Principles & Elementary Models

REAL ESTATE IN GERMANY

Group 7:

Lou DE GAETANO NERINO DE VISCONTI,

Matilde MONTI,

Joaquin ARIAS,

Sicheng HUANG,

Himanshu MIDHA

INDEX

1 Introduction

Understanding the dynamics of the rental market is crucial for addressing housing accessibility, affordability, and fairness for all parties involved. In Germany, one of the largest countries in Europe, there are significant variations in rental prices and property characteristics across different regions, property types, and features.

A data-driven approach is essential to identify patterns and provide models that can become actionable insights.

This project focuses on a dataset containing rental housing offers across Germany[1]. The dataset includes various features, such as pricing, property characteristics, and geographical details.

Throughout the project, we encountered unique challenges, including high dimensionality, significant missing values, and the presence of outliers in the data. Many features were incomplete or inconsistent, needing systematic cleaning and the application of imputation techniques.

This process aims to apply machine learning techniques acquired from the Principles and Elementary Model course to understand the key drivers of rental pricing, identify significant market patterns, and predict future trends. The goal is to develop effective tools that provide tenants with transparency in the rental market and offer a comprehensive understanding of its behavior. Additionally, these insights will help property owners make informed decisions about pricing.

2 Data Cleaning

The raw dataset contains 49 variables and over 260,000 rows. During the cleaning process, we identified and assessed the proportions of missing data, detected

[1] Kaggle dataset: data scraped from Immoscout24, the largest real estate platform in Germany.

outliers in various columns, and ensured that the data types were appropriate for each variable.

2.1 Duplicated Rows

We first analyzed the dataset for duplicate rows and removed any existing duplicates. In this dataset, there are no duplicate rows; therefore, the dataset maintained its size and shape.

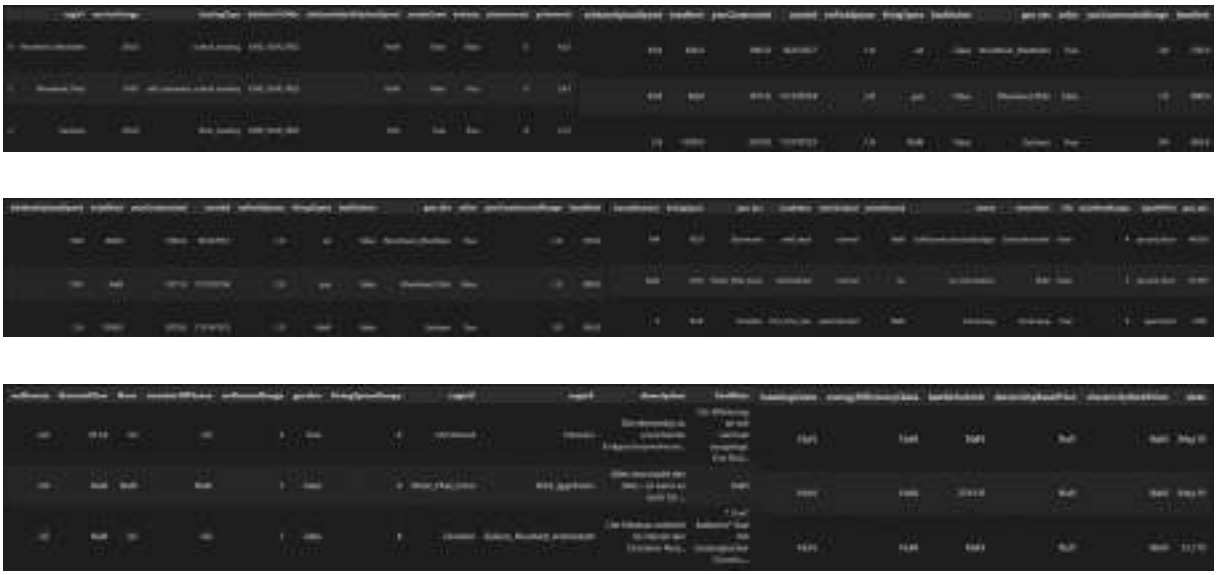
2.2 Null Values

In our initial attempt to assess the number of missing values in each column, we utilized the `isnull()` function along with `sum()` to count them. The results revealed that several columns had a substantial number of missing values. To better understand their impact, we developed a function that calculates the percentage of missing values for each column. This function produces a data frame that organizes the columns in descending order based on the percentage of missing values.

This is the output:

	Column	Missing Values	Percentage Missing	Data Type
0	telekomHybridUploadSpeed	223830	83.254603	float64
1	electricityBasePrice	222004	82.575414	float64
2	electricityKwhPrice	222004	82.575414	float64
3	energyEfficiencyClass	191063	71.066766	object
4	lastRefurbish	188139	69.979171	float64
5	heatingCosts	183332	68.191185	float64
6	noParkSpaces	175798	65.388879	float64
7	petsAllowed	114573	42.615957	object
8	interiorQual	112665	41.906267	object
9	thermalChar	106506	39.615399	float64
10	numberOfFloors	97732	36.351869	float64
11	houseNumber	71018	26.415473	object
12	streetPlain	71013	26.413614	object
13	condition	68489	25.474800	object
14	yearConstructed	57045	21.218151	float64
15	yearConstructedRange	57045	21.218151	float64
16	firingTypes	56964	21.188023	object
17	facilities	52924	19.685326	object
18	floor	51309	19.084620	float64
19	heatingType	44856	16.684397	object
20	totalRent	40517	15.070485	float64
21	typeOfFlat	36614	13.618747	object
22	telekomUploadSpeed	33358	12.407662	float64
23	telekomTvOffer	32619	12.132788	object
24	description	19747	7.344988	object
25	serviceCharge	6909	2.569834	float64
26	pricetrend	1832	0.681421	float64
27	scoutId	0	0.000000	int64
28	hasKitchen	0	0.000000	bool
29	balcony	0	0.000000	bool
30	newlyConst	0	0.000000	bool
31	regio1	0	0.000000	object
32	picturecount	0	0.000000	int64
33	geo_bln	0	0.000000	object
34	noRooms	0	0.000000	float64
35	baseRentRange	0	0.000000	int64
36	lift	0	0.000000	bool
37	street	0	0.000000	object
38	cellar	0	0.000000	bool
39	geo_krs	0	0.000000	object
40	livingSpace	0	0.000000	float64
41	baseRent	0	0.000000	float64
42	geo_plz	0	0.000000	int64
43	regio3	0	0.000000	object
44	garden	0	0.000000	bool
45	noRoomsRange	0	0.000000	int64
46	regio2	0	0.000000	object
47	livingSpaceRange	0	0.000000	int64
48	date	0	0.000000	object

Understanding which columns have a significant percentage of missing values is essential. Columns with a substantial amount of missing data will be removed, while those with fewer missing values will be addressed using various imputation methods. To ensure that we didn't overlook any important details by dropping columns, we first analyzed them to determine the best approach for handling the missing information.



2.3 Column Removal

The following columns have been removed due to high missing values, redundancy, or irrelevance to the analysis:

- **geo_bln**: contains the same values as regio1, making it redundant.
- **telekomHybridUploadSpeed**: over 80% of its values are missing.
- **pricetrend**: values assigned by the publishing website are ambiguous and lack clarity.
- **picturecount**: tracks the number of pictures in a listing, which has no impact on the analysis.
- **scoutId**: a unique identifier for each row, serving no analytical purpose.

- **houseNumber**: identifies properties but is irrelevant to the analysis and adds no meaningful value.
- **street**: dropped as it is redundant; **streetPlain** provides the same data with proper encoding.
- **description** and **facilities**: contain unstructured German text that is challenging to process and redundant with other features.
- **heatingCosts**: contains over 70% missing values, making it unreliable.
- **electricityBasePrice** and **electricityKwhPrice**: removed due to over 80% missing values.
- **date**: represents the scraping date of the data, which is irrelevant to the planned analysis[2].

2.4 Handling Missing Values and Imputation Techniques

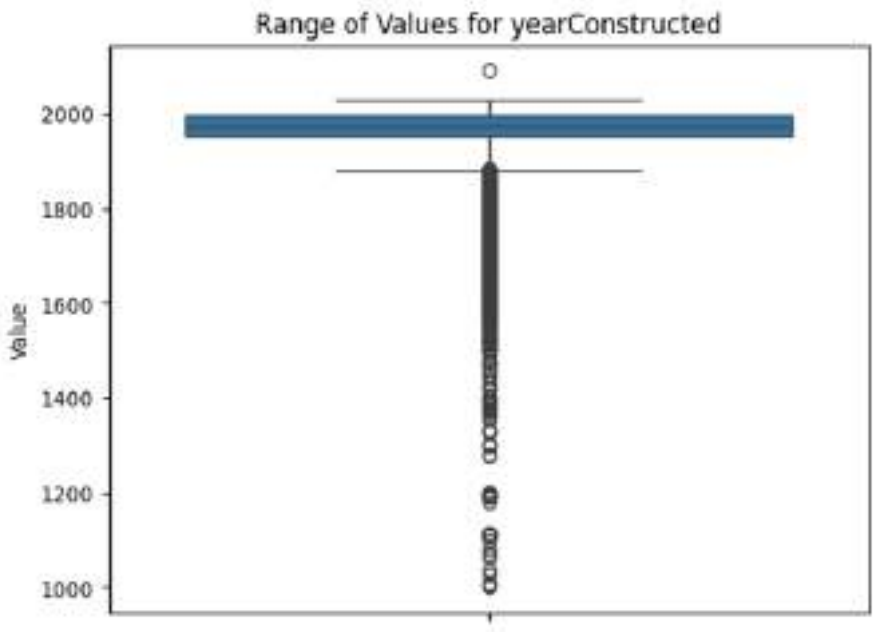
heatingType explains how the property heating system functions. The most common type is central heating, followed by district heating. This column contains only 16% of missing values, which could be significant for our modeling process. We have decided to address these missing values by imputing them as “unmentioned,” assuming that this indicates a lack of information on the website.

telekomTvOffer explains if the offer includes TV service or not. The null values were inferred as a lack of service provided.

garden is a boolean variable indicating whether the property has a garden or not. Assuming that if no information was provided about a garden, the property does not have one, missing values were filled with False.

yearConstructed details the year in which the property was built. A box plot was plotted to highlight outliers that can be classified as imputation errors:

[2] For PCA, we retrieved the column to calculate the age of the property.



All values over 2024 were interpreted as mistakes and therefore removed, while missing values were replaced with the median.

noParkSpaces (quantity of parking available for the tenant), when null was assumed to be equal to 0 (the offer doesn't include parking spaces). We also identified outliers that can be understood as imputation errors: to omit them, we used the clip function and got rid of all values higher than 10.

firingTypes specifies the main source of energy of the house. We have identified that many of the offers had more than one option (divided by the characters ":" and "and"). These cases were replaced with "multiple" to avoid having a high cardinality. For the null values, after doing some research, we found out that most of the houses in Germany are powered by gas[3]. The NaN values were therefore replaced with this firing type.

condition details the current state of the property. It has a lot of null values that were replaced with "well-kept" as it is the most common value and, unless specified the opposite, we can assume that the spaces offered as rental are in a liveable condition.

[3] Also the mode of the dataset.

For **interiorQual**, we replaced null values with the mode = “normal”. We also transformed the “simple” category into “normal” to avoid having a category with few values similar to another.

petsAllowed is a categorical variable describing the possibility of keeping pets in the house. The null values were filled with “negotiable”, as we can assume that in case of a decisive decision, it would have been clearly stated in the offer.

streetPlain was cleaned by replacing null values with “undisclosed”, as filling with the mode or other value would complicate the modeling and include potentially unreal information.

thermalChar defines the energy efficiency class of the property. We clipped the outliers in the top 1% to get rid of imputation errors, and we filled in the missing ones with the median.

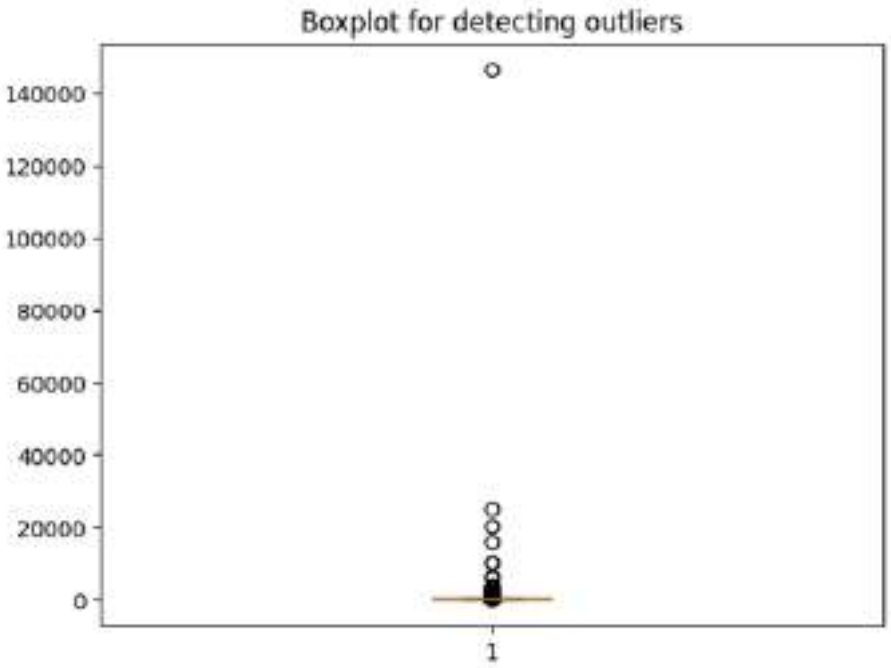
In the case of the variables **Floor** and **numberOfFloors**, variables indicating respectively the floor the property is and the total number of floors of the building, after doing some research we found out that the highest building in Germany has 63 floors. We clipped all values above this and replaced the null values with the median.

For **lastRefurbish**, indicating the last year the house was renovated, null values were replaced with “never_refurbished”, assuming that they were not modified after the year of construction.

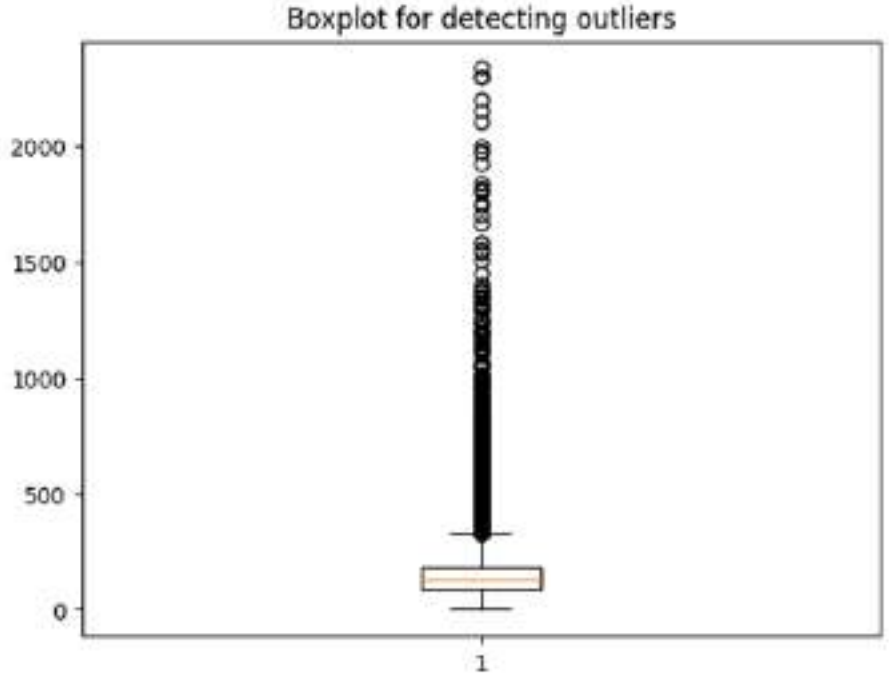
2.5 Special Treatment of Outliers and Imputation

serviceCharge entails the auxiliary costs such as electricity or internet, and assuming the service charges declared as null values mean that there is no service charge associated, or it is included in the total rent, we imputed 0 in all the null values.

To detect outliers and possible imputation errors, we plotted a boxplot with the following result:

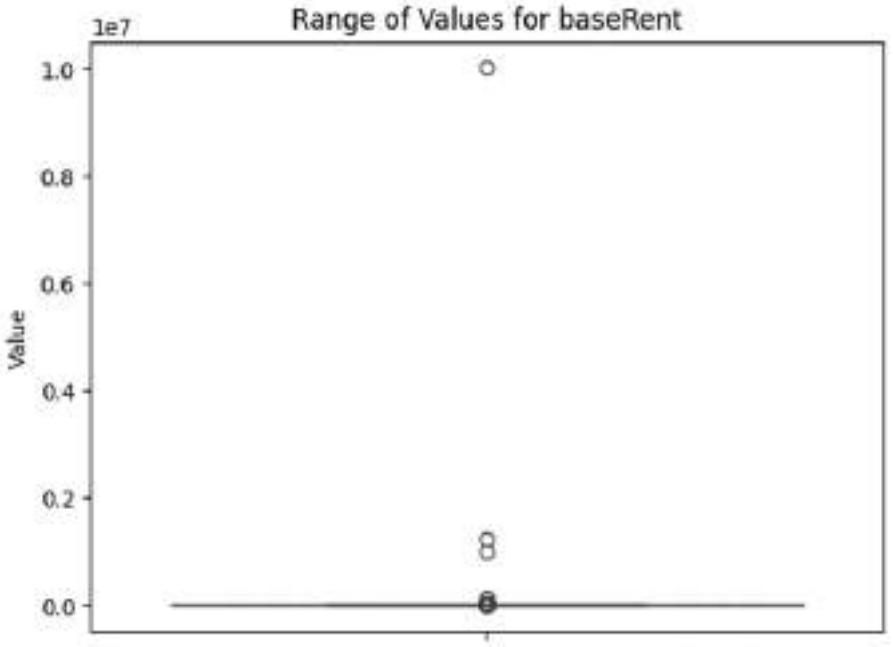


We assumed that service charge values exceeding 2,500 were unrealistic and classified them as errors. In total, we identified 13 of such values. To address these errors, we replaced them with the median values, grouping the data by **regio1** and **newlyConstructed**. This approach is based on the assumption that the service charge is influenced by these two factors. As a result, the boxplot now displays fewer outliers.

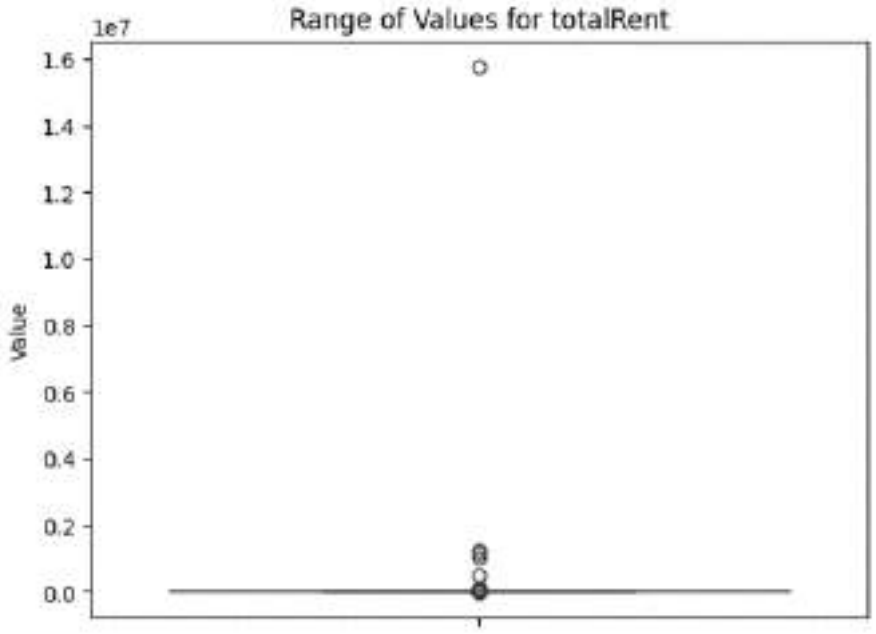


baseRent is the value of rent without electricity, heating, or other charges.

The outliers present in the data were removed at the 5% percentile.



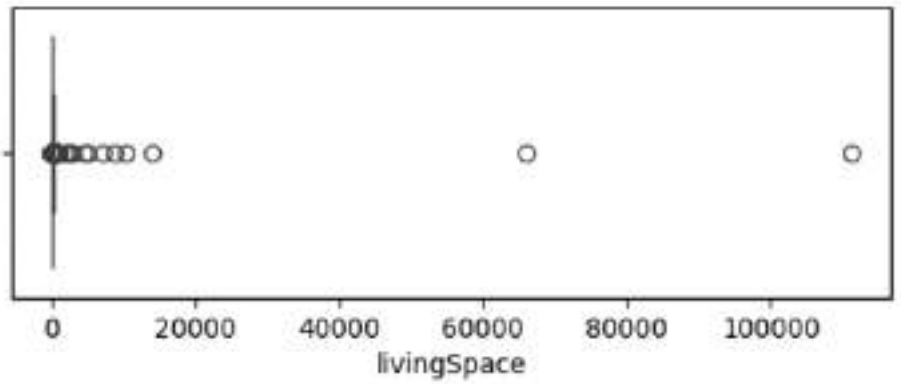
totalRent is the sum of **baseRent** and **serviceCharge**. The null values were plotted on this boxplot:



Due to the high number of outliers, we clipped them at the 0.10 quartile and for values above 10,000.

To fill the null values, we summed the **baseRent** and **serviceCharge** (if the data was available), otherwise, they were filled by the median.

livingSpace details the total meters squared of the property.



The boxplot clearly indicates the presence of outliers and values equal to 0. We considered the minimum and maximum living space values to be 5mq (room) and 450mq, respectively, and removed all rows with values outside these ranges.

	livingSpace	totalRent	baseRent	serviceCharge
35630	0.0	389.0	0.0	0.0
93073	0.0	389.0	0.0	0.0
231174	0.0	389.0	0.0	0.0
69577	0.0	389.0	0.0	1.5
232236	0.0	389.0	10.0	80.0
...	...	...	...	...
77422	2.0	1002.0	1000.0	2.0
19381	3.0	389.0	50.0	0.0
109242	3.0	389.0	225.0	155.0
5206	3.0	1970.8	1500.0	228.0
41232	5.0	650.0	25.0	0.0

101 rows × 4 columns

	livingSpace	totalRent	baseRent	serviceCharge
175397	111111.00	389.00	679.00	0.00
151184	66100.00	1400.00	1200.00	115.00
243713	14000.00	650.00	1500.00	133.00
223187	10259.00	1641.50	1385.00	128.25
51540	8684.00	650.00	504.00	146.00
202205	7008.00	650.00	519.00	107.00
229248	4947.00	650.00	200.00	140.00
172399	4340.00	389.00	200.00	60.00
215900	2782.00	650.00	245.00	60.00
222457	2420.00	650.00	283.14	99.35
92451	2257.88	650.00	800.00	111.00
50683	1717.74	650.00	0.00	0.00
227775	1000.00	541.50	539.00	2.50
12420	649.00	720.00	520.00	200.00
170816	601.85	10000.00	1500.00	1259.00
97475	600.00	389.00	340.00	0.00
14123	600.00	10000.00	1500.00	1400.00
190283	566.00	10000.00	1500.00	157.00

2.6 Unmodified Variables

- Variables like **balcony**, **cellar,** and **newlyConst** had complete data without null values and therefore needed no imputation or treatment.
- Variables like **geo_plz** and **typeOfFlat** were categorical or numeric and did not need filling in missing values.

The dataset has been systematically cleaned by addressing missing values, removing irrelevant columns, and managing outliers and imputation errors where necessary. We can now proceed with univariate and bivariate analysis as well as modeling.

3 Univariate Analysis

3.1 Quantitative Variables

- **baseRent** shows moderate rent levels mostly centered around 700 and 800. There is a high variability due to the luxury properties that have values above 10.000 which skew the distribution.
- **serviceCharge** has an average of around 150 and some extreme values suggest luxury or different-purpose rentals.
- Finally, **totalRent** has an average of and is a composition of base rent and service charges. The **livingSpace** median is approximately 60 to 80 square meters which shows that most properties are modest in size, likely representing urban apartments.
- In **noRooms** the median is three.

3.2 Categorical Variables

- The states that most commonly offer rental properties are North Rhine-Westphalia, Bavaria, and Baden-Württemberg.
- The predominant heating types are gas and central heating, while less common options include innovative solutions such as solar heating and traditional methods like wood chip heating.
- Most properties are well-maintained or refurbished. The most common type of rental unit is an apartment, although penthouses and ground floor units are also available, but they are less common.

3.3 Boolean Variables

Over 60% of properties have a balcony, making balconies a desirable feature.

- Only 30% have a garden, likely because these properties are in rural or suburban areas.
- As expected, the majority of properties are not newly constructed.

4 Bivariate Analysis

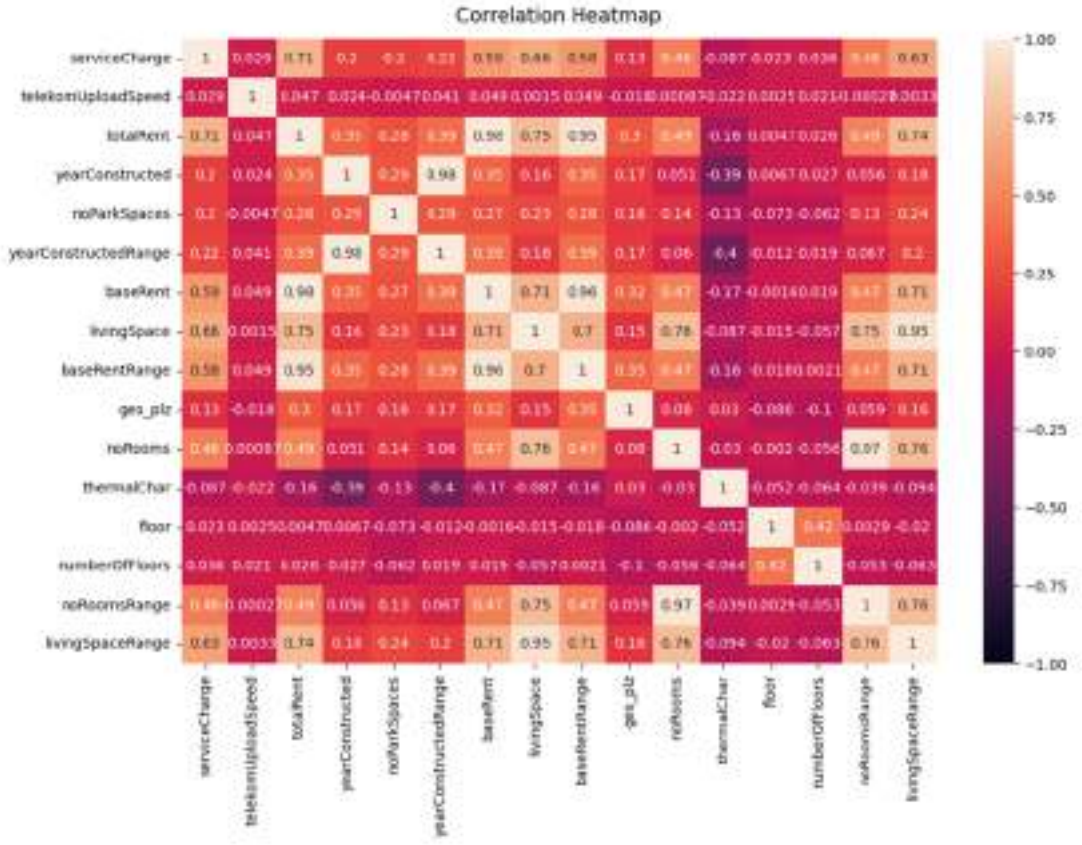
The bivariate analysis examines the relationships between different variables in the dataset, building on the most recent analysis presented in the notebook. This analysis includes three types of relationships: quantitative vs. quantitative, categorical vs. categorical, and quantitative vs. categorical. It is supported by statistical findings and visual references.

4.1 Quantitative vs. Quantitative Variables

The relationships between numerical variables highlight critical drivers of rental pricing. The correlation between **baseRent** and **totalRent** is exceptionally strong (r = 0.9848, p-value = 0.0), confirming that base rent constitutes the majority of total rent, with service charges and additional costs contributing marginally. Similarly, **totalRent** correlates strongly with **livingSpace** (r = 0.7465, p-value = 0.0), underscoring the role of property size in determining rental prices.

Moderate correlations were observed for other variables (see correlation map below):

- **serviceCharge** and **totalRent** (r = 0.7143, p-value = 0.0): Higher service charges are associated with premium properties that offer additional amenities or services.
- **noRooms** and **baseRent** (r = 0.4665, p-value = 0.0): Properties with more rooms tend to command higher rents, reflecting the value of additional space.



4.2 Categorical vs. Categorical Variables

Cross-tabulations and statistical tests such as Chi-Square and Cramer's V assess relationships between categorical features.

balcony and **kitchen**: the presence of balconies is significantly associated with the availability of kitchens. Properties with balconies are more likely to have kitchens, suggesting a clustering of desirable features. This association is statistically significant ($p < 0.05$).

balcony and **lift**: properties with balconies frequently include lifts, reflecting the tendency of premium properties to bundle high-end amenities. The strength of this association is confirmed using Cramer's V.

These relationships provide insights into how specific features co-occur, influencing tenant preferences and property value.

To conclude, the bivariate analysis provides several key insights:

- **Quantitative Variables**: property size (livingSpace, **noRooms**) and **baseRent** are the strongest predictors of **totalRent**, highlighting their importance in the rental pricing.
- **Categorical Variables**: features like balconies, lifts, and kitchens significantly increase rental values. Their presence is often correlated, especially in high-end properties.
- **Feature Clustering**: categorical features tend to cluster in premium properties, as shown by the relationships between balconies, lifts, and kitchens.

Visualizations such as heatmaps and scatterplots enhanced the understanding of these relationships. These findings highlight the importance of both size and feature-driven factors in optimizing rental strategies.

4.3 Quantitative vs. Quantitative Variables

Categorical variables such as balcony, lift, and kitchen significantly influence rental pricing. Boxplots in the notebook provide insights into these relationships.

baseRent and **balcony**: properties with balconies show higher median rents and wider variability compared to those without balconies. This indicates that balconies are a premium feature, often associated with luxury or high-end properties.

baseRent and **lift**: higher rents are observed for properties with lifts, reflecting the value tenants place on convenience and accessibility in multi-story buildings.

baseRent and **kitchenAvailability**: properties with kitchens exhibit narrower rent ranges and fewer extreme outliers, demonstrating that kitchens are considered essential features, stabilizing rental pricing.

The notebook's statistical tests confirm that these differences in rental prices are significant across categories.

5 Modeling

Before modeling, the dataset needs to be converted into a format suitable for machine learning. This process involves encoding and scaling the data.

Different encoding methods were used based on the cardinality of the categorical variables. For features with a cardinality greater than 30, we applied Label Encoding, which maps each unique value to an integer. In contrast, low cardinality variables were transformed using the “get_dummies()” method.

To enhance the model's efficiency in handling the data, we scaled the numerical features using StandardScaler. This ensures that the model is not biased toward features with larger numerical ranges.

The dataset is now ready for predicting the dependent variable.

5.1 Supervised Learning - Regression

5.1.1 Model Selection

With the completed dataset we selected our target variable: **totalRent**.

After splitting the data into train and test, we trained several regression models to predict the dependent variable:

- Linear Models: Linear Regression, Ridge Regression, Lasso Regression
- Nearest Neighbour Models: KNeighborsRegressor
- Tree-based Models: DecisionTreeRegressor, RandomForestReggressor
- Ensemble Learning Models: Gradient Boosting Regressor, XGBoost, LightGBM, CatBoost

By comparing their performance on the test set, we identified the best model for the final prediction. To quantify their accuracy in prediction, we used four metrics: MAE

(Mean Absolute Error), MSE (Mean Squared Error), RMSE (Root Mean Squared Error), and R^2 (Coefficient of Determination). For the first three metrics, smaller values indicate better performance, while for R^2, values closer to 1 are preferred.

The different models yielded the following results:

Model	**MAE**	**MSE**	**RMSE**	**R^2**
Linear Regression	0.36	0.40	0.63	0.61
Ridge Regression	0.36	0.40	0.63	0.61
Lasso Regression	0.65	1.02	1.01	0.01
KNeighbors Regressor	0.45	0.61	0.78	0.41
Decision Tree	0.32	0.45	0.67	0.56
Random Forest Regressor	0.24	0.24	0.49	0.77
Gradient Boosting Regressor	0.30	0.31	0.56	0.70
XGBoost	0.26	0.25	0.50	0.76
LightGBM	0.27	0.26	0.51	0.75
CatBoost	0.25	0.24	0.49	0.77

Based on these results, we found that RandomForestRegressor is the best-performing model. It stands out among all models, with the lowest MAE (0.24), lowest MSE (0.24), and RMSE (0.49), as well as one of the highest R^2 scores of 0.77.

Following Random Forest, CatBoost and XGBoost demonstrate comparable performance, with R^2 scores of 0.77 and 0.76 respectively. While their errors are slightly higher, they remain within acceptable range.

In contrast, Linear Regression and Ridge Regression showed moderate performance, with R^2 scores of 0.58, struggling to capture the nonlinear relationship in the data. Lasso Regression performed the worst, with almost an R^2 of 0, indicating strong underfitting. Other models, such as KNeighbourRegressor and Decision Tree showed acceptable performance but fell short compared to other ensemble learning models.

To choose the best model among these, we compared the performance of RandomForestRegressor and CatBoostRegressor on both train and test sets to check for signs of overfitting.

For the Random Forest, the training set performance metrics were much better than the ones obtained from the test set:

- Train set: MAE = 0.09, MSE = 0.03, RMSE = 0.19, R^2 = 0.97

Compared to the test set results (R^2 = 0.77), the training set performance indicated that the model explains 97% of the variance in the training data, while the lower R^2 score for the test set reflects a weaker ability to generalize the model to unseen data. This significant discrepancy suggests that the RandomForest Regressor may be overfitting the data.

CatBoost instead showed still signs of overfitting, but less strongly than the previous model:

- Train set: MAE = 0.24, MSE = 18, RMSE = 0.43, R^2 = 0.82

The smaller gap between the train and test R^2 scores show that the model is less prone to overfitting.

Due to the difference in the ability of the algorithms to generalize. We chose **CatBoostRegressor** as the best model.

5.1.2 Hyperparameter Tuning

After selecting the CatBoostRegressor as our model, we proceeded with hyperparameter tuning to optimize its performance. Using RandomizedSearchCv we first defined the following parameters to explore to find the best combination:

- 'depth' = [4, 6, 8, 9, 10, 12] (controls the maximum depth of the trees)
- 'learning_rate' = [0.01, 0.05, 0.1] [specified the step shrinkage used in updates to prevent overfitting)
- 'iterations' = [100, 200, 500] (defines the number of boosting operations)
- 'l2_leaf_reg' = [3, 5, 7] (controls the L2 regularization to prevent overfitting)

For each combination, the model was evaluated using a 3-fold cross-validation (cv = 3), and performance metrics were calculated.

The optimal parameters identified by the model were: 'learning_rate': 0.1, 'l2_leaf_reg': 5, 'iterations': 200, 'depth': 12.

With these optimal parameters, the CatBoost model was retrained on the training dataset, resulting in increased performance:

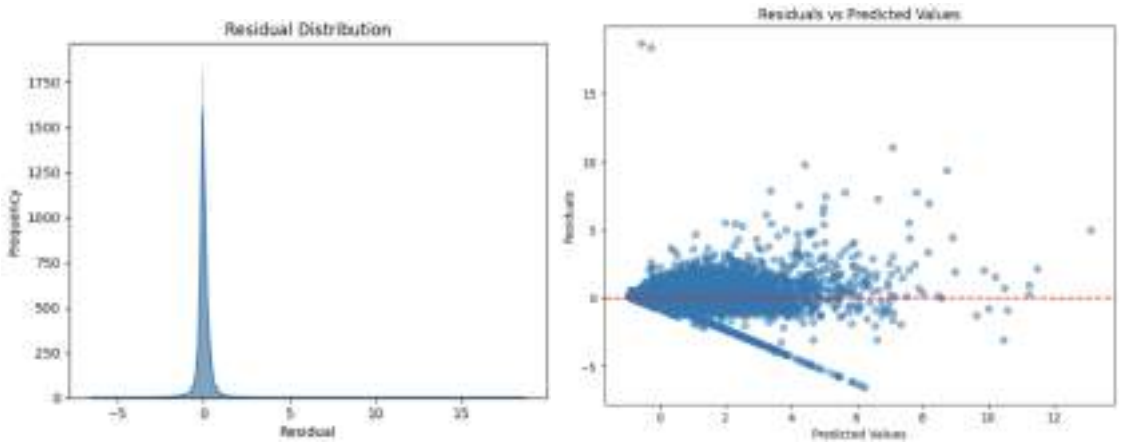
- MAE = 0.24, MSE = 0.22, RMSE = 0.47, R^2 = 0.79

5.1.3 Residuals Analysis

The formula for calculating the residuals is:

$$Residual = y_{actual} - y_{predicted}$$

To evaluate the residuals, we first plotted a histogram and a scatter plot of the residuals compared to the predicted values to further analyze the model's performance. These plots help analyze the distribution of residuals and whether there are any systematic errors in the data.



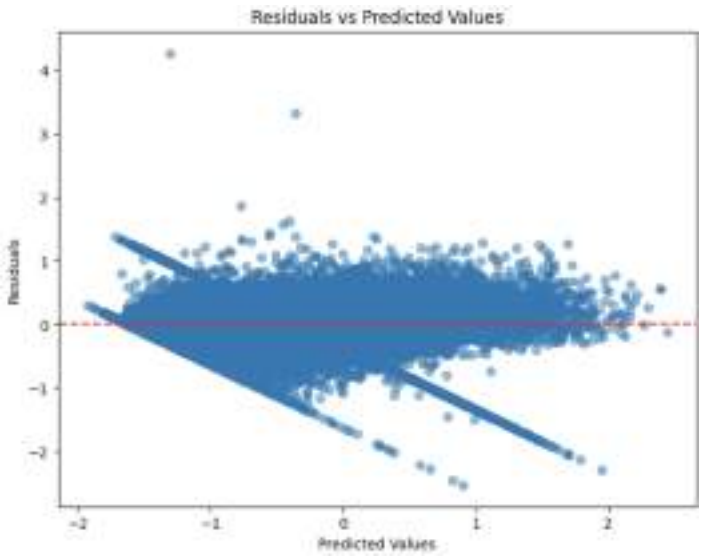
As visible from the graphs, the distribution is bell-shaped and symmetric, centered around 0, indicating that the model's predictions are unbiased.

The second graph instead shows a light funnel shape, indicating that the residuals are not completely randomly distributed. It suggests the presence of heteroskedasticity[4], with the model's prediction errors increasing with larger predicted values.

One way to address this problem is by looking at the target variable. From the univariate analysis, we can see that the dependent variable **totalRent** has a right-skewed distribution. To address this concern, we can take the log of the column and train the model on the normalized target.

After fitting the model, the performance of the model significantly improved, from 0.79 to 0.84 R^2, and with smaller errors (MAE = 0.23, MSE = 0.11, RMSE = 0.33).

[4] The variance of the residuals changes as the predicted values increase.



After the transformation, we can see that the residuals are more concentrated towards the horizontal zero line, which indicates less bias. One feature that stands out from the graph is the presence of two diagonal lines, suggesting the presence of systematic patterns in the data that the model cannot fully capture. This could be due to underlying structures in the dataset such as clusters or interactions between features that might explain this behavior.

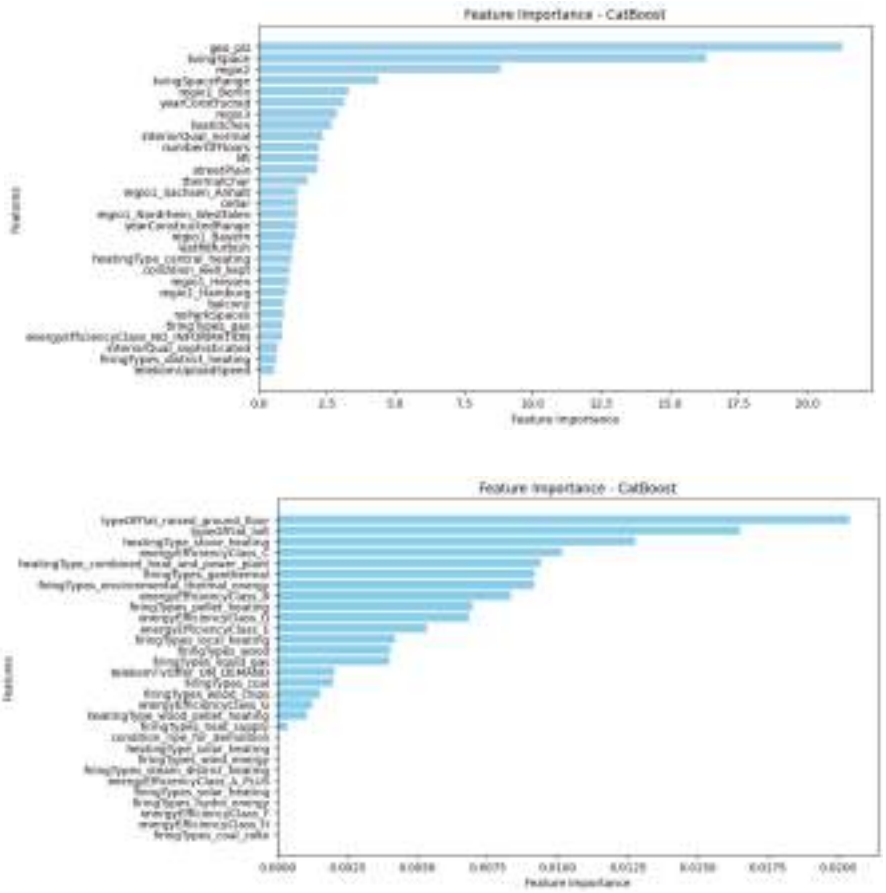
5.1.4 Feature Importance

The last step we computed in improving the model was feature importance.

The model may be having some trouble understanding some underlying patterns in the data due to the presence of irrelevant columns.

For this reason, we performed a feature importance analysis.

Based on the graph of the contribution of each feature in the prediction, we inferred the least influential columns from the tail of the distribution and removed them. However, training the model on this reduced subset of variables did not improve the prediction accuracy. This suggests that some of the removed features may be correlated with others, indirectly contributing useful information to the model. Feature importance may therefore not fully capture these indirect relationships.



One observation we can make is that the least important features in the ranking likely represent “outliers” in housing appliances. These include the extremes in energy efficiency classes, advanced heating technologies (such as solar heating and wind energy), or the marginal house states (condition_ripe_for_demolition). These outliers might correspond to the two distinct patterns observed in the residuals, which could reflect systematic differences in predictions for high-end properties and outdated properties.

5.2 Supervised Learning - Classification

The objective of this part is to predict some important features of properties such as the presence of a balcony or a garden, using the information provided by our dataset and the most relevant features. We are going to use logistic regression and random forest classifier models to predict the two variables mentioned before.

Our features are **livingSpace**, **baseRent**, **serviceCharge**, **noRooms**, **floor**, and **yearConstructed** and the first target variable was the presence or absence of a balcony in the rental offer. The dataset went through preprocessing such as scaling numerical features for the Logistic Regression and addressing class imbalance for the Random Forest.

5.2.1 Logistic Regression: Balcony Prediction

The results of this model are the following:

- **Accuracy:** 69.47%
- **Precision (True class):** 73%
- **Recall (True class):** 83%
- **F1-Score (True class):** 77%

Confusion Matrix:

	Predicted: No Balcony	Predicted: Balcony
Actual: No Balcony	8.898	9.976
Actual: Balcony	5.528	26.303

Logistic Regression performs moderately well and achieves balanced precision and recall when predicting whether a property has a balcony. However, there is a high false positive rate.

5.2.2 Random Forest: Balcony Prediction

The results of the Random Forest model are the following:

- **Accuracy:** 72.61%
- **Precision (True class):** 75%
- **Recall (True class):** 84%

- **F1-Score (True class):** 79%

Confusion Matrix:

	Predicted: No Balcony	Predicted: Balcony
Actual: No Balcony	10.062	8.902
Actual: Balcony	5.006	26.825

Random Forest outperforms Logistic Regression with better accuracy and F1 scores. It reduces false positives while increasing true positives.

Random Forest is the better model, with improved accuracy (72.61%) and balanced precision-recall trade-offs. It is suitable for applications where the prediction of balcony availability is essential, such as in tenant-facing platforms or real estate recommendation systems.

5.2.3 Logistic Regression: Garden Prediction

When predicting whether an offer has a garden or not the Logistic Regression is performed obtaining the following results:

- **Accuracy:** 80.30%
- **Precision (True class):** 53%
- **Recall (True class):** 0.01%
- **F1-Score (True class):** 0.01%

Confusion Matrix:

	Predicted: No Garden	Predicted: Garden
Actual: No Garden	40.755	31
Actual: Garden	9.974	35

Despite high overall accuracy, the Logistic Regression model struggled with predicting garden presence, as evidenced by extremely low recall (0.01%). This indicates that it is heavily biased toward predicting the absence of gardens. Such poor recall makes the model unsuitable for this application.

5.2.3 Random Forest: Garden Prediction

- **Accuracy:** 63.57%
- **Precision (True class):** 30%
- **Recall (True class):** 62%
- **F1-Score (True class):** 40%

Confusion Matrix:

	Predicted: No Garden	Predicted: Garden
Actual: No Garden	26.123	14.663
Actual: Garden	3.841	6.168

Both models show limitations, with Logistic Regression failing and Random Forest performing moderately. For predicting garden availability, Random Forest provides a starting point but requires further refinement.

5.3 Unsupervised Learning

5.3.1 Principal Component Analysis

Principal Component Analysis is a dimensionality reduction technique that can transform a high-dimensional dataset into a smaller number of components while retaining the most significant information. PCA was applied to the German Rental Offers to uncover the key factors that impact rent properties and their prices.

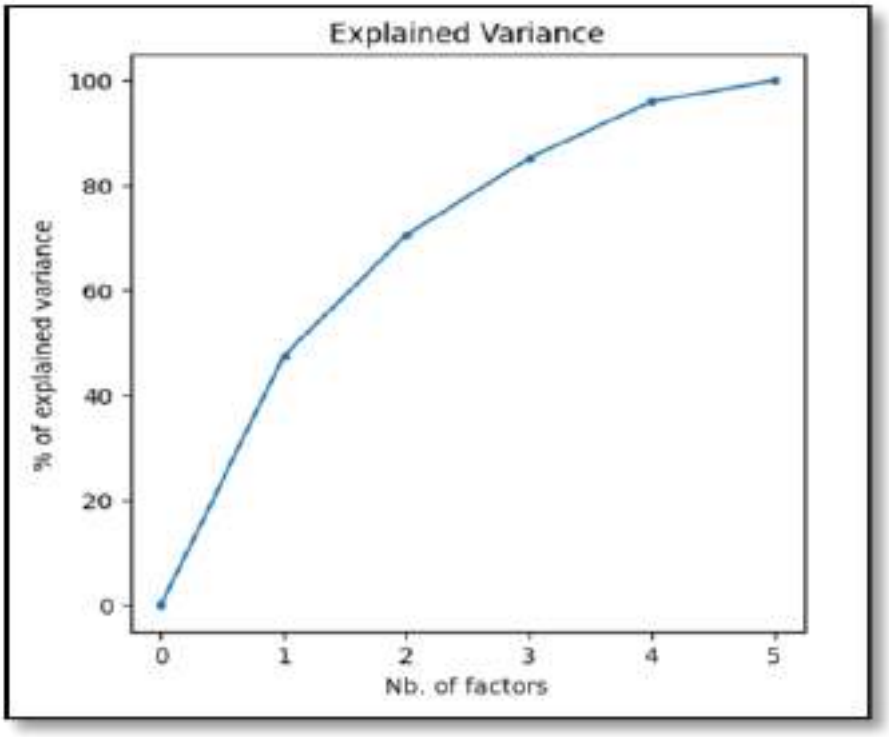
Based on the correlation matrix, the following features were deemed as the most relevant for the analysis:

- **noParkSpaces**
- **livingSpaceRange**
- **noRooms**
- **property_age**
- **totalRent**

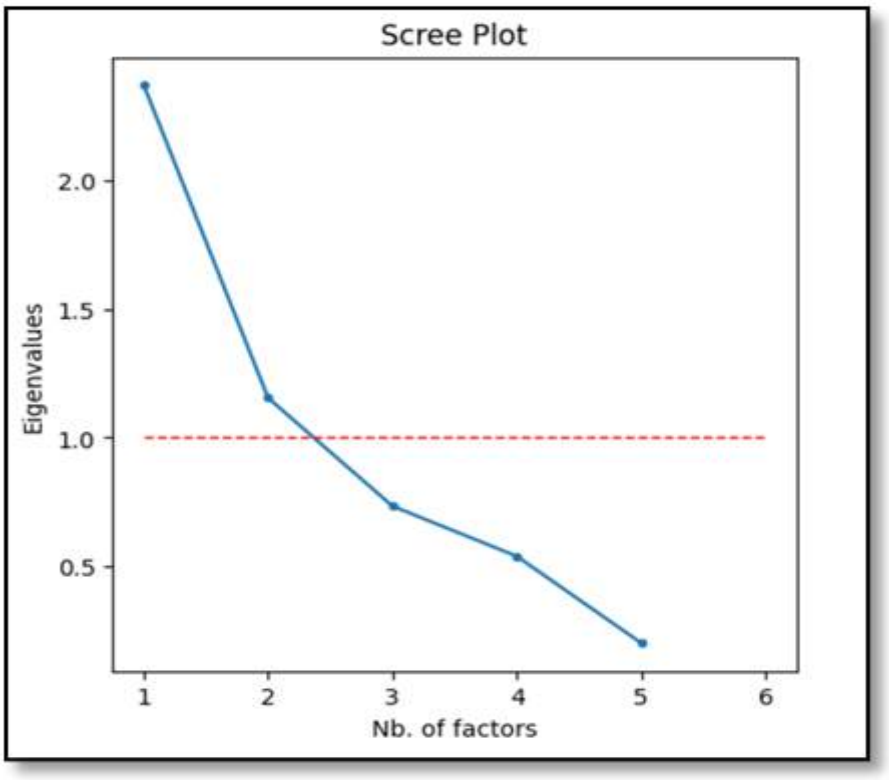
The variable property_age was created using feature engineering as the difference between the construction year and the extraction year. These variables were also standardized using MinMaxScaler, ensuring all features were between 0 and 1.

Results:

The first two components capture a significant proportion of the German Rental Offers variance. F1 explains 47% of the total variance and F2 23%, accounting for 70% of the total variance.



Eigenvalues are greater than 1 (using Kaiser´s Criterion) for the first two components, confirming their significance.

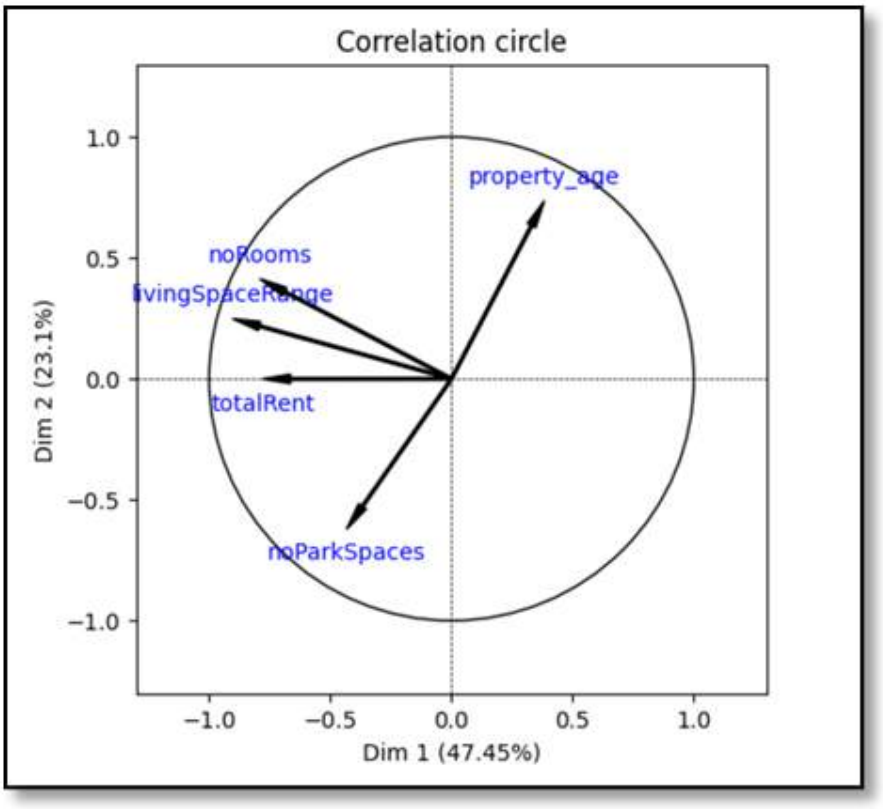


Factor 1 is dominated by features related to the apartment size and the rental cost:

- **livingSpaceRange** contributes 34%
- **noRooms** contributes 26%
- **totalRent** contributes 25%

Factor 2 represents external infrastructure and construction-related characteristics:

- **property_age** contributes 47% with a negative correlation with the dominant direction.
- **noParkSpaces** contributes 33%.



Bartlett's test indicates a significant p-value, less than 0.05, confirming that PCA is appropriate for this dataset.

Karlis-Saporta-Spinaki Threshold: Aligns with retaining two components, consistent with the result obtained in the scree plot and broken stick model.

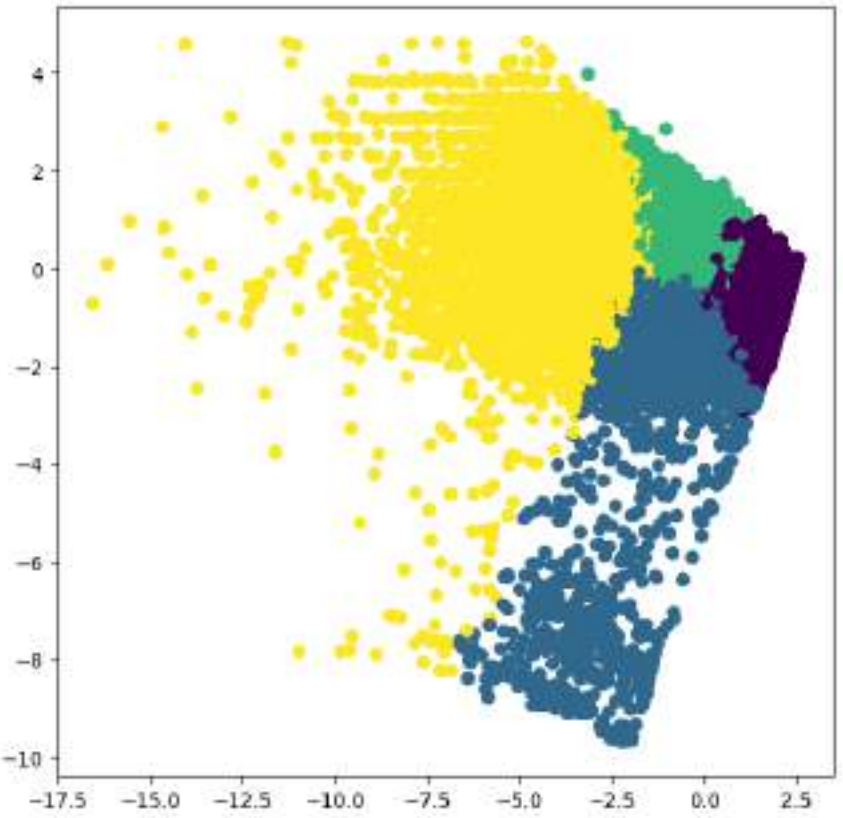
The use of PCA has made our dataset easier to analyze and interpret. The first key dimension that influences the real estate market in Germany can help emphasize the relationship between apartment size and rental pricing, helping identify optimal spaces within the budget of the tenants. The second one shows the importance of parking availability, property quality and age to the people interested in renting an apartment.

5.3.2 Clustering

After applying Principal Component Analysis (PCA) to reduce dimensionality and uncover the most significant factors influencing rental offers in Germany, K-Means

clustering was employed to group properties based on the reduced feature space. By leveraging PCA, which condensed the dataset's complexity into two principal components that explain 70% of the total variance, clustering became a practical method to uncover patterns and insights about property characteristics.

K-Means clustering identified four distinct property groups. These clusters can be interpreted based on the underlying factors represented by the two principal components:



Yellow Cluster:

- Represents a diverse range of properties with mixed characteristics, such as varying rental prices, living spaces, and parking availability.
- This group likely captures a heterogeneous set of properties, serving as a general cluster.

Blue Cluster:

- Likely corresponds to budget-friendly properties with smaller living spaces or older construction.
- This group is significant for tenants with limited budgets, seeking affordable housing options.

Green Cluster:

- Represents properties with moderate features, such as average rent, medium living space, and some parking availability.
- This cluster could appeal to families or mid-income tenants looking for a balance between cost and comfort.

Purple Cluster:

- Likely highlights premium properties with newer construction, higher rental prices, and ample parking availability.
- This group targets tenants seeking modern apartments with enhanced features and amenities.

The clustering analysis achieved a silhouette score of **0.258**, suggesting that the clusters are not optimally separated. While the clusters provide meaningful interpretations, the low silhouette score indicates potential overlap between groups or the influence of additional variables not captured by the first two principal components.

7 Conclusions

This project provides a comprehensive analysis of the German rental market using a combination of data cleaning, and supervised and unsupervised machine learning techniques. The findings show the importance of a data-driven approach in uncovering key drivers of rental prices and property characteristics, while also offering actionable insights for tenants and property owners.

From a **technical perspective**, the dataset was systematically cleaned to address missing values, outliers, and redundancies. Principal Component Analysis (PCA) effectively reduced dimensionality, enabling the identification of two significant factors accounting for 70% of the variance: rental pricing and apartment size (Factor 1) and external features such as parking availability and property age (Factor 2). K-Means clustering further segmented the properties into meaningful groups, despite suboptimal separation as indicated by the silhouette score. Supervised learning models demonstrated their utility in predicting both quantitative (e.g., total rent) and categorical outcomes (e.g., balcony and garden availability). The CatBoost Regressor emerged as the best-performing model for rental price prediction due to its ability to generalize well while minimizing overfitting.

From a **practical perspective**, these analyses reveal significant trends and patterns in the rental market. For tenants, the results highlight the trade-offs between price, size, and additional features, helping them identify properties that align with their preferences and budget. Property owners can use these insights to competitively price their listings while emphasizing features that tenants value most, such as balconies, lifts, and kitchens. Overall, the integration of PCA, clustering, and predictive modeling not only simplifies the complexity of the dataset but also

translates data insights into actionable strategies for diverse stakeholders in the rental market.