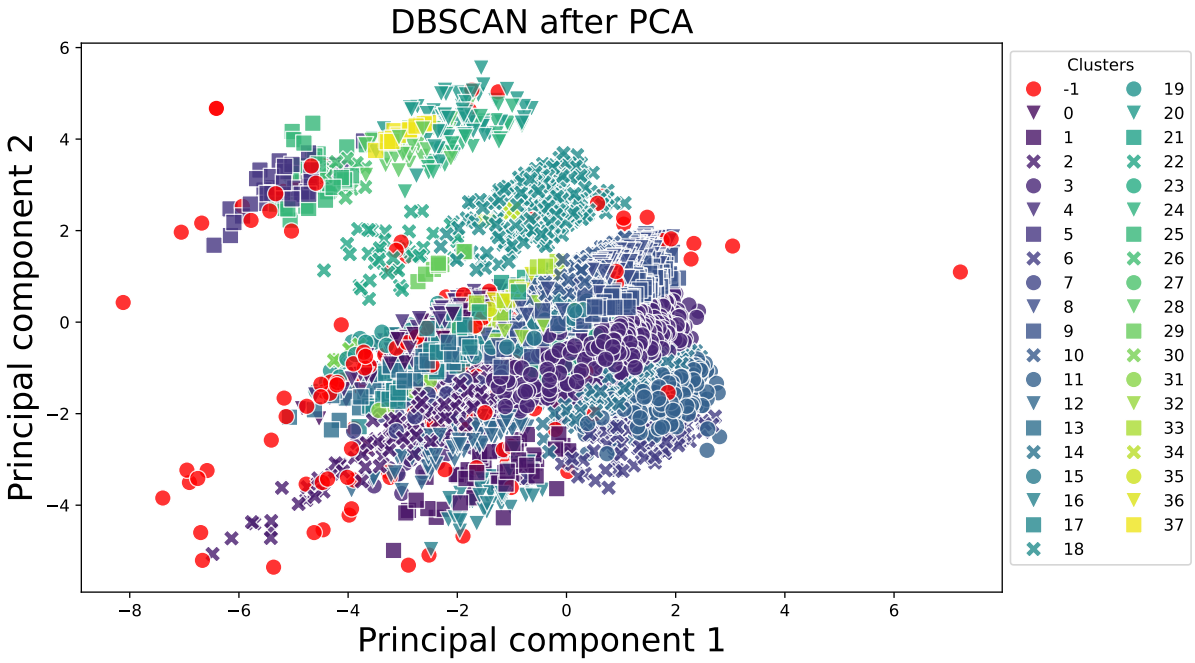

Figure 16: 37 clusters and 209 points of noise (red dots), accounting for 3.34% of all points, identified via DBSCAN and projected onto two PCA components.

3.5 Clustering model 4: DBSCAN within T-Distributed Stochastic Neighbor Embedding (TSNE)

For high-dimensional datasets, features could be nontrivial to disentangle and cluster within the original feature space. PCA helps to some extent as it searches for alternative parameter space in terms of linear combinations of the raw features. But as can be seen from figures 12, 14, and 16, many identified clusters overlap with each other. Although this might not reflect the full reality using all PCA components, it is well motivated to apply the same idea of identifying separations in another paramter space. For the fourth model of this project, we embed the original data features via the TSNE method and subsequently perform DBSCAN to group similar points together. The DBSCAN model parameter is inherited from before in figure 15 and the performance results are shown in figure 17.
The left panel of figure 17 shows the data points transformed and distributed within the TSNE space, and the right panel of figure 17 illustrates 61 clusters and 47 points of noise, which account for 0.76% of the total, found by DBSCAN. 4 times fewer noise points are flagged in TSNE space than among PCA components using the same DBSCAN clustering method. This could be due to the more efficient grouping from the TSNE embedding, as is visibly seen by comparing figures 16 and 17.

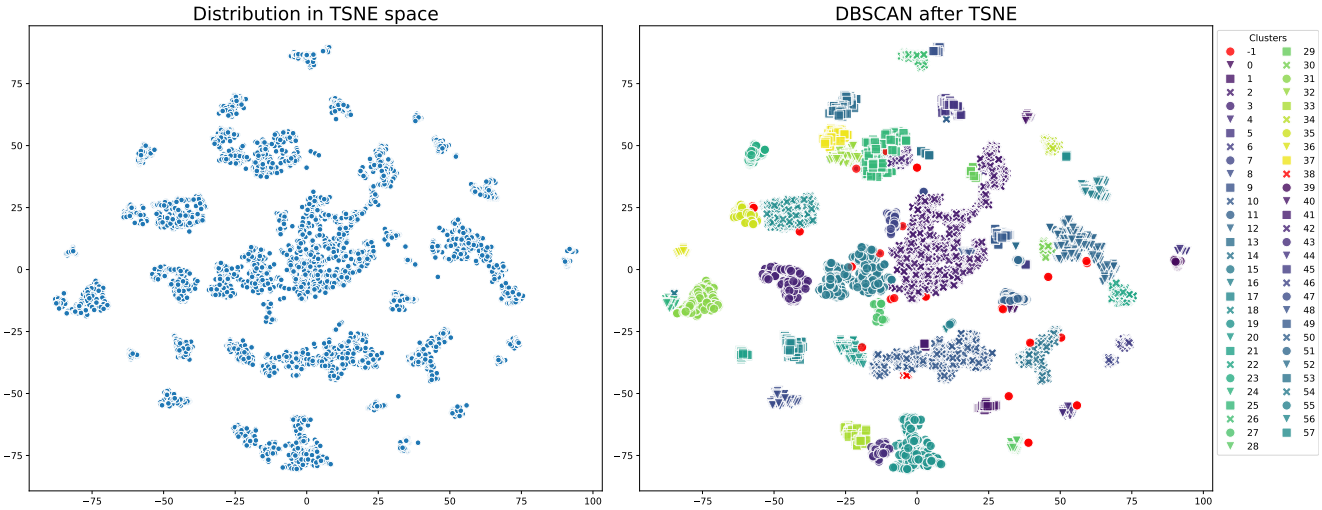

Figure 17: 61 clusters and 47 points of noise (red dots), account for 0.75% of all points, identified via DBSCAN in the TSNE space.

3.6 Best clustering model for the objective: DBSCAN within TSNE

We compare the identified clusters from the 4 ML models presented in sections 3.2–3.5 and choose model 4, i.e., DBSCAN within TSNE space, as the most suitable for cluster identifications of the dataset presented in section 2. We make the following notes:

1. The verdict is made qualitatively by visually examining the clustering patterns from 4 different models. The quantitative only came in the form of the percent of noise points between model 3 and 4. However, to make a more rigorous assessment, we would ideally need to perform similarity/dissimilarity tests and rank the performance of the models numerically. We leave this as the next step.

2. The model outputs also depend on the way the initial dataset is handled during the preprocessing and exploration stage. For example, we decided to encode the neighbourhood column and thus expanded into a larger feature space. If the column was dropped from the start instead, the number of input features would also be lower and thus ordinary PCA might have done a better job at clustering.

4 Insights and key findings

Over the course of the project, we summarize the following key insights and findings:

1. Compared to supervised tasks such as regression and classification models, unsupervised objectives present their own challenges. But many tools are also made readily available such as the sklearn package in Python, which this project heavily relied upon.

2. Data preprocessing and exploration also takes up most of the project time, as in the case for supervised models. If chosen well, the dataset could be adapted more optimally for clustering tasks than the brute-force approach of dropping all texts and encoding all labels/categorical values.

3. ML models are project specific and need to be customized to address the individual questions and datasets. This is why we constructed 4 different clustering models and tentatively determined DBSCAN combined with TSNE was the best to separate data points into groups.

4. Clustering models explored in this project seem to have a lower level of human interpretability than many supervised learners such as decision trees and regression. This is because the features are transformed into parameter spaces that are easier to operate but that may not provide straightforward intuition to the tangible parameters from the original dataset. But precisely because they

are free to search for data features more abstractly, potentially significant groups could be identified (such as the case in TSNE space) that would be harder when working with the raw features directly.

5 Next steps

A number of aspects could be improved in the future, in term of both the technical model development and the data aspects.

1. As alluded to in section 3.6, to accurately deciding the best model requires quantifying model performance via different similarity scores. So a next-step task could be to reserve a fraction of the data as the cross-validation set to provide a comparison with each model.
2. A related aspect is to grid search through the input model parameters in order to obtain a set of more fine-tuned results for the particular clustering task. In this project we only tuned the parameter values by hand.
3. In terms of the dataset, more information could be extract, such as the text column(s) regarding the listing names. This could serve as a complementary task in parallel to the work on extracting patterns from the columns with numeric values.