Student IDs: [redacted]

Date: 28/02/2025

Assignment 1 – Machine Learning and Deep Learning (CDSCO2041C)

Introduction

During the 2024 Summer Olympics in France, the demand for Airbnb rental properties saw a significant increase, as both participants and spectators sought convenient accommodations in Paris. Although the overall increase in price was relative to most rental property owners, some Airbnb booking prices experienced a disproportionate increase, compared to others. This raises the problem statement:

"What are the primary factors that contribute to the increase in Airbnb booking prices?"

This report aims to analyze the primary factors that contribute to the rental prices, in the context of the 2024 Summer Olympics held in France. The analysis discovers these factors through an Exploratory Data Analysis (EDA), followed by a K-means Cluster Analysis, and Principles Component Analysis (PCA), while actively extracting meaningful insights into the data provided.

Furthermore, the framework of this analysis relies heavily on the 2nd chapter in Aurélien Géron's book, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*.

Exploratory Data Analysis

Exploratory data analysis (EDA) is essential to discover the primary factors, increasing booking prices for Airbnb listings in Paris. In this section, we will explain how we utilized EDA to clean, transform and to find patterns in the data. Additionally, we will shortly explain how EDA helped us gain insights through visualizations and statistical analysis.

Firstly, to restrict the range of the duration to the 2024 Summer Olympics, the March calendar CSV was introduced and limited to the timeframe 16 March – 11 August. Additionally, the data type of the price column from the calendar CSV was converted to float, and the "$" was removed to ensure proper formatting and accurate analysis. At last, the date was converted to datetime data, to appropriately support the analysis of historical data.

To find relevant features to include in the cluster analysis, `cov_corr_function` was built to perform covariance and correlation-based analysis and to detect price variability. This function was utilized several times in conjunction with `visualize_correlation` for verifying which features had the largest impact on price inflation. The code for these functions can be found in the Jupyter Notebook attached with the report and in Appendix A.

Next, we deemed it necessary to transform the data, since there was almost no correlation with price on first inspection. See Figure B.1 in the Appendix. First, we log transformed the '*price*' feature. Furthermore, we transformed other numeric variables with either the log, square- or cube root, as it reduced the variability of the data and thus made it easier to find features that correlates with the price. Numeric features such as longitude, latitude and dummy features were not transformed. Plotting the price distribution across Paris we clearly see that location correlates with price, as the listings get more expensive the closer, they get to the city center (see Figure B.2 Appendix B). Therefore, we created dummy features for three neighborhoods: '*city_center'*, '*inner_city*' and '*outer_city*' and dropped '*neighborhood'* as a feature afterwards. The same was done with the '*room_type*', which was split into four dummy features.

Lastly, we scaled the data with the standardization method. We deemed it was necessary to both transform and scale data as it showed non-linear patterns when plotting histograms for each selected feature. The results, we use for our cluster analysis is in Appendix: Figure B.3.

Cluster Analysis

Clustering segment groups of similar instances, making it possible to understand underlying patterns in the data. In this section, we outline our approach for the cluster analysis, through feature selection, model training, evaluation, and fine tuning. We utilized two clustering models – K Means and DBSCAN – and compared their outputs to discover what impacts booking prices on Airbnb in Paris.

First, relevant features were chosen based on a correlation threshold of |0.1| or higher in comparison to the price feature. This ensured we kept the most impactful features. Next, we leveraged the sklearn library to estimate the KMeans model for our chosen features. We fine-tuned the model by trying a selection of different parameter values to get the best silhouette score.

We created the function `plot_elbow_silhouette` to fulfill this task. As the name implies, it plots the inertia and silhouette score for KMeans models with clusters in the range of 2 to 10. The plot in Appendix C shows the best silhouette score appears at three clusters, leading to a silhouette score of approximately 0.32 (see Appendix: Figure C.1). Next, we estimated a DBSCAN model with the features selected features. The model was fine-tuned through another function `get_scores_and_labels`. The best result from this function gave us a value selection of $\varepsilon = 0.05$ and a minimum sample of 500 as the best parameters for DBSCAN. This provided a silhouette score of approximately 0.25 with 3 clusters.

Figure C.2 and C.5 in the Appendix are pairwise plots of the selected features (x-axis) compared to the price (y-axis), where the observations are colored in their respective clusters. From the pairwise plot for the KMeans clusters there looks to be some sort of division between the clusters, when looking at features '*calculated_host_listings_count*' and '*minimum_nights*' compared to the price. Figure C.3 and C.4 shows this division between clusters better in two 3D plots. Here it becomes clearer that there is a pattern in the clusters, verifying the features impact on price. On the contrary, the pairwise plot in Figure C.5, does not show any apparent pattern for the clusters made by DBSCAN. However, this might be due to the non-linear nature of the data. Figure C.6 and C.7 are 3D plots, where Figure C.6 are the most positively correlated features to price (x- and y-axis) against price (z-axis) and Figure C.7 are the most negatively correlated features (x- and y-axis) to price against price (z-axis). While there is some pattern in the data appearing in Figure C.7, most of the observations are displayed as noise. Thus, the best model to display clusters in the data is the KMeans with three clusters.

Principal Component Analysis (PCA)

Overfitting is problematic when including too many features in the machine learning models, leading to slower computation and more inaccurate estimates. To address this problem, dimensionality reduction is utilized, to preserve important information while reducing unrelated features. PCA reduces the dimensionality by uncovering the most unrelated features, and how much they represent the total variance. Considering our dataset relatively small with approximately 90,000×9 observations we use the default PCA estimator from sklearn for dimensionality reduction.

First, we estimated a PCA of 2 components, since first two PCs typically capture the majority of the dataset's variance. PC1 was heavily influenced by the features '*availability_365*', '*calculated_host_listings_count'*, and '*number_of_reviews*', representing the listing popularity or availability. PC2 had high influence from '*number_of_reviews'*, '*room_type_Entire home/apt'*, and '*minimum nights*', which capture booking frequency and property type differences. The PCA with two components can be found in the Jupyter notebook attached with this report.

When we want to retain the top '*n*' principal components that explains at least 95% of the variance, we use the PCA and find the explained variance ratio. First, by fitting PCA to the dataset, purposely ignoring to specify the number of components, the model decides the explained variance by each PC. The optimal '*n*' is then found using *np.argmax(cumsum >= 0.95) + 1,* identifying the least amount of components required to preserve 95% of the total variance. In our case, the PCA suggests that we should have 6 features as the minimum number of dimensions to preserve 95% of the variance. This can be seen in Figure D.1 in the Appendix, where the optimal components are found by the elbow in the curve where the explained variance stops growing fast.

Introducing a new feature as a linear combination of the existing features, is simply redundant and leads to overfitting. The focus of a Principal Component Analysis is to reduce dimensionality, while incorporating a new feature as a linear combination of existing features, work counter intuitive through increased dimensionality with no new information. Therefore, it is not a good idea to. Although introducing this new feature do not add any new information, the principal components will be affected to account for the new feature, changing its directions. Nevertheless, the structure remains the same, and in some cases where the new feature is highly correlated to any existing PCs, they might line up similarly. Explained Variance is the total variance in each PC, as the new feature is just a combination of the existing ones, the total variance is not affected.

Given that the optimal number of principal components are six, when wanting to preserve 95% of the variation, we ran DBSCAN and KMeans again with these new features added to the models. The PCs form a new basis in which the dataset is expressed. These components are statistically uncorrelated and help in reducing dimensionality while preserving the most significant variation in the data. From the Jupyter notebook, we uncovered that 50% of the variation is explained by the first two principal components. When including six principal components we preserve approximately 97.83% of the variation in the features.

Table D.1 summarizes the silhouette scores of the models run with selected features based on correlation and with the six principal components. From this table, we see that the PCA improves the silhouette score for both KMeans and DBSCAN. But only by a little. Thus, to avoid overfitting, we chose KMeans with features selected based on the correlation as the final model.

Conclusion

In conclusion, based on the manually built correlation function, we discovered that the features positively correlating with price included as follows; '*availability_365*', '*room_type Entire home/apt*', '*calculated_host_listings_count*', '*neighbourhood_group city_center*'. On the other hand, the features that negatively correlated with price the most were '*minimum_nights*', '*longitude*', '*room_type Private room*', '*number_of_reviews*', '*neighbourhood_group outer_city*'. Furthermore, these features were chosen for finding clusters in the Airbnb data. With the KMeans model we identified three clusters based on these selected features, colored in blue, pink and yellow on the plots. The clusters can be summarized as follows:

- Cluster 0 (Blue): The first cluster is characterized by listings with fewer host-managed properties. These listings have fewer minimum nights, indicating that they cater to a wider segment travelling in Paris. Prices for listings in this cluster are both in the high and low end.
- Cluster 1 (Pink): This group contains listings with higher host activity, meaning that they likely are managed by experienced Airbnb hosts with multiple properties. These listings tend to have moderate prices with a wide range too.
- Cluster 2 (Yellow): The last cluster, consists of listings with availability throughout all season and having higher amounts of minimum nights required, indicating a group of listings which is possibly target for people staying in Paris for a longer time than the Olympics.

Appendix

A) Manually built code for covariance & correlation (Question 1.1)

```
import numpy as np
import pandas as pd

def cov_corr_function(df, feature):
    numerical_df = df.select_dtypes(include=[np.number])

    if feature not in df.columns:
        raise ValueError(f"Variable '{feature}' not found in DataFrame
columns.")

    if feature not in numerical_df.columns:
        raise ValueError(f"Variable '{feature}' is not numerical.")

    column1 = numerical_df[feature].tolist()
    feature_mean = sum(column1) / len(column1)

    feature_std = (sum((value - feature_mean) ** 2 for value in
column1) / len(column1)) ** 0.5

    results = {}

    for col in numerical_df.columns:
        if col != feature:  # Compare feature with all others
            compare_col = numerical_df[col].tolist()
            compare_mean = sum(compare_col) / len(compare_col)
            compare_std = (sum((value - compare_mean) ** 2 for value in
compare_col) / len(compare_col)) ** 0.5

            covariance = sum((column1[i] - feature_mean) *
(compare_col[i] - compare_mean)
                             for i in range(len(column1))) /
len(column1)

            correlation = covariance / (feature_std * compare_std) if
feature_std > 0 and compare_std > 0 else 0

            results[col] = {'Covariance': covariance, 'Correlation':
correlation}

    result_df = pd.DataFrame.from_dict(results, orient='index')

    return result_df
```

B) Findings from visualization function and other plots (Question 1.2)

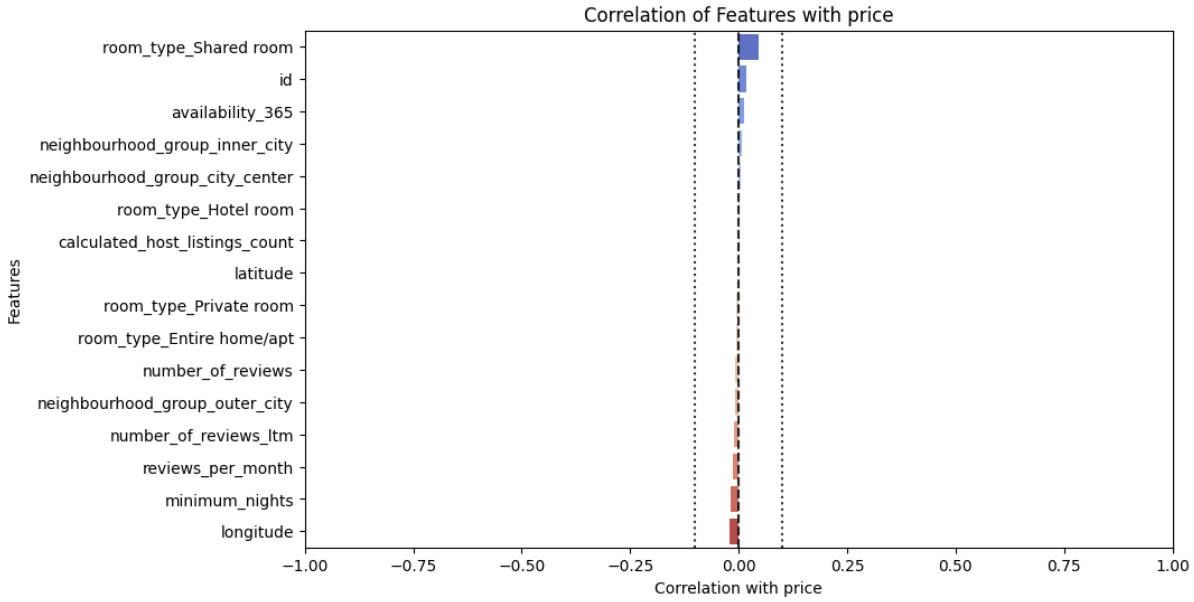

Figure B.1: Correlation with price without transformation.

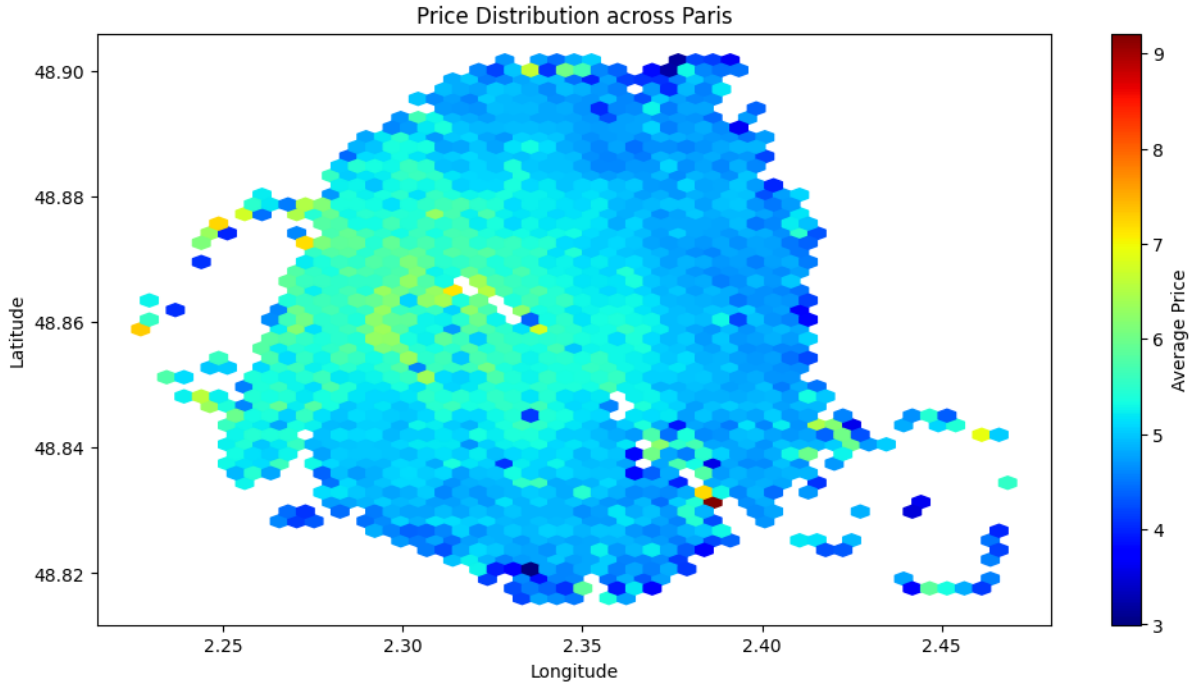

Figure B.2: Price distribution plot across longitude and latitude.

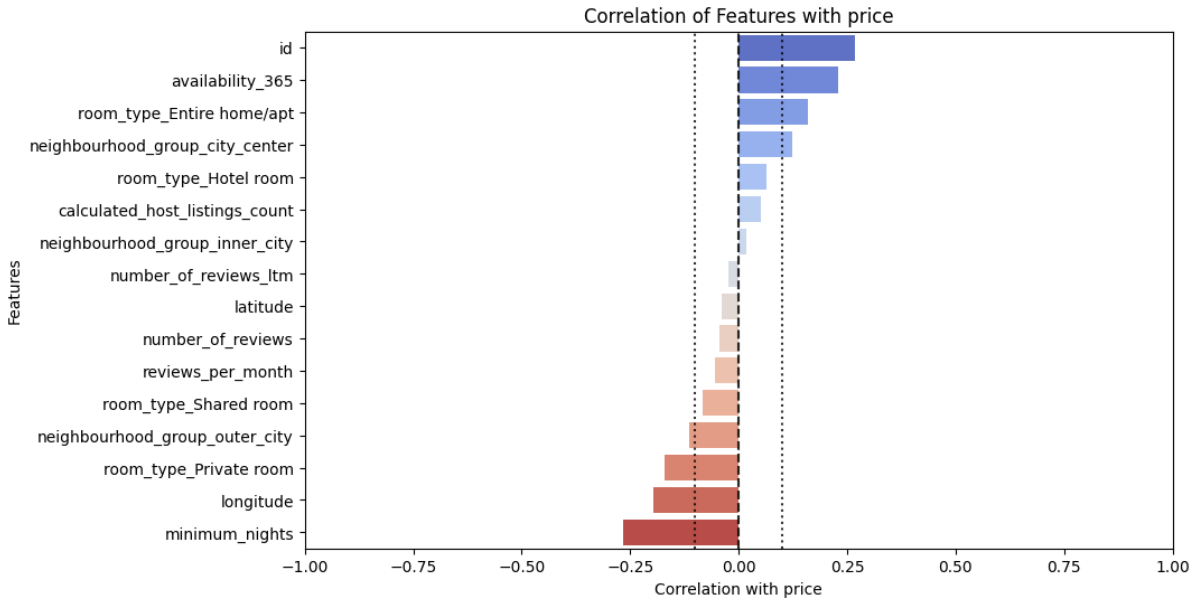

Figure B.3: Correlation coefficients after transformation and scaling.

C) Findings from Cluster Analysis (Question 2)

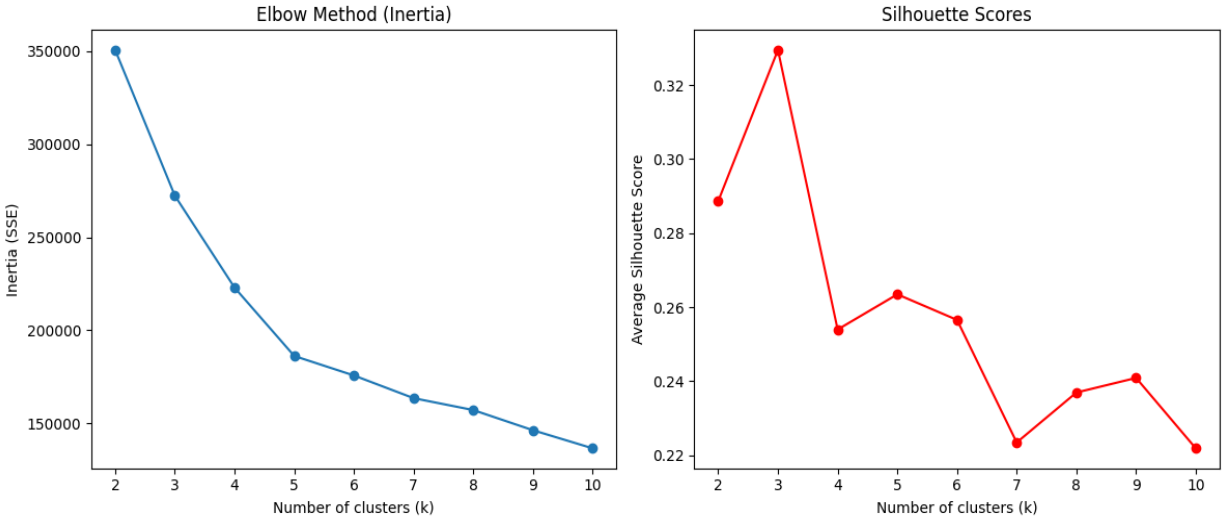

Figure C.1: Elbow and Silhouette score for KMeans-model with selected features.

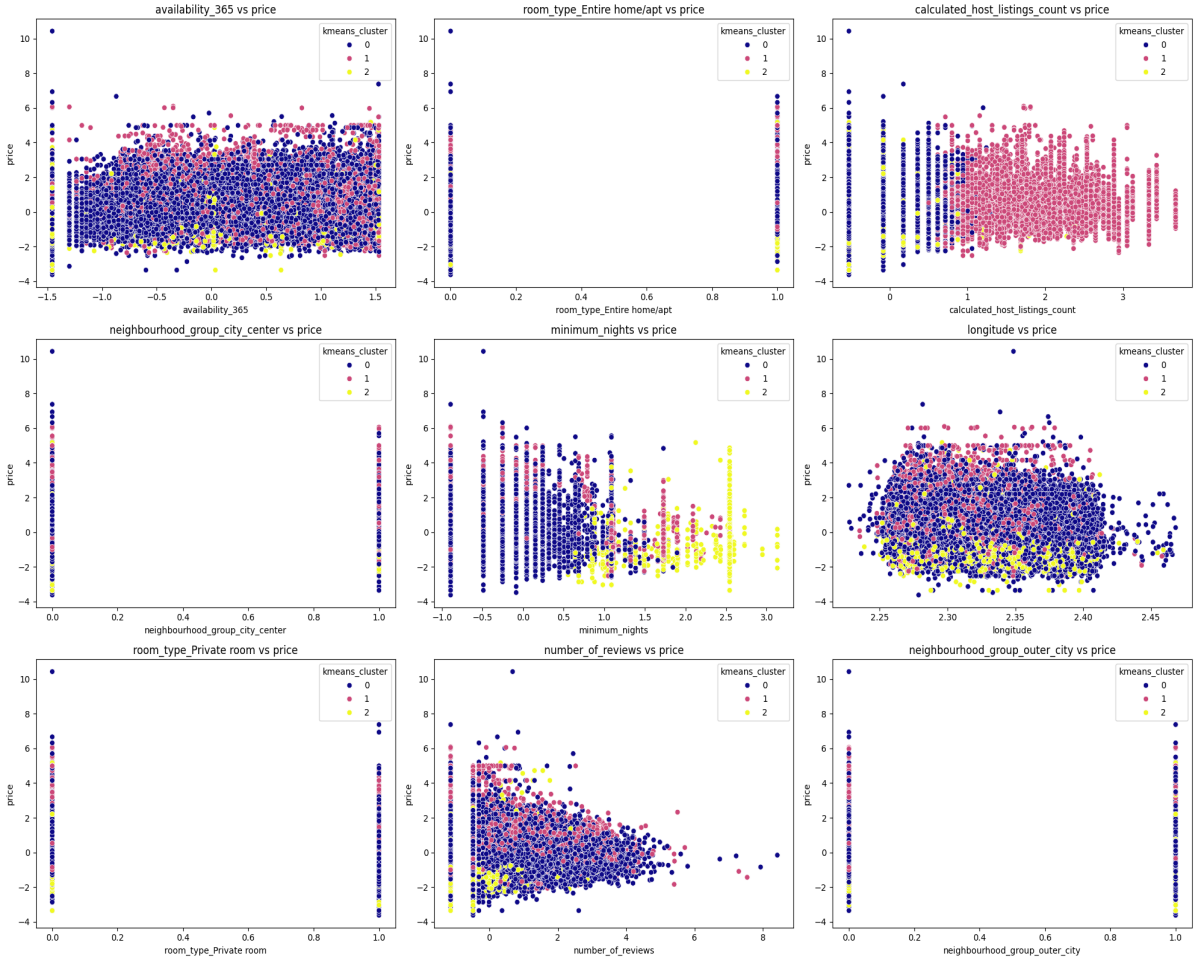

Figure C.2: Pairwise plot for KMeans clusters.

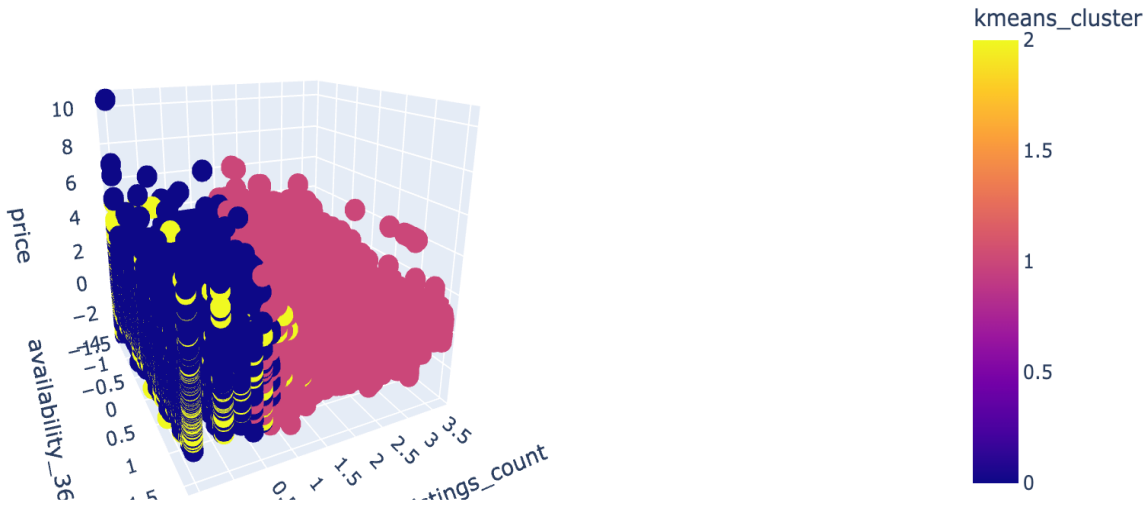

Figure C.3: 3D plot of most positively correlating features with price in KMeans.

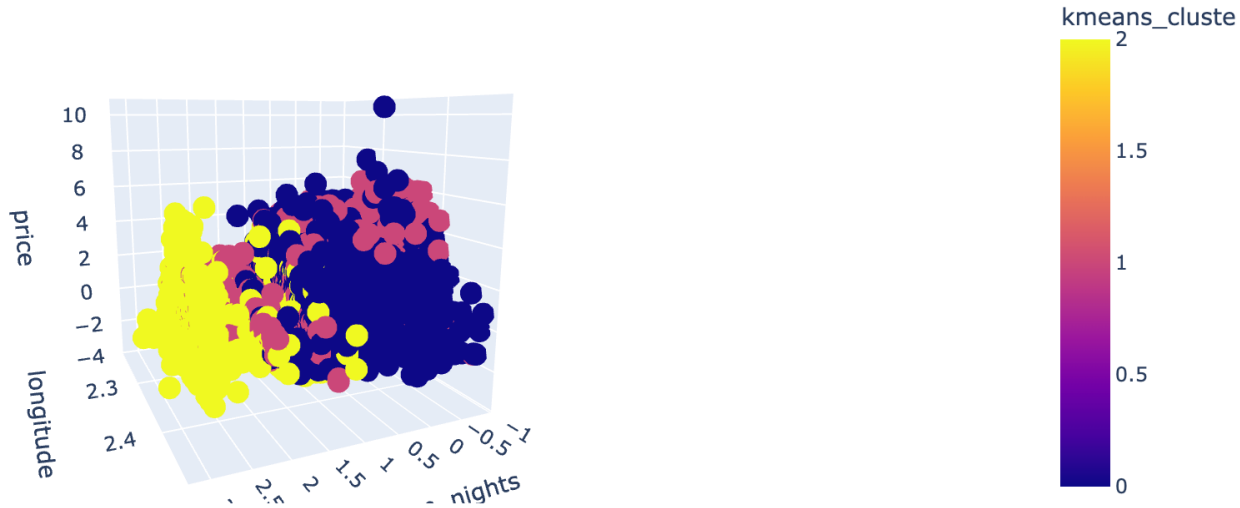

Figure C.4: 3D plot of most negatively correlating features with price in KMeans.

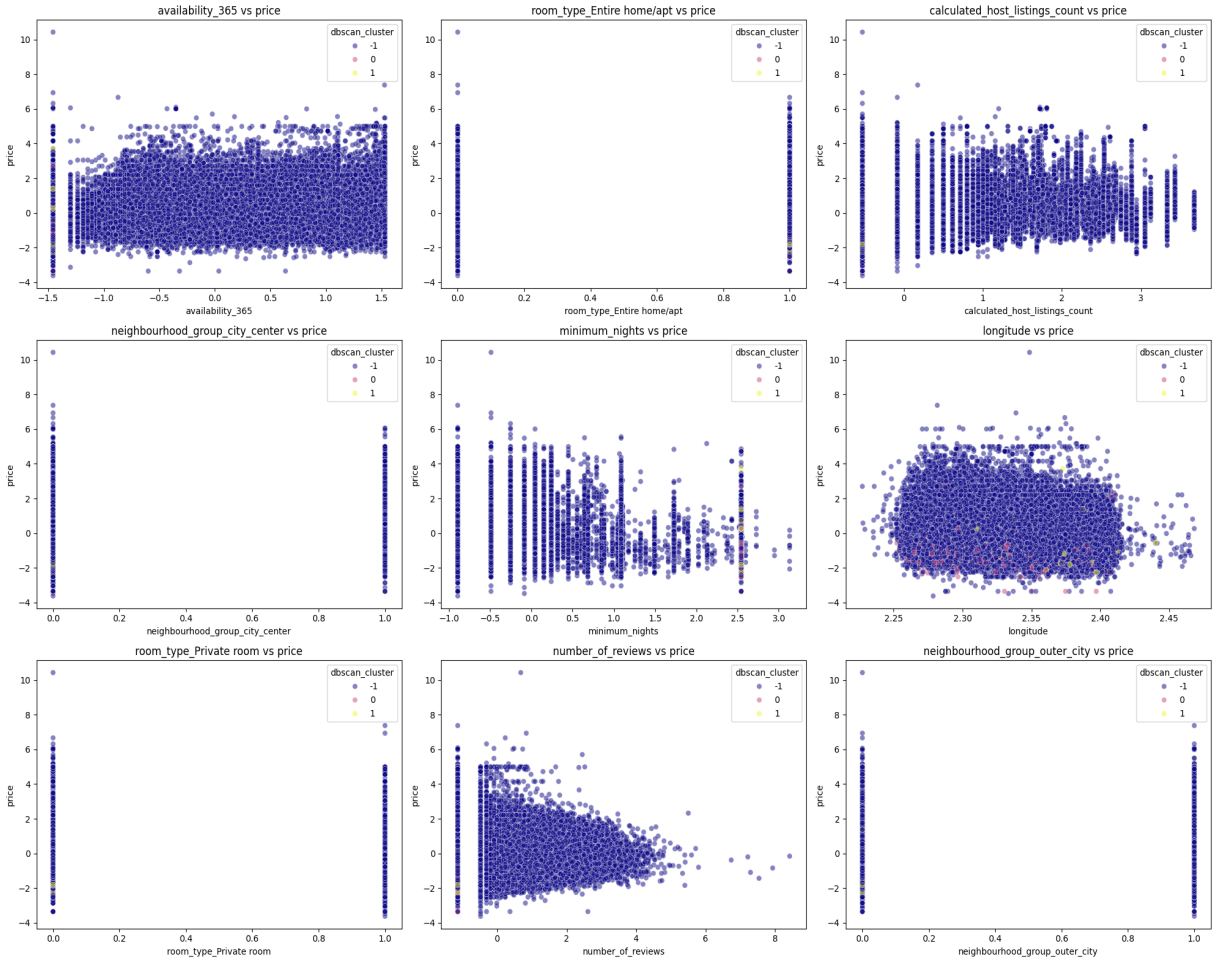

Figure C.5: Pairwise plot for DBSCAN clusters.

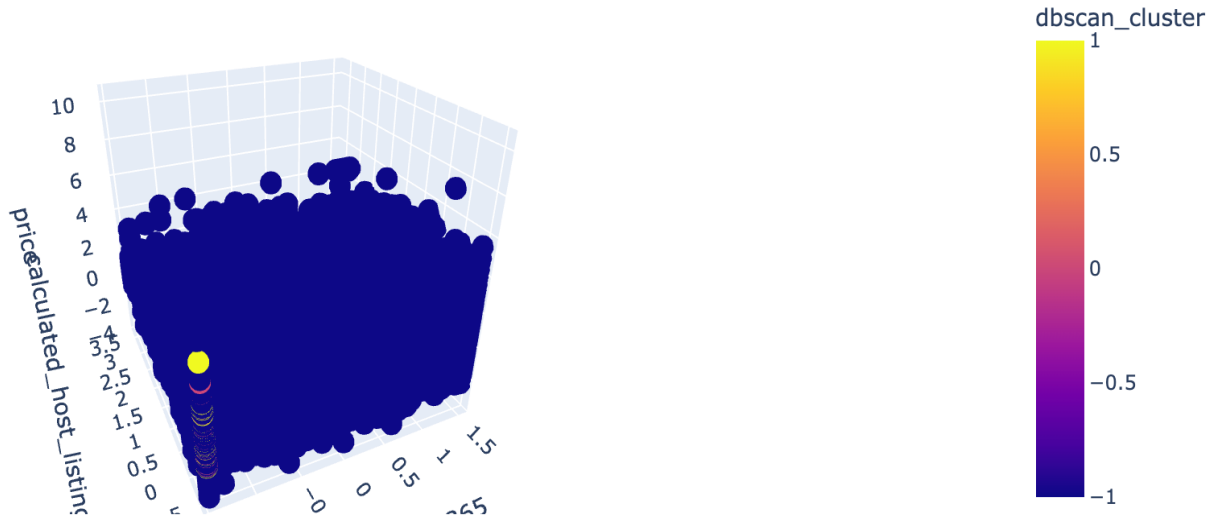

Figure C.6: 3D plot of most positively correlating features with price in the DBSCAN model.

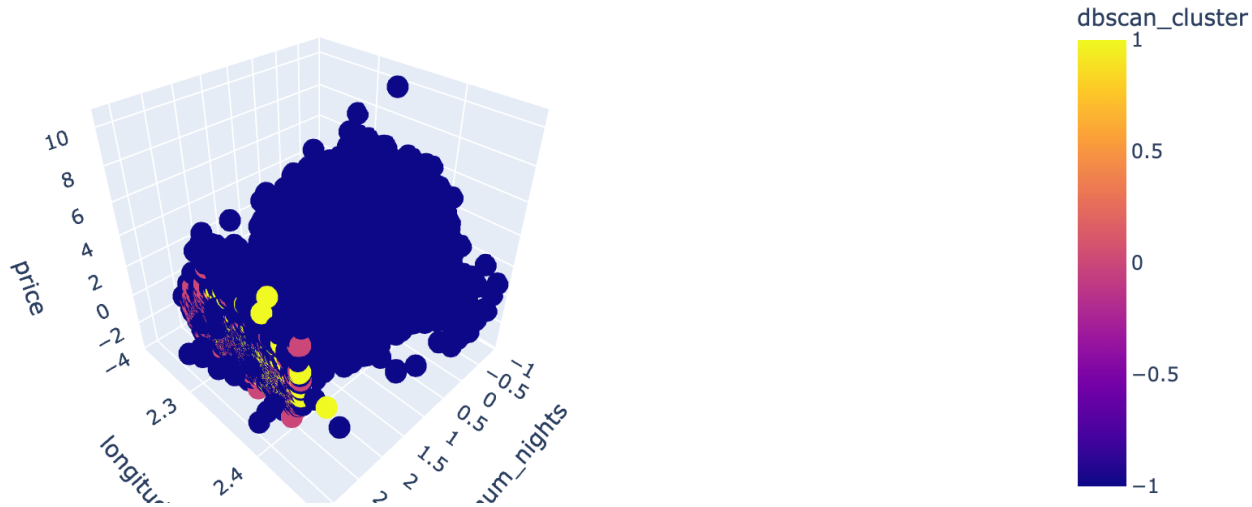

Figure C.7: 3D plot of most negatively correlating features with price in the DBSCAN model.

D) Principal Component Analysis (Question 3)

	KMeans	DBSCAN
Selected features	0.3295	0.2498
PCA features	0.3375	0.2656

Table D.1: Approximate silhouette scores for all clustering models.

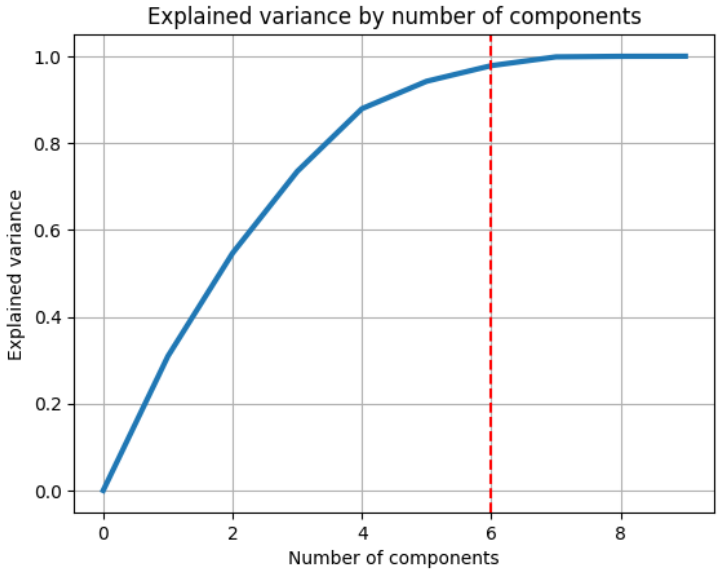

Figure D.1: Optimal Principal Components.