**Open Source SW Contribution**

**1. combination of various data scaling and categorical features encoding methods.**

9 combinations of each encoding, scaling methods.

One\_hot encode - standard scale

- min-max scale

- robust scale

Label encode - standard scale

- min-max scale

- robust scale

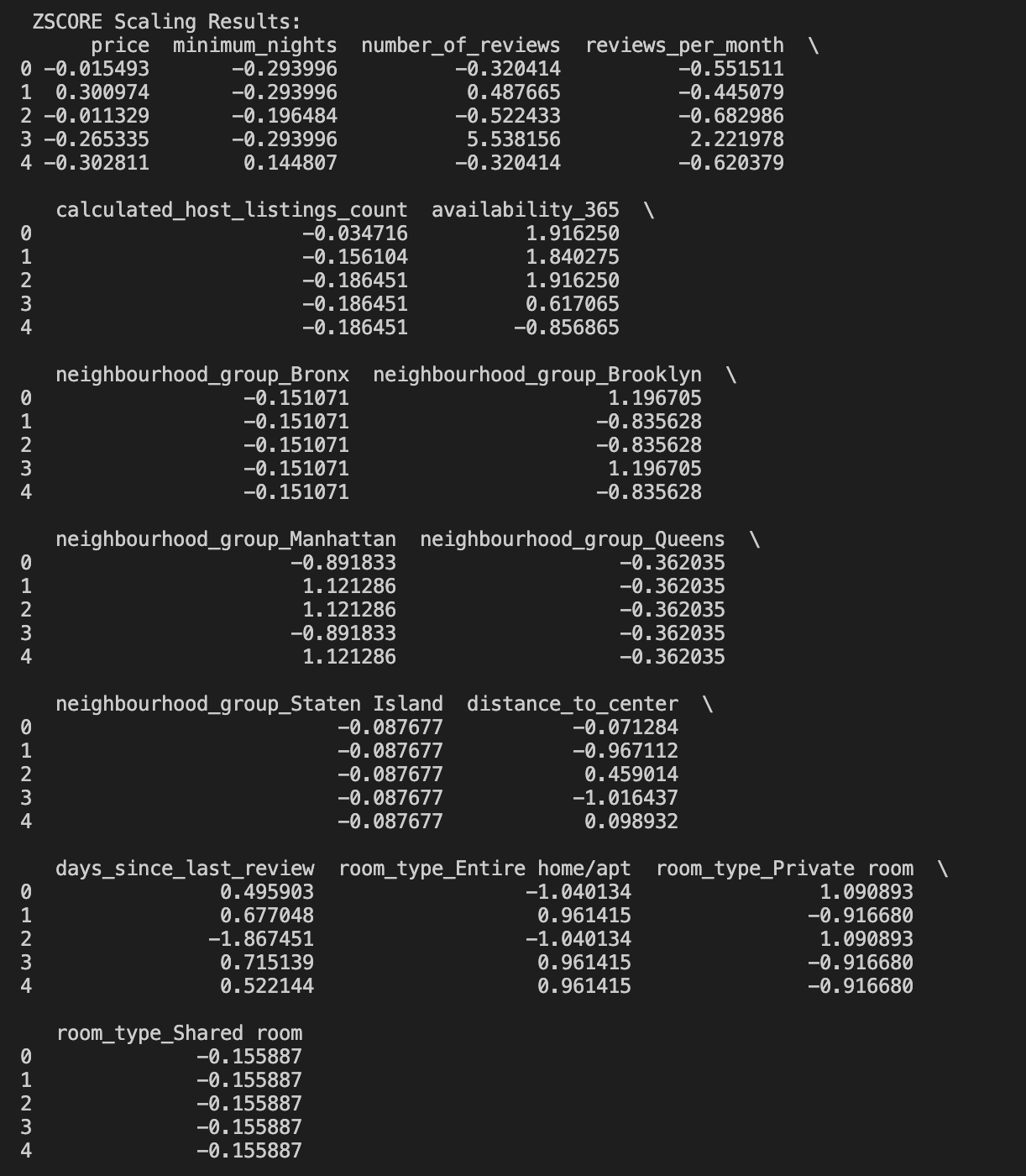
Ordinal encode - standard scale

- min-max scale

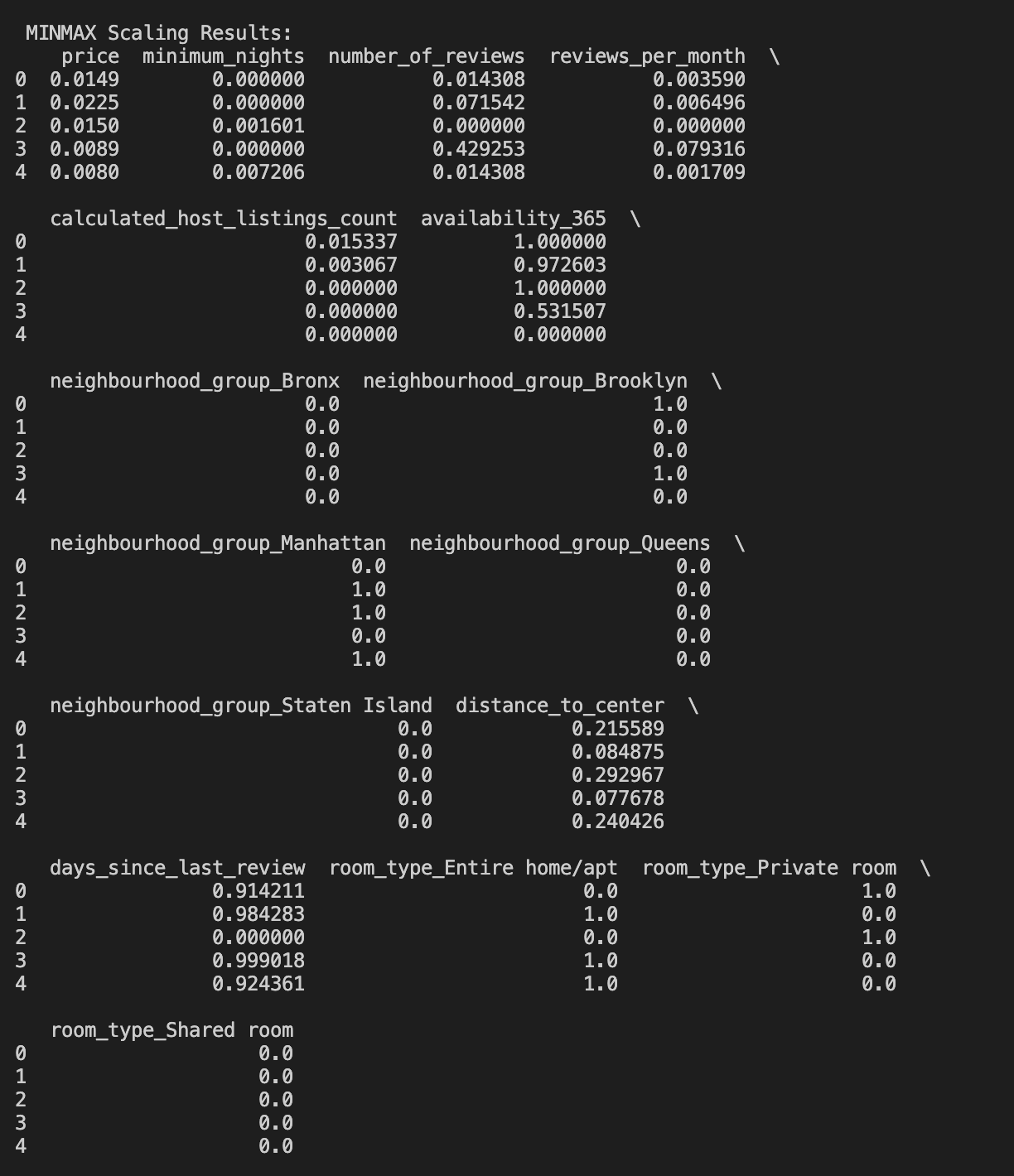
- robust scale

There are results of each method combinations below.

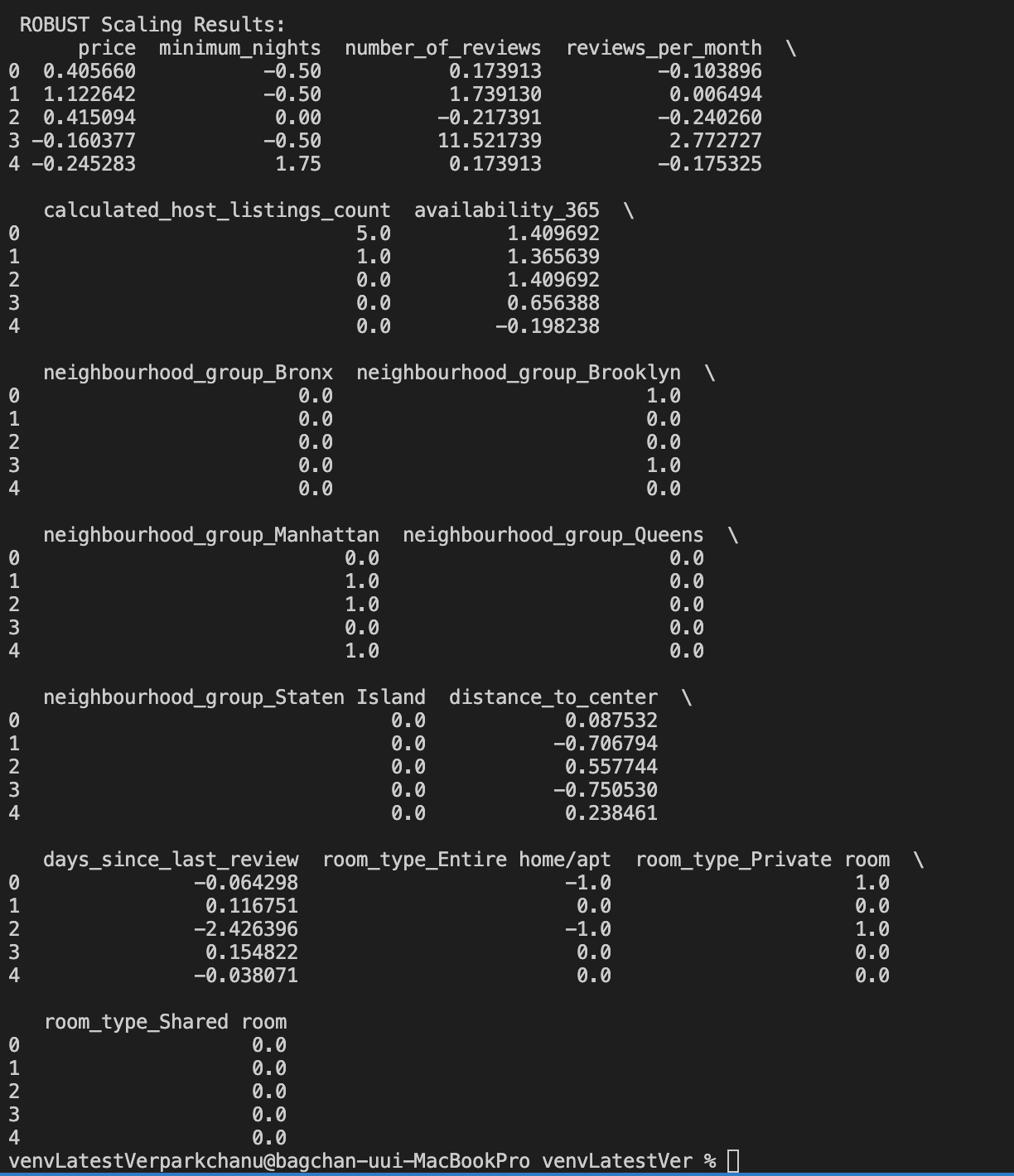
One-hot encode & Standard(Z-score) scale:



One-hot encode & Min-max scale:

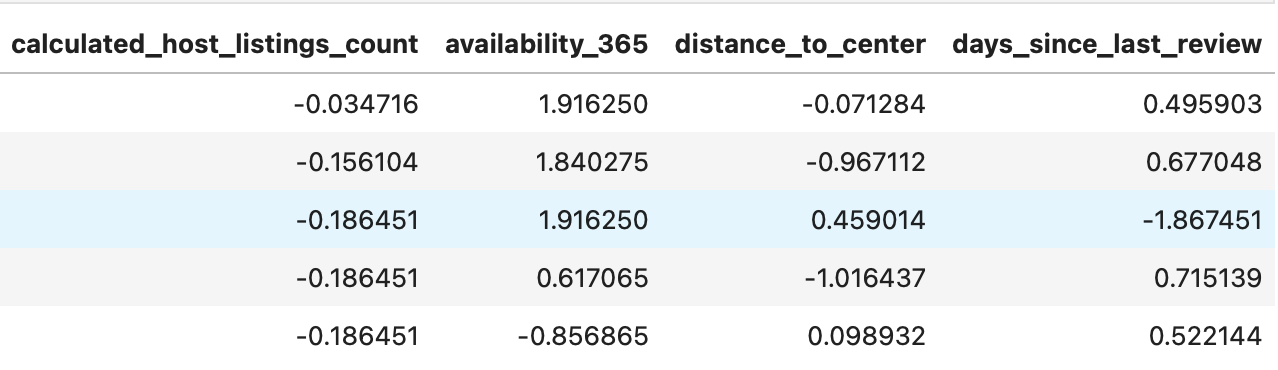


One-hot encode & Robust scale:

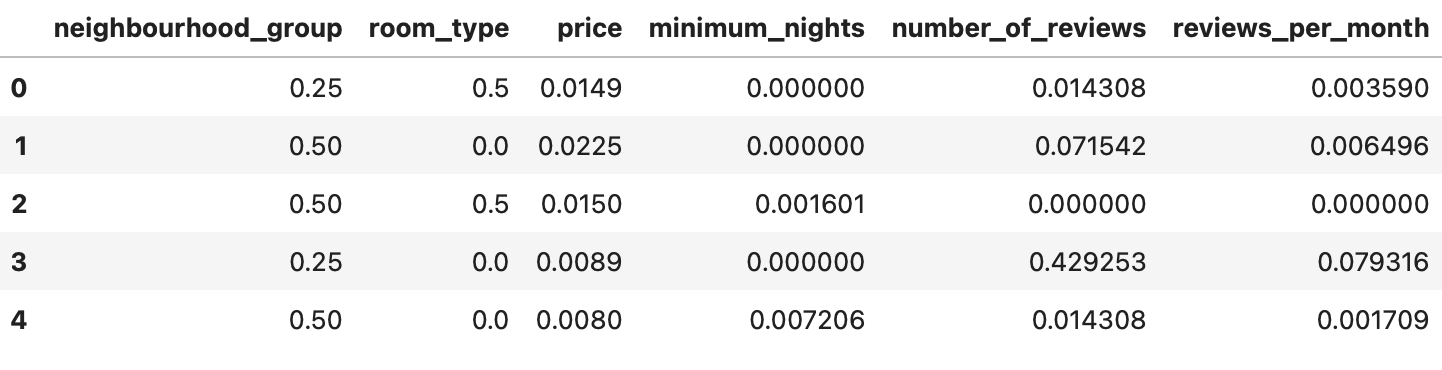


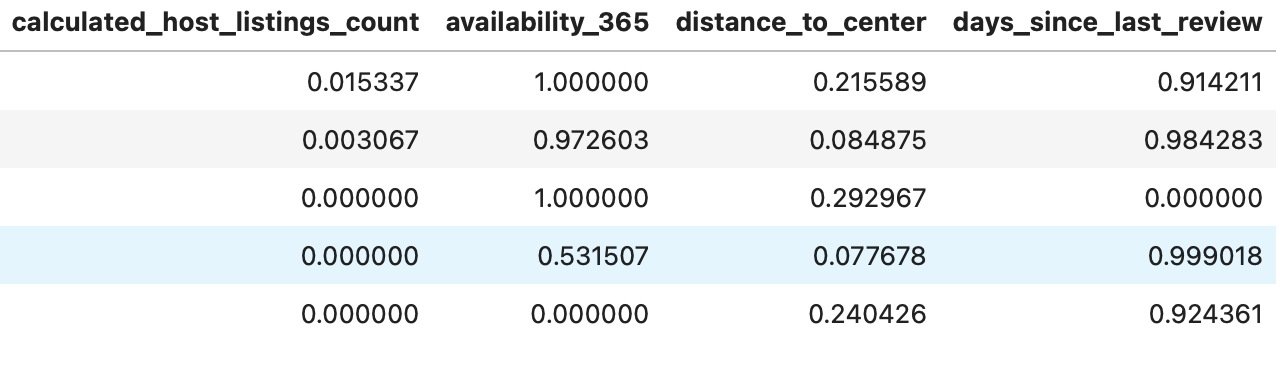
Label encode - standard scale:





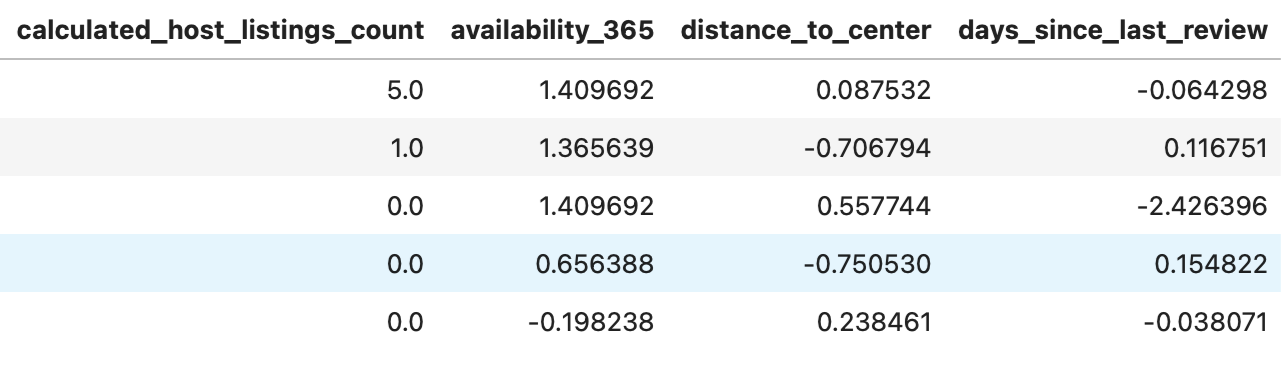
Label encode – Min-max scale:



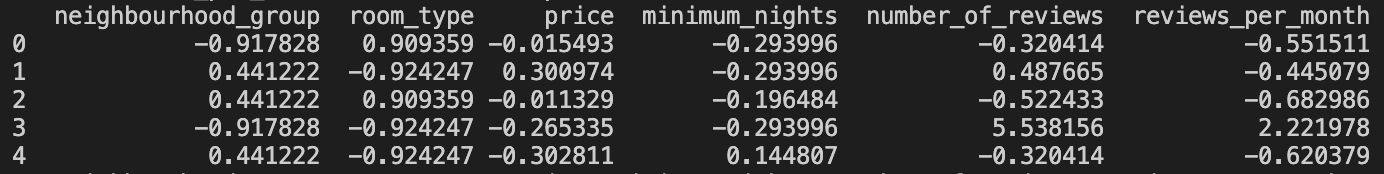


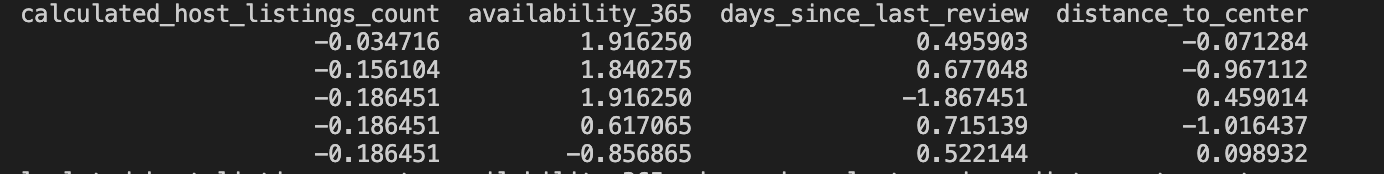
Label encode – Robust scale:



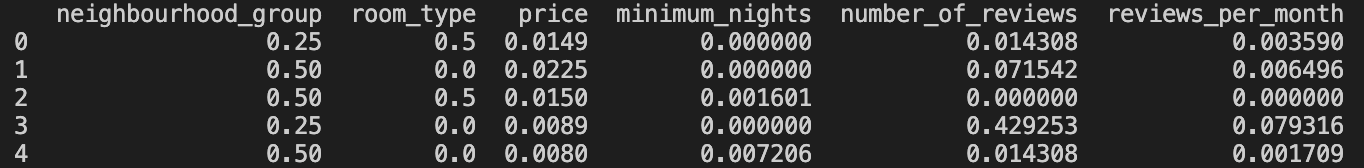


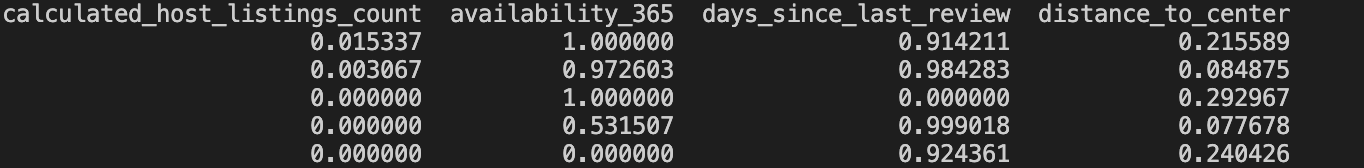
Ordinal encode - standard scale:



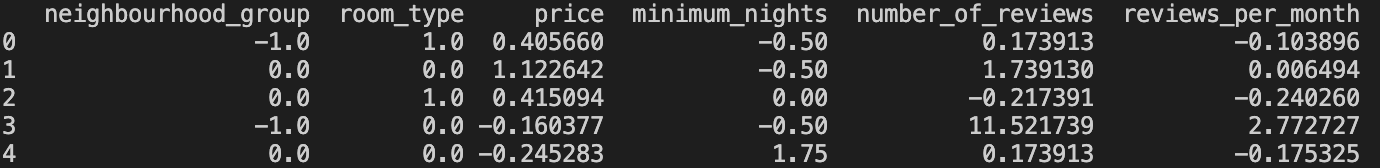


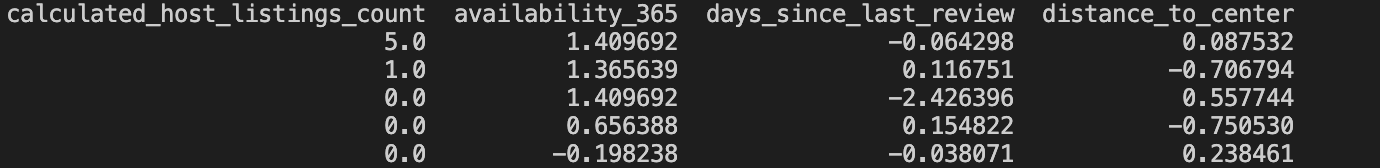
Ordinal encode – Min-max scale:





Ordinal encode – Robust scale:





From the above results, we can see that the label encoding and the ordinary encoding methods produce the same results. This seems to be because both methods map categorical data into integers in advance and start encoding. Therefore, we considered the label encoding and the Ordinal encoding to be the same, and compared the results of the one-hot encoding with a total of six results of the label encoding.

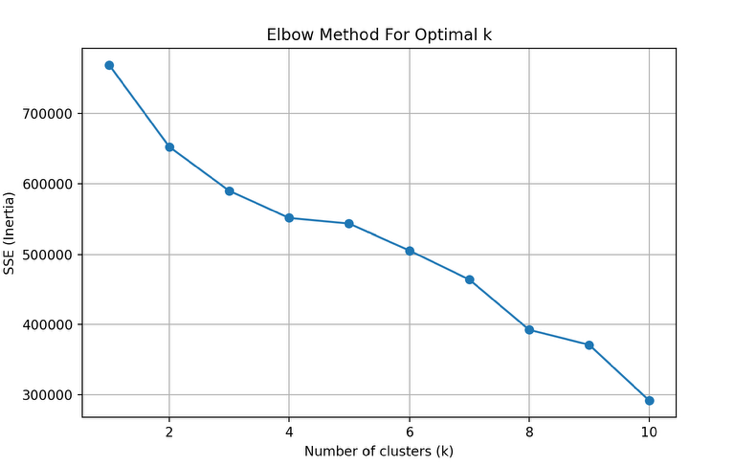
Therefore, we considered the label encoding and the Ordinal encoding to be the same, and compared the results of the one-hot encoding with a total of six results of the label encoding.

**2. Learning Model training and testing.**

**main\_host**

**1. KMeans Clustering**

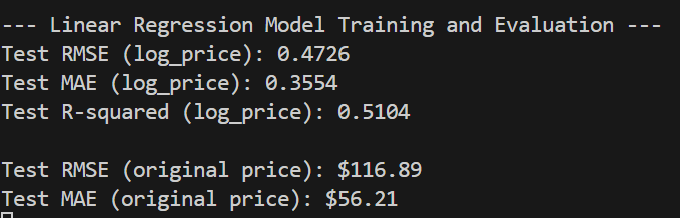
The KMeans algorithm was applied to the Airbnb NYC dataset with **k=4clusters**, selected based on the Elbow Method. **Standardized features** were used to ensure effective distance-based clustering. The resulting cluster labels were added to the original dataframe as a new feature **(cluster)**, which was later used in regression modeling. The centroid values of each cluster provided interpretable insights into the data distribution and contributed to enhancing the explanatory power of subsequent models.



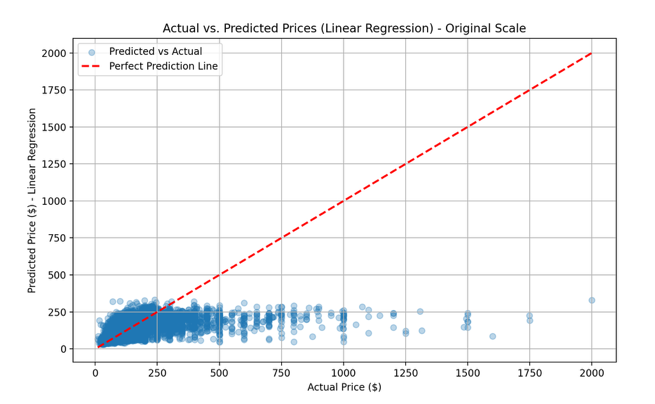
**2. Multiple linear regression**

The Multiple Linear Regression (MLR) model served as a baseline approach for predicting log-transformed Airbnb prices. The model was trained on all available features, including the engineered clusterand distance variables. Performance metrics and plots suggest that while the model captured some meaningful relationships, its predictive capacity was constrained by the linear nature of the algorithm.

**◼ Performance Overview:**

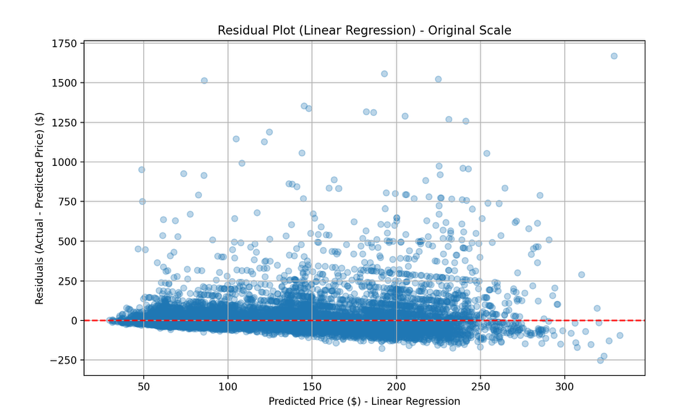


**◼ Actual vs. Predicted Price Visualization:**



The scatter plot comparing actual and predicted prices shows a wide spread around the ideal prediction line (red dashed line). Most predictions are tightly grouped between $0 and $400, even when actual prices go as high as $2,000. This highlights the model’s inability to extrapolatefor high-price listings and a general tendency to underpredict expensive units.

**◼ Residual Analysis:**



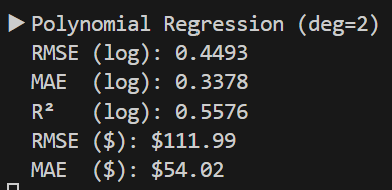
- A dense cloud of residuals clusters around low predicted prices, with large positive residuals (underpredictions) as actual prices increase.

- The residuals appear heteroscedastic, meaning the prediction error increases with the predicted value — a typical sign of model misspecification in linear regression.

- Additionally, the slight funnel shape in the residuals implies that a nonlinear model might capture the variability better.

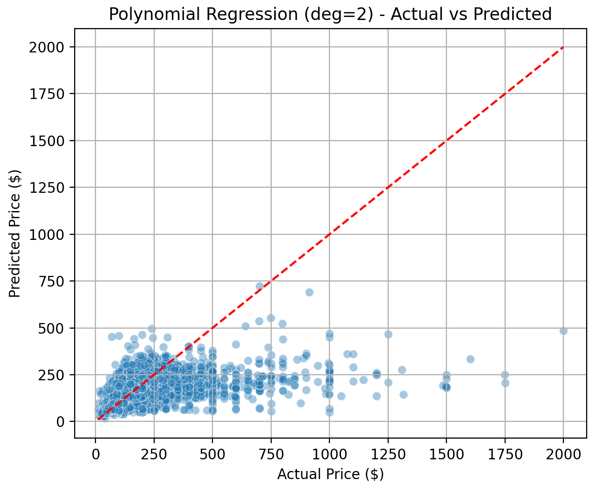
**3. Polynomial regression**

**◼ Performance Overview**:



Compared to Multiple Linear Regression, the Polynomial model shows a noticeable improvement in all key metrics. The higher R² indicates a better fit to the log-transformed data, and the reduced error values in dollar terms reflect improved accuracy in price prediction.

**◼ Actual vs. Predicted Price Visualization:**



The scatter plot of actual vs. predicted prices (in original scale) demonstrates better alignment with the red dashed reference line (ideal prediction). However, a similar pattern to linear regression remains: predictions tend to be more tightly clustered below $500, with the model struggling to accurately predict high-end listings. Nonetheless, there are more accurate high-price predictions than in the linear model, indicating that the added nonlinear terms provided some benefit.

**◼ Interpretation & Limitations:**

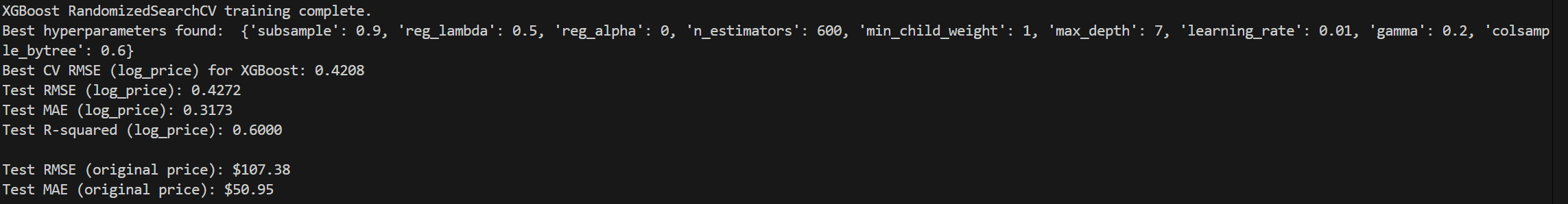
- The polynomial model captures curvilinear trends and some interaction effects, enhancing performance without drastically increasing model complexity.

- That said, performance at the extreme ends of the price distribution still lags. Outliers and heteroscedasticity remain issues.

- Also, as the degree of the polynomial increases, the risk of overfittinggrows. Degree=2 was chosen as a balance between model expressiveness and generalization.

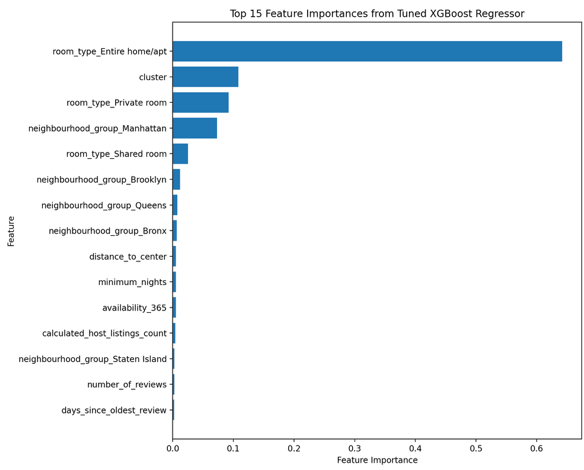
**4. XGBoost Regressor**

**◼ Performance Overview:**



These results clearly outperform the linear and polynomial models, especially in terms of generalization accuracy and error reduction. The log-scale R² of over 0.60 indicates strong predictive capability.

**◼ Feature Importance Insights:**

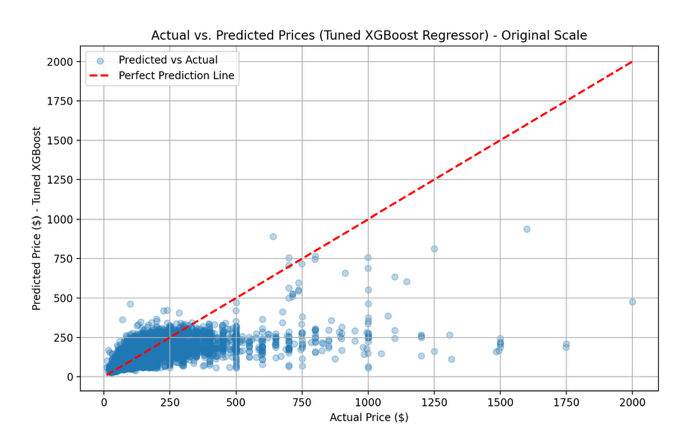


- room\_type\_Entire home/apt is by far the most important predictor, followed by cluster, room\_type\_Private room, and neighborhood-related variables.

- The cluster feature generated via KMeans clustering played a crucial role, validating the effectiveness of combining unsupervised learning with supervised modeling.

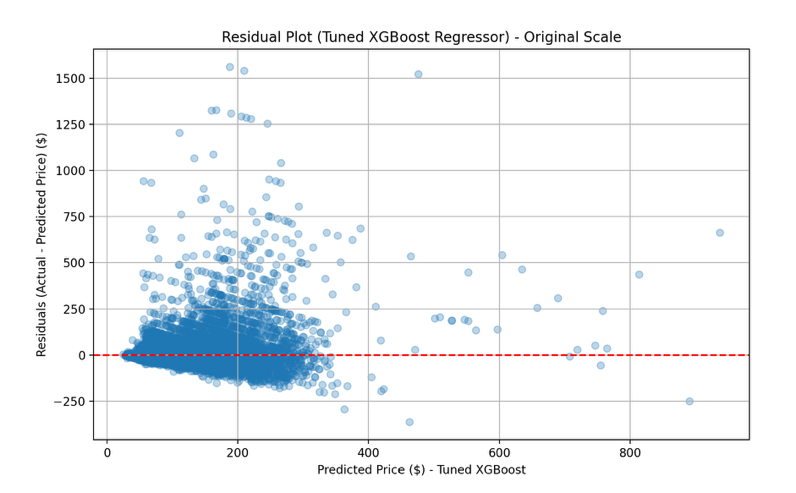
- Engineered features like distance\_to\_center and days\_since\_oldest\_review were also used but had relatively lower influence.

**◼ Actual vs. Predicted Price Visualization:**



The scatter plot comparing actual and predicted prices shows a much tighter clustering along the diagonal reference line (red dashed), especially in the $0–500 range. Unlike the linear and polynomial models, the XGBoost regressor correctly models more of the variation in higher-price listings, although some underprediction still occurs at the top end. This plot confirms that the model has learned a more realistic and flexible mapping from features to prices.

**◼ Residual Analysis:**



- Residuals remain close to zero across a wider range of predicted values.

- The spread of residuals is narrower overall, suggesting reduced error variance.

- Some heteroscedasticity still exists (i.e., larger residuals at higher predicted prices), but it's significantly mitigated compared to linear methods.

**5. Conclusion: Model Selection**

For the NYC Airbnb price prediction task, **the combination of log transformation, distance-based features, KMeans clustering, and a tuned XGBoost regressor** proved to be highly effective.  
XGBoost not only achieved the l**owest RMSE and MAE** but also demonstrated robust generalization capabilities, as seen in cross-validation and residual analysis.

Additionally, the model provided **interpretability through feature importance analysis**, which confirmed that variables like room\_type, cluster, and neighbourhood were key drivers of price.

**Therefore, the tuned XGBoost model was selected as the final model due** to its:

- Superior accuracy and reliability,

- Ability to handle nonlinearity and feature interactions,

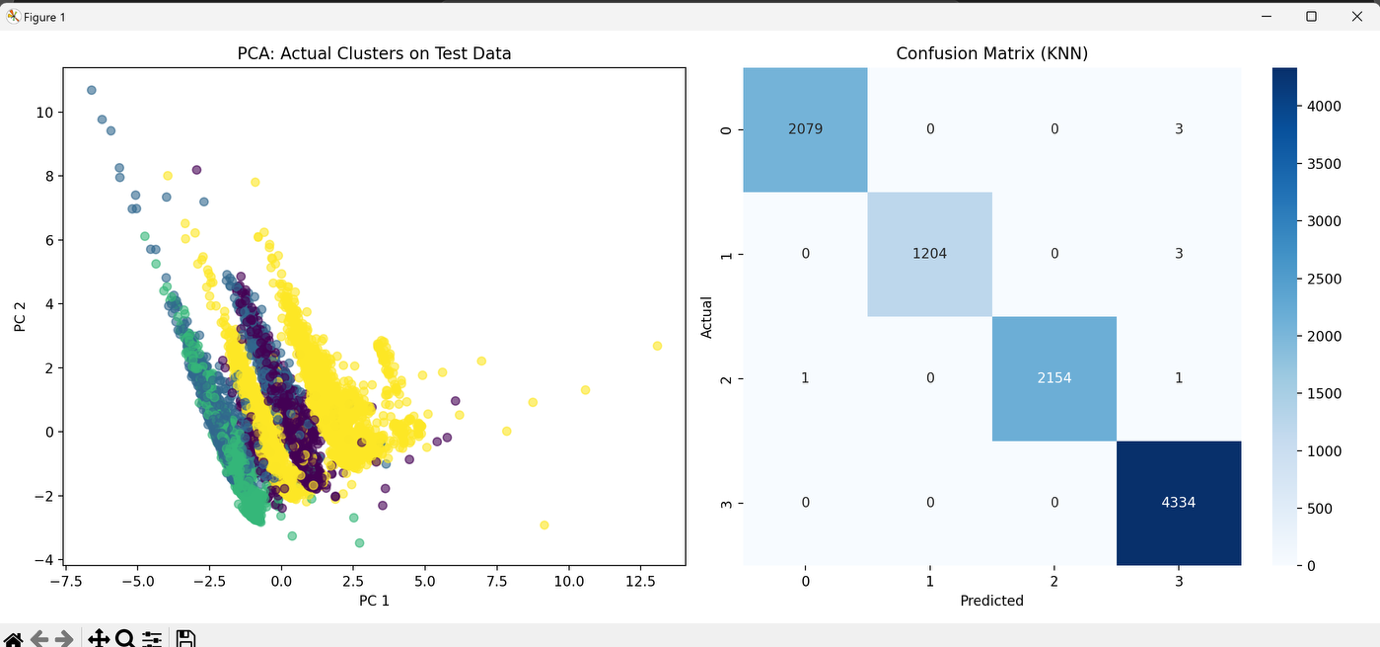
- Integration of both raw and engineered features,

- Strong generalization on unseen data.

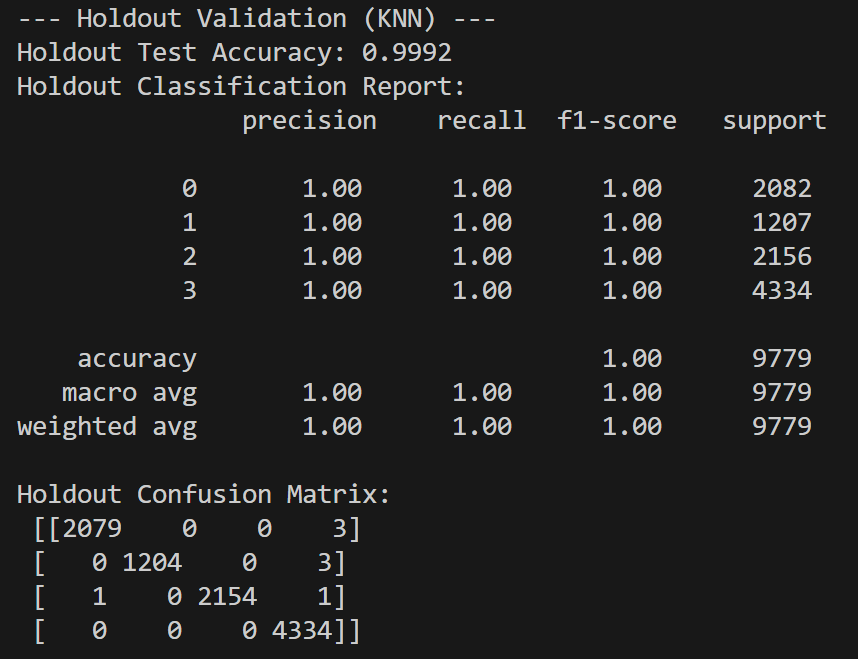
**main\_user**

Classification

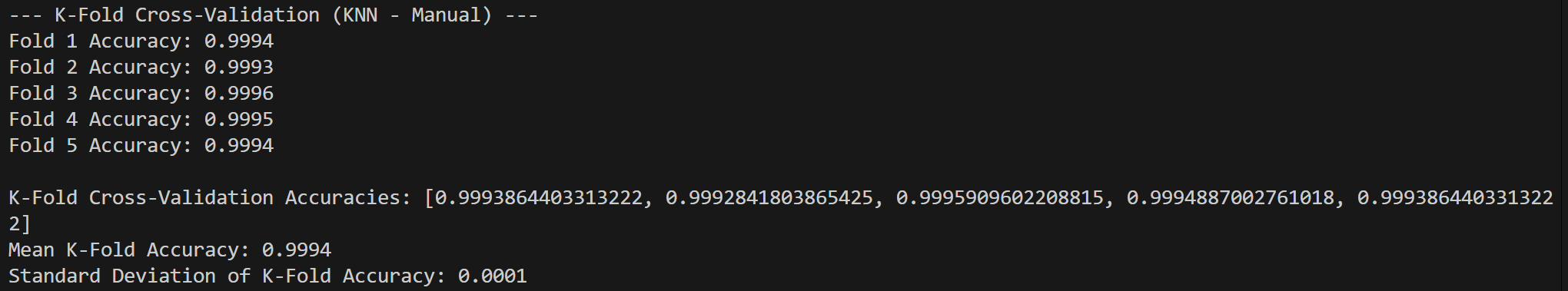
* k-nearest neighbors



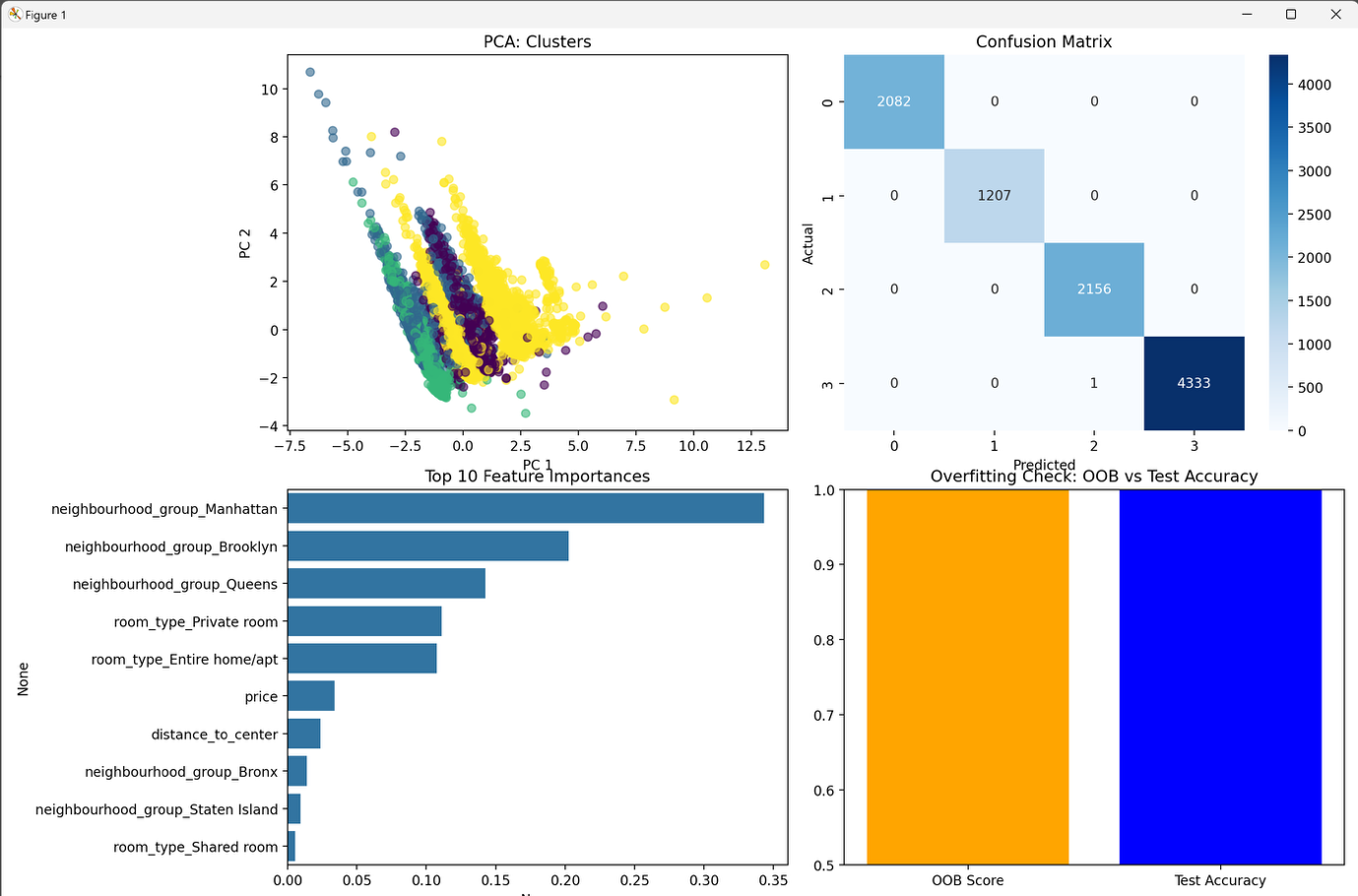
* holdout



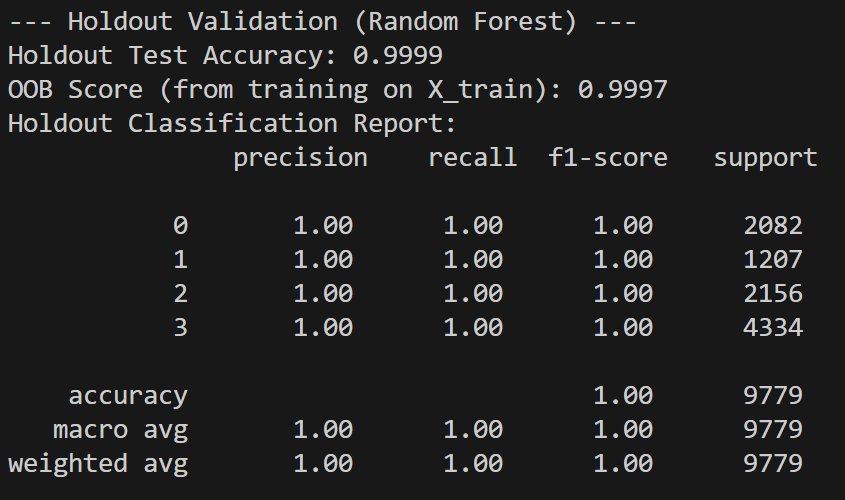
* k-fold



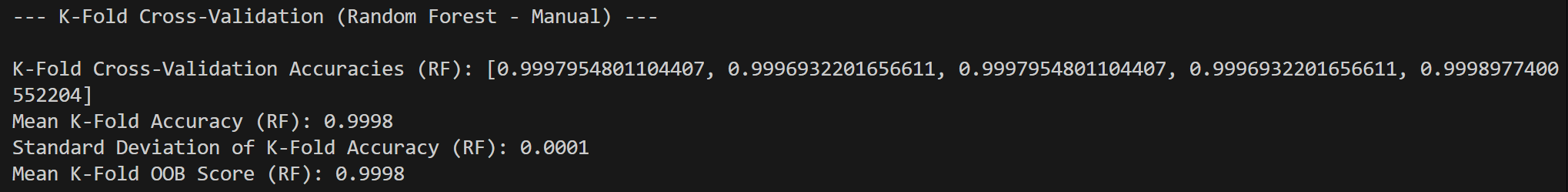
* Random Forest(RandomizedSearchCV: with best hyperparameter combination)



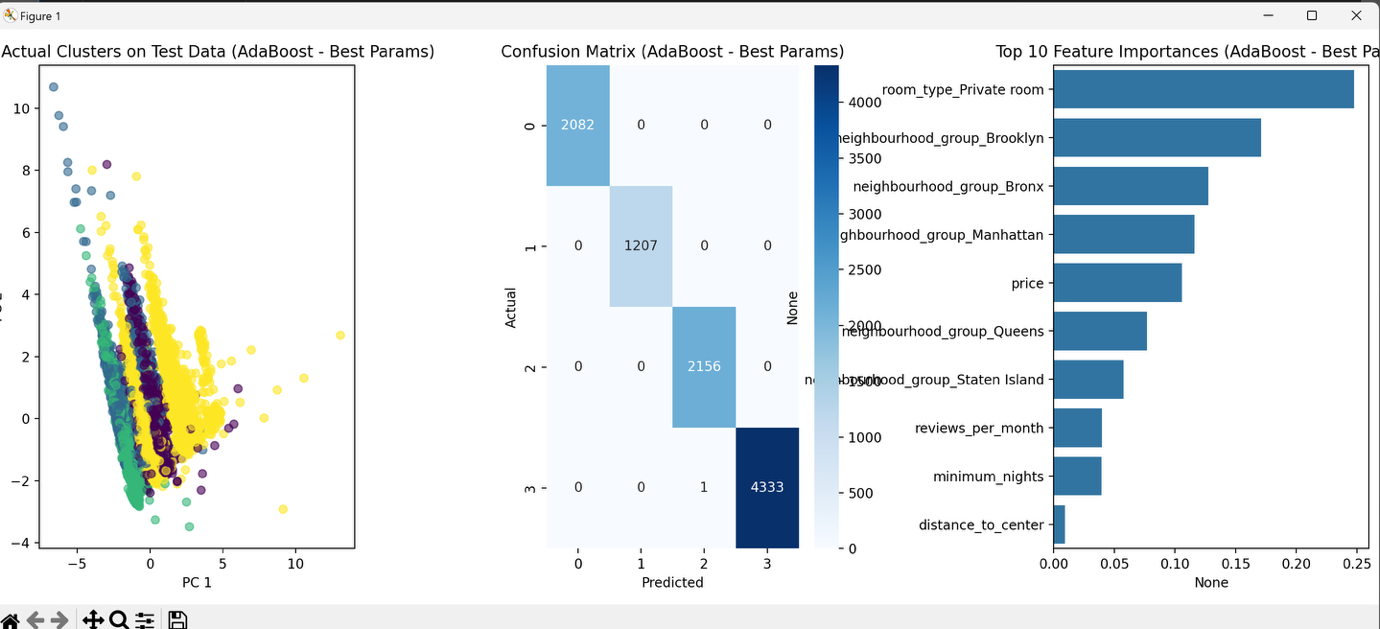
* holdout



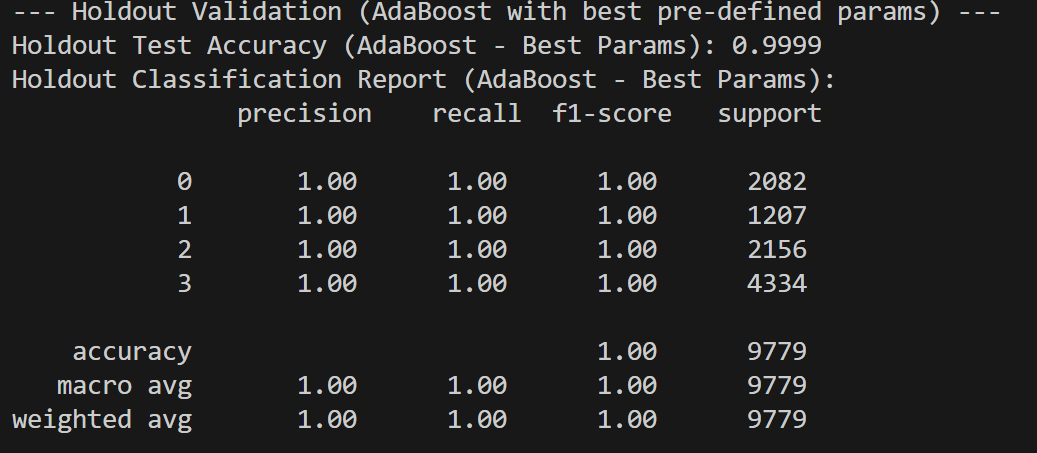
* k-fold



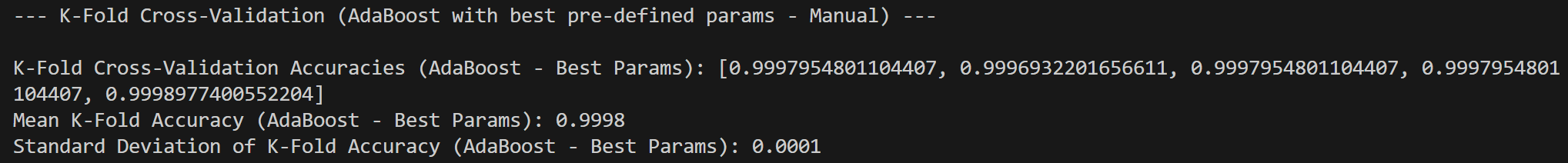
* Adaboost(RandomizedSearchCV: with best hyperparameter combination)



* holdout



* k-fold



1. **K-Nearest Neighbors (KNN):**

* The KNN algorithm recorded relatively lower accuracy compared to ensemble algorithms like Random Forest and AdaBoost. This could be attributed to KNN's reliance on local neighborhood relationships rather than learning the overall data structure or complex boundaries, and its performance can also degrade in high-dimensional data.

1. **Random Forest (RF) and AdaBoost:**

* Both Random Forest and AdaBoost algorithms achieved very high accuracy (e.g., above 0.99) in both holdout validation and K-Fold cross-validation methods.
* Such consistently high accuracy suggests that the models are very effective at predicting cluster labels, not only on the training data but also on new, unseen data.

1. **Overfitting Potential and Review:**

* Extremely high accuracy always warrants consideration of the possibility that the model has simply memorized the training data, i.e., overfitting.
* **Random Forest:** The potential for overfitting could be further scrutinized using OOB (Out-of-Bag) evaluation. The OOB score, an internal validation metric using data not directly involved in training each tree, was observed to be very high and nearly identical to the holdout test accuracy and the mean cross-validation accuracy (as indicated in the provided code's output). This strong agreement suggests that the Random Forest model generalizes well to new data rather than merely memorizing the training set, thereby mitigating concerns about overfitting. **This capability to leverage OOB evaluation for robust generalization assessment was a key factor in selecting the Random Forest algorithm for this project.**
* **AdaBoost:** Since AdaBoost does not provide an OOB score, its overfitting assessment relies more heavily on the consistency between holdout and K-Fold cross-validation results. If the accuracy variance across K-Fold splits is low, and both holdout test accuracy and mean cross-validation accuracy are consistently high (as was observed), it can be inferred that the AdaBoost model also possesses good generalization capabilities.

1. **Impact of KMeans Clustering:**

* An important consideration is that these classification models were trained to predict labels generated by KMeans clustering. If KMeans successfully created very distinct and well-separated clusters (e.g., if PCA visualization showed clear separation between clusters), subsequent classification models can easily learn these clear boundaries, leading to high accuracy.
* In such a scenario, high accuracy might not solely reflect the model's ability to learn complex patterns but rather indicate that it is solving a **'well-defined classification problem.'** This implies that the underlying data structure might have already been well-captured during the clustering phase.

**Overall Opinion:**

Random Forest and AdaBoost demonstrated excellent performance on the given cluster classification task. Random Forest, in particular, offers the advantage of OOB scoring as an additional means of validating against overfitting, which was a significant reason for its adoption in this project. KNN showed comparatively lower performance.

These results can be interpreted as the KMeans-generated clusters being relatively well-separated in the feature space, allowing powerful ensemble models to effectively learn their structure.

# -Modeling Approach(to find top 5 combination)

A top-level function (auto\_train\_and\_evaluate) was created to:

- Automatically handle preprocessing.

- Train and evaluate multiple regression models.

- Use GridSearchCV with 5-fold cross-validation.

- Rank models based on R² score.

- Return the top N performing models with optimal hyperparameters.

# Regression Models Used

- RandomForestRegressor: n\_estimators, max\_depth

- GradientBoostingRegressor: n\_estimators, learning\_rate

- SVR: C, kernel

- DecisionTreeRegressor: max\_depth

- LinearRegression: No tuning

# Evaluation Metric

- R² Score was used to evaluate model performance.

- The higher the R², the better the model explains variance in the data.

# Results (Top 5 Models)

1. GradientBoostingRegressor – R²: 0.6153 – Best Params: {'n\_estimators': 100, 'learning\_rate': 0.1}

2. RandomForestRegressor – R²: 0.6042 – Best Params: {'n\_estimators': 100, 'max\_depth': 20}

3. SVR – R²: 0.5527 – Best Params: {'C': 1, 'kernel': 'rbf'}

4. DecisionTreeRegressor – R²: 0.5045 – Best Params: {'max\_depth': 10}

5. LinearRegression – R²: 0.4651 – Best Params: {}