Machine Learning-Based Rental Price Prediction
A Comprehensive Development and Implementation for EarlyBuild GmbH

by
Thusitha Wijesooriya

Table of Contents

(1) Abstract

This capstone project is developing a machine learning rental price forecasting model tailored for EarlyBuild GmbH, a revolutionary German real estate tech company. Precise computation of rental values in today's volatile housing market is an overriding challenge considering the myriad of influencing factors varying from property features to locational advantage. The mission of this endeavour is to establish a fitting and interpretable model that can predict overall rent with very high precision.

By using information collected from Germany's leading real estate website, ImmoScout24, and amenity information queried through OpenStreetMap's Overpass API, the team undertook extensive pre-processing and feature engineering to transform raw property listings into a processed analytical dataset. The enriched data captures both inherent property characteristics (i.e., size, condition, interior quality) as well as spatial attributes (i.e., proximity to public facilities, commercial districts, and schools).

Through diligent exploratory data analysis and use of various forecasting methods, the developed solution not only offers accurate rental forecasts but also unlocks actionable insight into the key determinants of rent. Equipped with such a model, EarlyBuild GmbH is best positioned to advance effective pricing policies, optimise openness in listings, and offer an analytics-driven experience for both the proprietors and future tenants.

(2) Introduction

Predictive precision of rental prices is more urgently required in the modern, fast-paced real estate industry. Homeowners would like to competitively list their listings, while residents desire fair returns on investment for their money. Meanwhile, a more affluent set of data assets paired with big analytics offers companies the ability to optimise pricing algorithms and offer more utility to consumers.

This project was collaborated on with EarlyBuild GmbH with the vision of developing a comprehensive rental price prediction model based on real estate and location data. The team acquired property listing data from Germany's largest online home platform, ImmoScout24, and enriched it with amenity data from OpenStreetMap. This merge allows the model to include both property-level features and contextual neighbourhood-level information.

By employing machine learning techniques, not only was it possible to more accurately predict rental rates, but patterns and trends could be discerned that would inform EarlyBuild's pricing strategy and product development. The result is an information-driven system that is designed to improve decision-making, increase market transparency, and maximise EarlyBuild GmbH's technological prowess in real estate analytics.

(3) Business Understanding

(3.1) Business Context

EarlyBuild GmbH is a real estate technology company operating in the German housing market. With constant shifts in housing demand, changing urban landscapes, and rising customer expectations, accurately pricing rental properties has become more important than ever. For property owners, fair and informed pricing can help reduce vacancy rates, while renters benefit from increased transparency and trust in listings.

To support this, EarlyBuild GmbH is aiming to integrate data-driven tools into its operations, particularly through the use of machine learning models that can predict rental prices more accurately. The project uses real-world housing data collected from Germany's largest property platform, along with information about nearby amenities sourced from OpenStreetMap. By combining these sources, the company hopes to better understand what factors truly drive rental prices in different locations.

(3.2) Business Goals and Problems Addressed

The primary business objective of this project is to develop a machine learning-based rental price prediction model that can provide accurate estimates of total rent for residential properties listed on real estate platforms.

The following specific goals and challenges were addressed.

- Develop an accurate rental price prediction model that considers both property-specific features (e.g., size, condition, interior quality, heating type) and external factors (e.g., proximity to amenities such as schools, clinics, restaurants, and public-transport).

- Improve pricing transparency for property owners and renters by using interpretable models and feature-based explanations.

- Clean and handle incomplete data by building a strong preprocessing workflow that ensures the model receives reliable and consistent input, improving overall prediction accuracy.

- Enhance each property listing with details about its surrounding neighborhood by integrating location-based amenity data. This was done using geospatial bounding boxes and data retrieved from the Overpass API.

- Use exploratory data analysis and correlation techniques to uncover the most influential factors affecting rental prices, helping EarlyBuild GmbH make better pricing decisions and tailor its marketing approach more effectively.

This project supports EarlyBuild GmbH's strategic vision to integrate advanced analytics into its business processes, reduce human pricing errors, and enhance customer experience through personalized recommendations and fair rental assessments.

(4) Data Understanding

(4.1) Sources of Data

- Apartment rental offers in Germany (CSV) - Rental offers scraped from Germany's biggest real estate online platform, Immoscout24.
- Amenities in Germany from OpenStreetMap's (fetched through Overpass API)

List of Amenities considered.

Amenity	**Description**
restaurant	Restaurant (not fast food, see **amenity**=fast_food). The kind of food served can be tagged with cuisine=* and diet:*=*.
fast_food	Fast food restaurant (see also **amenity**=restaurant). The kind of food served can be tagged with cuisine=* and diet:*=*.
bar	**Bar** is a purpose-built commercial establishment that sells alcoholic drinks to be consumed on the premises. They are characterised by a noisy and vibrant atmosphere, similar to a party and usually don't sell food. See also the description of the tags **amenity**=pub;bar;restaurant for a distinction between these.
college	Campus or buildings of an institute of Further Education (aka continuing education)

kindergarten	For children too young for a regular school (also known as preschool, playschool or nursery school), in some countries including afternoon supervision of primary school children.
school	School and grounds - primary, middle and secondary schools
university	A university campus: an institute of higher education
bus_station	May also be tagged as public_transport=station.
parking	Parking area for vehicles. Streets on car parking are often tagged highway=service and service=parking_aisle.
bank	Bank or credit union: a financial establishment where customers can deposit and withdraw money, take loans, make investments and transfer funds.
clinic	A medium-sized medical facility or health centre.
dentist	A dentist practice/surgery.
doctors	A doctor's practice/surgery.
hospital	A hospital providing in-patient medical treatment. Often used in conjunction with emergency=* to note whether the medical centre has emergency facilities (A&E (brit.) or ER (am.))
cinema	A place where films are shown (US: movie theater)
community_centre	A place mostly used for local events, festivities and group activities; including special interest and special age groups.
nightclub	A place to drink and dance (nightclub).

	The German word is "Disco" or "Discothek". Please don't confuse this with the German "Nachtclub" which is most likely amenity=stripclub.
theatre	A theatre or opera house where live performances occur, such as plays, musicals and formal concerts. Use **amenity**=cinema for movie theaters.
police	A police station where police officers patrol from and that is a first point of contact for civilians
post_office	Post office building with postal services
drinking_water	**Drinking water** is a place where humans can obtain potable water for consumption. Typically, the water is used for only drinking. Also known as a **drinking fountain** or **bubbler**
toilets	Public toilets (might require a fee)
water_point	Place where you can get large amounts of drinking water
waste_disposal	A medium or large disposal bin, typically for bagged up household or industrial waste.
Source : https://wiki.openstreetmap.org/wiki/Key:amenity	

(4.2) Description of Fields and Variables of the combined dataset (immo_ammenities_1000.csv

Variables	Description
Regio1	Geographical region or administrative district where the property is located (in Germany)
ServiceCharge	Monthly or yearly service charges associated with the property, excluding rent.
HeatingType	Type of heating system in the property
TelekomTvOffer	Availability of Telekom TV service at the property
TelekomHybridUploadSpeed	The upload speed of Telekom Hybrid internet service provided at the property (in Mbps)
NewlyConst	Indicates whether the property is newly constructed
Balcony	Indicates whether the property has a balcony
Picturecount	The number of images available for the property listing.
Pricetrend	Historical price trends for properties in the same location
TelekomUploadSpeed	The upload speed of Telekom internet service provided at the property (in Mbps).
TotalRent	The total rental price of the property, including all additional charges
YearConstructed	Year the property was constructed.
ScoutId	A unique identifier associated with the property listing on the platform.
NoParkSpaces	The number of parking spaces available with the property.
FiringTypes	Types of fire safety equipment or installations available in the property
HasKitchen	Indicates whether the property has a fully equipped kitchen
Geo_bln	Geographical boundary or code for the specific district in Berlin
Cellar	Indicates whether the property has a cellar or basement
YearConstructedRange	The range of years when the property was constructed
BaseRent	Base rental price of the property, excluding additional costs like service charges or utilities.
HouseNumber	The street number of the property.
LivingSpace	Total living area of the property, typically measured in square metres.
Geo_krs	Geographical code for the specific district or region
Condition	Condition of the property
InteriorQual	Quality of the interior features
PetsAllowed	Indicates whether pets are allowed in the property
Street	The street or road name where the property is located
StreetPlain	Simplified or standardized version of the street name
Lift	Indicates whether the property has an elevator
BaseRentRange	The range of base rental prices in the neighbourhood or district.

TypeOfFlat	Type of flat or apartment
Geo_plz	Postal code for the property location.
NoRooms	The number of rooms in the property
ThermalChar	Thermal characteristics or insulation rating of the property
Floor	The floor number on which the property is located
NumberOfFloors	Total number of floors in the building.
NoRoomsRange	Range of the number of rooms available in the area
Garden	Indicates whether the property has access to a garden
LivingSpaceRange	Range of living space areas in the area
Regio2	Second geographical region or district of the property
Regio3	Third geographical region or sub-region of the property.
Description	Textual description or features of the property.
Facilities	Facilities available with the property
HeatingCosts	Monthly or annual heating costs for the property.
EnergyEfficiencyClass	Energy efficiency rating of the property
LastRefurbish	Date of the last renovation or refurbishment of the property.
ElectricityBasePrice	Base price for electricity usage in the property
ElectricityKwhPrice	Price per kilowatt hour (kWh) for electricity usage in the property.
Date	Date of the property listing or data entry.
Latitude	Latitude coordinate of the property's location (geographical coordinate).
Longitude	Longitude coordinate of the property's location (geographical coordinate).
Restaurant	Number of restaurants within a 2 km radius.
Fast_Food	Number of fast food outlets within a 2 km radius.
Bar	Number of bars within a 2 km radius.
College	Number of colleges within a 2 km radius.
Kindergarten	Number of kindergartens or preschools within a 2 km radius.
School	Number of schools within a 2 km radius.
University	Number of universities within a 2 km radius.
Bus_Station	Number of bus stations within a 2 km radius.
Parking	Number of parking lots or facilities within a 2 km radius.
Bank	Number of banks within a 2 km radius.
Clinic	Number of clinics or healthcare facilities within a 2 km radius.
Dentist	Number of dental clinics within a 2 km radius.
Doctors	Number of general practitioner or doctor offices within a 2 km radius.
Hospital	Number of hospitals within a 2 km radius.
Cinema	Number of cinemas within a 2 km radius.
Community_Center	Number of community centers within a 2 km radius.
Nightclub	Number of nightclubs within a 2 km radius.
Theatre	Number of theatres within a 2 km radius.

Police	Number of police stations within a 2 km radius.
Post_Office	Number of post offices within a 2 km radius.
Drinking_Water	Number of accessible drinking water sources within a 2 km radius.
Toilets	Number of toilets in the property.
Water_Point	Number of general water points (e.g., public taps) within a 2 km radius.
Waste_Disposal	Number of waste disposal or recycling facilities within a 2 km radius.
Latitude_min	Minimum latitude value defining the southern edge of the 2 km bounding box around the property.
Latitude_max	Maximum latitude value defining the northern edge of the 2 km bounding box around the property.
Longitude_min	Minimum longitude value defining the western edge of the 2 km bounding box around the property.
Longitude_max	Maximum longitude value defining the eastern edge of the 2 km bounding box around the property.

(4.3) Data Types

Data Type	Count	Description
String	33	Textual/categorical fields such as region names, types, street names, etc.
Boolean	6	Binary fields representing True/False conditions like balcony, hasKitchen, lift, etc.
Numeric	37	Continuous or discrete numerical values such as rent, year constructed, area size, counts, coordinates, etc.

(4.4) Combining the Data sources.

The dataset was cleaned and corrected by fixing swapped latitude and longitude values and removing any rows with missing or invalid coordinates. The first 1,000 entries were selected to keep the dataset manageable for further processing. For each property, a 2-kilometer radius bounding box was calculated, and the Overpass API was used to gather information about nearby amenities such as restaurants, schools, clinics, and public transportation. The number of these amenities within the bounding box was added to the dataset as new features, providing valuable context about each property's surroundings. This spatial enrichment aimed to capture how location-based factors might influence rental prices. For further cleaning and pre-processing, the dataset was saved as **immo_amenities_1000.csv** for use in the next stages of the project.

```python
import pandas as pd
import requests
import time
import numpy as np

# Function to calculate bounding box (adjusted to 2 km radius)
def get_bounding_box(lat, lon, km=2):
    delta_lat = km / 111.32  # Convert km to degrees latitude
    delta_lon = km / (111.32 * np.cos(np.radians(lat)))  # Convert km to degrees longitude
    return lat - delta_lat, lat + delta_lat, lon - delta_lon, lon + delta_lon

# Function to query Overpass API
def query_overpass(lat_min, lat_max, lon_min, lon_max, amenity, max_retries=3):
    overpass_url = "https://overpass-api.de/api/interpreter"
    query = f"""
    [out:json];
    node[amenity={amenity}]({lat_min},{lon_min},{lat_max},{lon_max});
    out;
    """

    for attempt in range(max_retries):
        try:
            response = requests.get(overpass_url, params={"data": query}, timeout=10)
            if response.status_code == 200:
                data = response.json()
                count = len(data.get("elements", []))
                print(f"✔ Found {count} {amenity}(s) in ({lat_min}, {lon_min}) - ({lat_max}, {lon_max})")
                return count  # Store the number of amenities found
        except requests.exceptions.RequestException as e:
            print(f"API Request Error: {e}")
            time.sleep(2 ** attempt)  # Exponential backoff
```

The full code used for this process is available here - [DatasetPrep.py on GitHub](#).

(5) Data Preparation (Data Cleaning and Pre-processing)

(5.1) Strategic Feature Selection

Isolated 58 predictive features while eliminating the unwanted features of the dataset.

```
import pandas as pd
import numpy as np
import os
from sklearn.preprocessing import LabelEncoder, StandardScaler

# Load dataset
file_path = "../data/processed/immo_amenities_1000.csv"
df = pd.read_csv(file_path)

# --- STEP 1: SELECT REQUIRED COLUMNS ---
selected_columns = [
    "serviceCharge", "heatingType", "telekomTvOffer", "telekomHybridUploadSpeed",
    "newlyConst", "balcony", "picturecount", "pricetrend", "telekomUploadSpeed",
    "totalRent", "yearConstructed", "scoutId", "noParkSpaces",
    "hasKitchen", "cellar", "yearConstructedRange", "baseRent",
    "livingSpace", "condition", "interiorQual", "petsAllowed",
    "lift", "baseRentRange", "typeOfFlat", "noRooms",
    "thermalChar", "floor", "numberOfFloors", "noRoomsRange", "garden", "livingSpaceRange",
     "heatingCosts",
    "lastRefurbish", "electricityBasePrice", "electricityKwhPrice", "restaurant", "fast_food", "bar", "college", "kindergarten",
    "school", "university", "bus_station", "parking", "bank", "clinic", "dentist",
    "doctors", "hospital", "cinema", "community_centre", "nightclub", "theatre",
    "police", "post_office", "drinking_water", "toilets", "water_point", "waste_disposal"
]
```

(5.2) Data Type Conversion (String to Numeric)

Converted the data type of a few variables that should be in numeric format but are in string format in the dataset.

```
# --- STEP 2: CONVERT STRING COLUMNS TO NUMERIC ---
numeric_cols = [
   "telekomTvOffer", "serviceCharge", "telekomHybridUploadSpeed", "pricetrend", "telekomUploadSpeed",
    "totalRent", "yearConstructed", "noParkSpaces", "thermalChar", "floor",
    "numberOfFloors", "heatingCosts", "electricityBasePrice", "electricityKwhPrice"
]

# Convert to numeric & handle errors
for col in numeric_cols:
    df[col] = pd.to_numeric(df[col], errors="coerce")
```

(5.3) Handling Missing Values

Missing data was addressed through imputation.

- **Numerical features** (e.g. heatingCosts) used median imputation (€85) for robustness against outliers
- **Categorical variables** (e.g., heatingType) were assigned an "unknown" category to preserve their qualitative nature.
- **Binary fields** (e.g., hasKitchen) defaulted to modal values (1 for true/yes cases) to maintain logical consistency.

```
# --- STEP 3: HANDLE MISSING VALUES ---
# Fill missing numerical values with median
num_cols = df.select_dtypes(include=[np.number]).columns
df[num_cols] = df[num_cols].apply(lambda x: x.fillna(x.median()))

# Fill missing categorical values with 'unknown'
cat_cols = df.select_dtypes(exclude=[np.number]).columns
df[cat_cols] = df[cat_cols].fillna("unknown")
```

(5.4) Dealing with duplicate records.

Duplicate records have been removed from the dataset.

```
# --- STEP 4: REMOVE DUPLICATES ---
df.drop_duplicates(inplace=True)
```

(5.5) Encoding categorical features.

- Categorical features have been encoded to numerical features with the labelEncoder technique.
- The categorical features encoded are heatingType, condition, interiorQual and petsAllowed.

```
# --- STEP 6: ENCODE CATEGORICAL VARIABLES ---
label_enc = LabelEncoder()
for col in ["heatingType", "condition", "interiorQual", "typeOfFlat", "petsAllowed"]:
    df[col] = label_enc.fit_transform(df[col])
```

(5.6) Standardizing the numerical features.

All number-based property details were scaled to the same range for balanced analysis. This puts different measurements like square footage and amenity counts on equal footing when determining rental prices.

```
# --- STEP 7: STANDARDIZE NUMERICAL DATA ---
feature_cols = df.select_dtypes(include=[np.number]).columns.drop(['totalRent'])
scaler = StandardScaler()
df[feature_cols] = scaler.fit_transform(df[feature_cols])
```

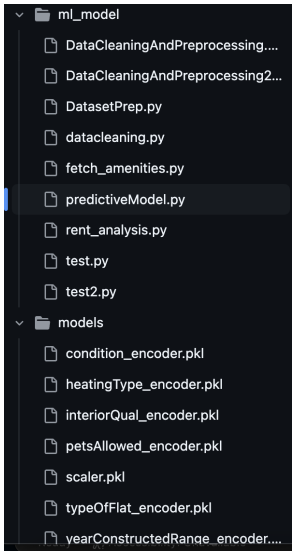
For complete implementation details

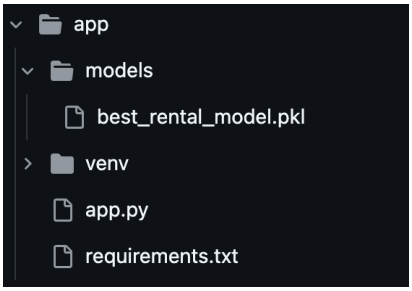
- DataCleaningAndPreprocessing.py
- DataCleaningAndPreprocessing2.py

(6) Modeling and Evaluation

(6.1) Environment Setup

- Creates directory structure for artifacts (/models, /reports/figures, app/models)





- Imports essential ML libraries (scikit-learn, pandas, visualization tools)

(6.2) Data Preparation for modeling

- Loads pre-processed rental data (cleaned_data.csv)
- Splits into features (X) and target (y = totalRent)
- 80/20 train-test split with fixed random state for reproducibility

```
# 3. Prepare data
X = df.drop('totalRent', axis=1)
y = df['totalRent']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

(6.3) Model Benchmarking

- Tests three algorithms
  - Linear Regression (baseline)
  - Decision Tree (non-linear)
  - Random Forest (ensemble)
- Tracks key metrics
  - R^2 (variance explained)
  - RMSE (error in original units)
  - MAE (interpretable error)
  - Cross-validated R^2 (generalization)

```
# 4. Model training and evaluation
models = {
    "Random Forest": RandomForestRegressor(random_state=42),
    "Decision Tree": DecisionTreeRegressor(random_state=42),
    "Linear Regression": LinearRegression()
}

results = []
for name, model in models.items():
    print(f"\nTraining {name}...")
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    results.append({
        'Model': name,
        'R2': r2_score(y_test, y_pred),
        'RMSE': np.sqrt(mean_squared_error(y_test, y_pred)),
        'MAE': mean_absolute_error(y_test, y_pred),
        'CV R2': cross_val_score(model, X_train, y_train, cv=5, scoring='r2').mean()
    })

# Display results
results_df = pd.DataFrame(results).sort_values('R2', ascending=False)
print("\n=== Model Performance ===")
print(results_df.to_string(index=False))
```

(6.4) Hyperparameter Optimization

- Grid search on Random Forest
  - Trees: 100-200
  - Depth: 10-20-unlimited
  - Splits: 2-5 samples
- 3-fold CV with R^2 scoring
- Parallelized (n_jobs=-1)

```
# 5. Hyperparameter tuning for best model (Random Forest)
print("\n=== Tuning Random Forest ===")
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [10, 20, None],
    'min_samples_split': [2, 5]
}

grid_search = GridSearchCV(RandomForestRegressor(random_state=42),
                           param_grid,
                           cv=3,
                           scoring='r2',
                           n_jobs=-1)
grid_search.fit(X_train, y_train)

best_rf = grid_search.best_estimator_
print(f"Best Parameters: {grid_search.best_params_}")
print(f"Tuned R2: {r2_score(y_test, best_rf.predict(X_test)):.4f}")
```

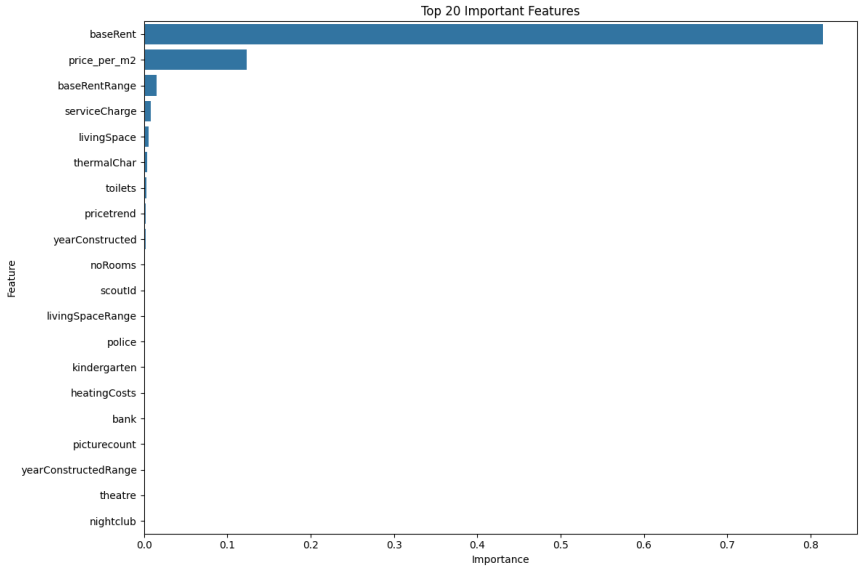
(6.5) Model Analysis

- Visualizes top 20 predictive features
- Saves plot as PNG for reporting
- Persists best model as pickle file

```
# 6. Feature importance visualization
plt.figure(figsize=(12, 8))
importances = best_rf.feature_importances_
features = pd.DataFrame({'Feature': X.columns, 'Importance': importances})
features = features.sort_values('Importance', ascending=False).head(20)

sns.barplot(x='Importance', y='Feature', data=features)
plt.title('Top 20 Important Features')
plt.tight_layout()

try:
    plt.savefig('../reports/figures/feature_importance.png')
    print("\nSaved feature importance plot to reports/figures/")
except Exception as e:
    print(f"\nCould not save plot: {str(e)}")

plt.close()

# 7. Save best model
joblib.dump(best_rf, '../models/best_rental_model.pkl')
print("\n=== Best Model Saved ===")
print("Saved to: ../models/best_rental_model.pkl")
```

(6.6) Output Artifacts

- Serialized model (best_rental_model.pkl)
- Feature importance visualization
- Console performance report



(6.7) Model performance and the evaluation

```
=== Model Performance ===
            Model       R2       RMSE        MAE    CV R2
    Random Forest 0.946649  86.283758  47.659658 0.933176
    Decision Tree 0.915140 108.819975  63.978986 0.888604
Linear Regression 0.617427 231.054100 111.516536 0.763459

=== Tuning Random Forest ===
Best Parameters: {'max_depth': 20, 'min_samples_split': 2, 'n_estimators': 200}
Tuned R2: 0.9483

Saved feature importance plot to reports/figures/

=== Best Model Saved ===
Saved to: ../models/best_rental_model.pkl
thusithawijesooriya@Thusithas-MacBook-Air ml_model %
```

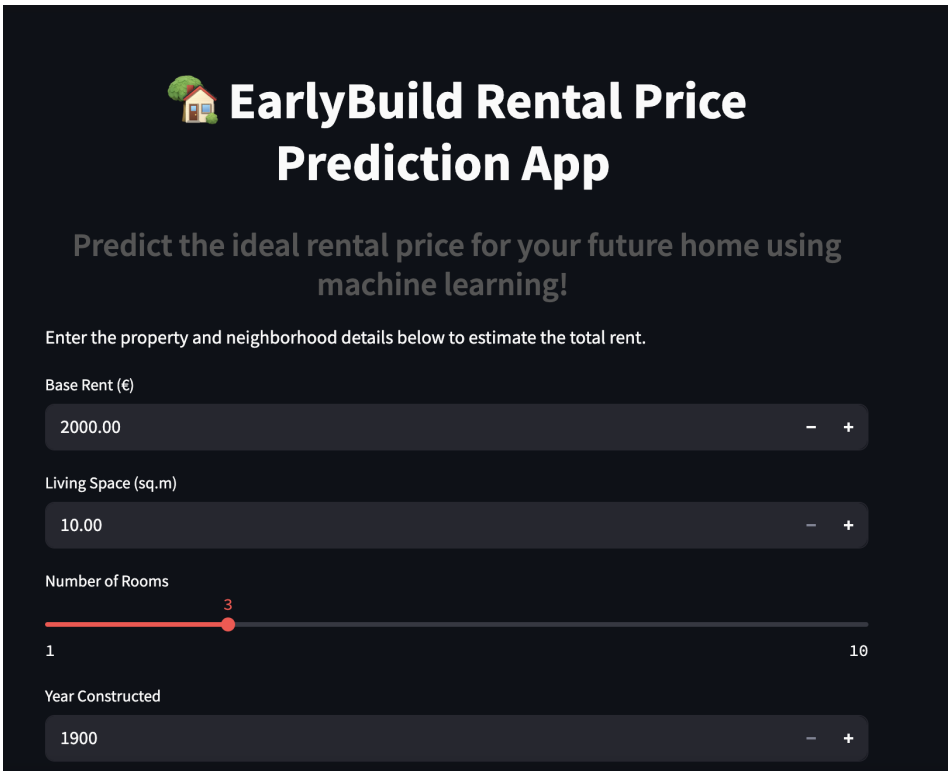
Metric	Random Forest	Decision Tree	Linear Regression	Ideal Value	What It Means
R^2	0.947	0.915	0.617	1.0	How well the model explains price variation (94.7% for RF)
RMSE (€)	86.28	108.82	231.05	0	Average prediction error in euros
MAE (€)	47.66	63.98	111.52	0	Average absolute error
CV R^2	0.933	0.889	0.763	Close to R^2	How well models generalize

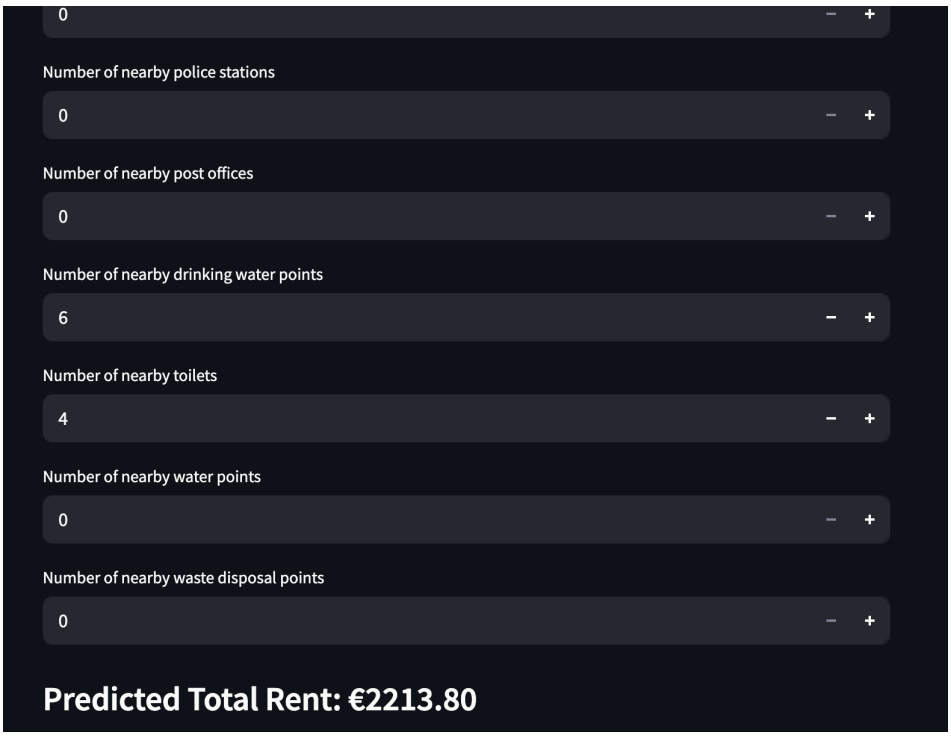
1. **Random Forest is Best Performer**
    - Explains **94.7%** of rent price variation (near-perfect R^2)
    - Predictions are off by **€86 on average** (RMSE)
    - **€47.66** typical absolute error (MAE)
    - Minimal overfitting (CV R^2 0.933 vs Test R^2 0.947)
2. **Decision Tree (Good but Less Robust)**
    - **91.5%** variance explained.
    - Larger gap between CV (0.889) and test R^2 (0.915) suggests mild overfitting.
3. **Linear Regression (Baseline)**
    - Only explains **61.7%** of variance.
    - High errors (€231 RMSE) show linear assumptions don't fit this data well.

(7) Model Deployment

To make our rental price prediction model accessible and interactive, the model has been deployed using **Streamlit**, a lightweight Python web framework ideal for rapid prototyping of machine learning applications.

Link to the deployed Earlybuil Rental Price Prediction App
https://keddtpoy5m8cz4ran7sgkh.streamlit.app/



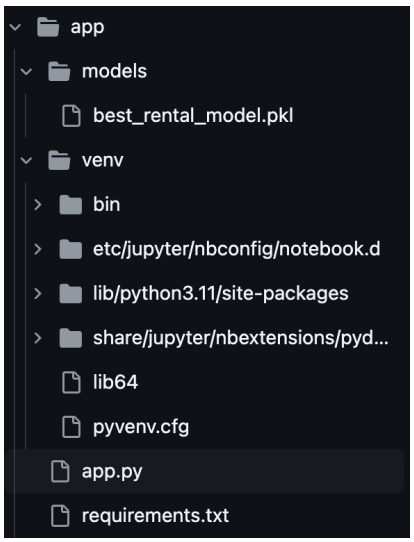


(7.1) Deployment Stack

- **User Interface** - Developed using Streamlit for intuitive and responsive inputs.
- **Backend Logic** - Powered by a trained random forest regression model, serialized using joblib.
- **Environment Management** - A dedicated Python virtual environment named vnev, created under the app/ directory, ensures clean dependency management and reproducibility.

(7.2) Virtual Environment - vnev/

To avoid conflicts and ensure consistency across different machines, a **virtual environment** named vnev has been created inside the app/ folder. This encapsulated all necessary libraries and packages specific to the application.



(7.3) App Workflow Overview

1. **Input Collection** - The app takes in user inputs such as area, rooms, floor level, building age, heating type, and distance to metro.

2. **Feature Engineering** - Derived features like price_per_m2 and location_score are computed.

3. **Preprocessing**

- Categorical values are encoded using pre-saved label encoders.
- Numerical features are scaled using a pre-trained StandardScaler.

4. **Prediction** - Inputs are passed to the model to generate a rental price prediction, which is then displayed in real-time.

(8) Results and Insights

Figure 1. Average Rent by Floor and Lift Availability

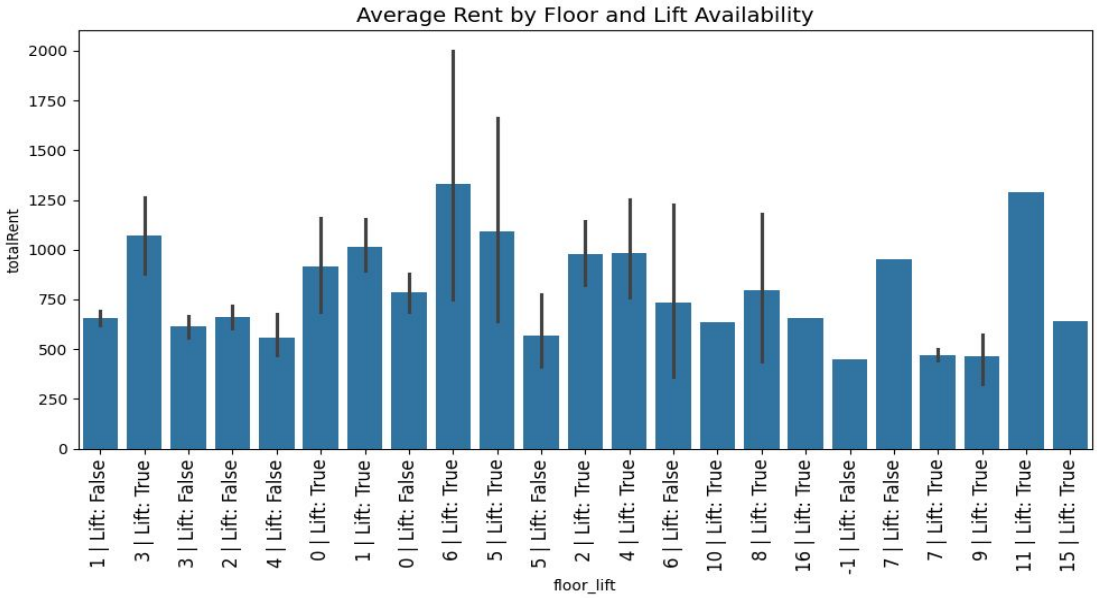


Figure 2. Balcony Distribution

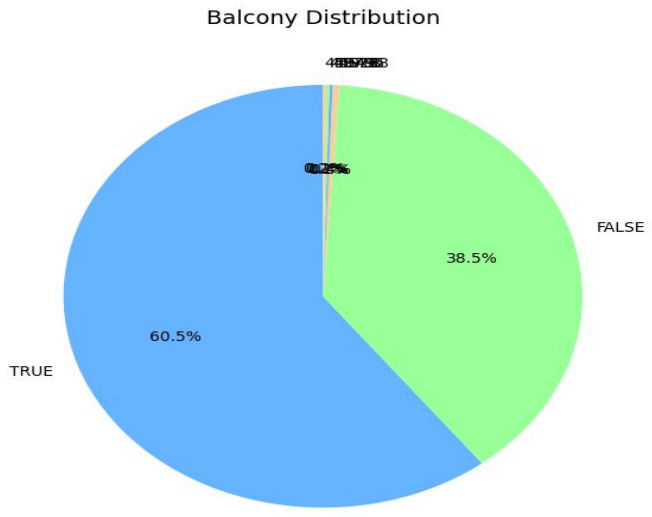


Figure 3. Distribution of Heating Types

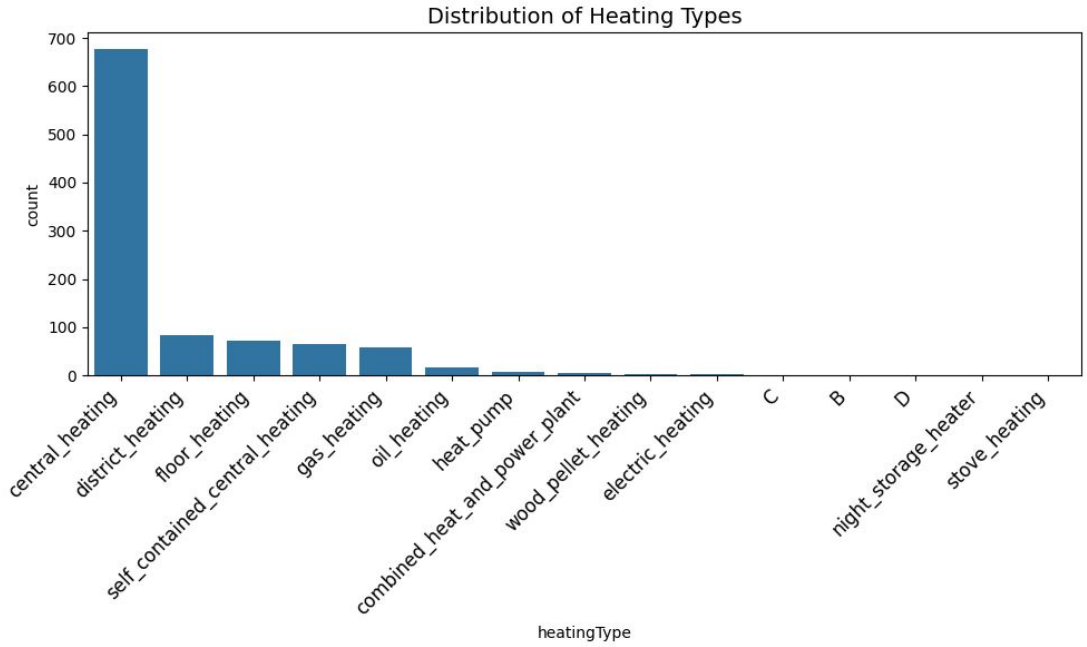


Figure 4. Garden Distribution

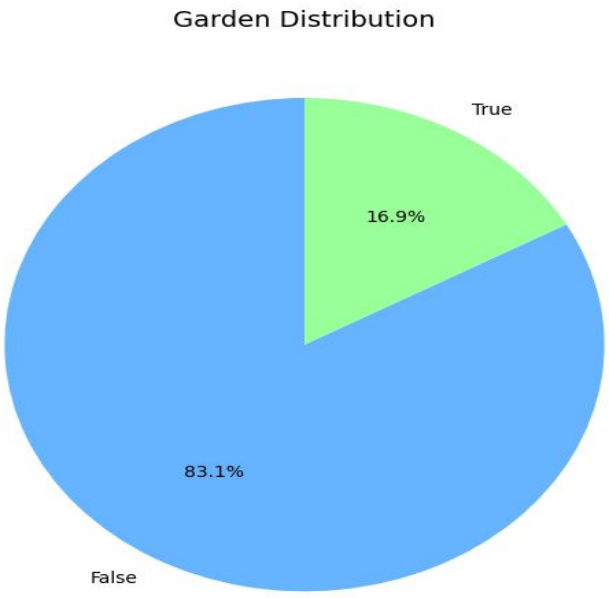


Figure 5. Heatmap of Amenities Correlation with Total Rent

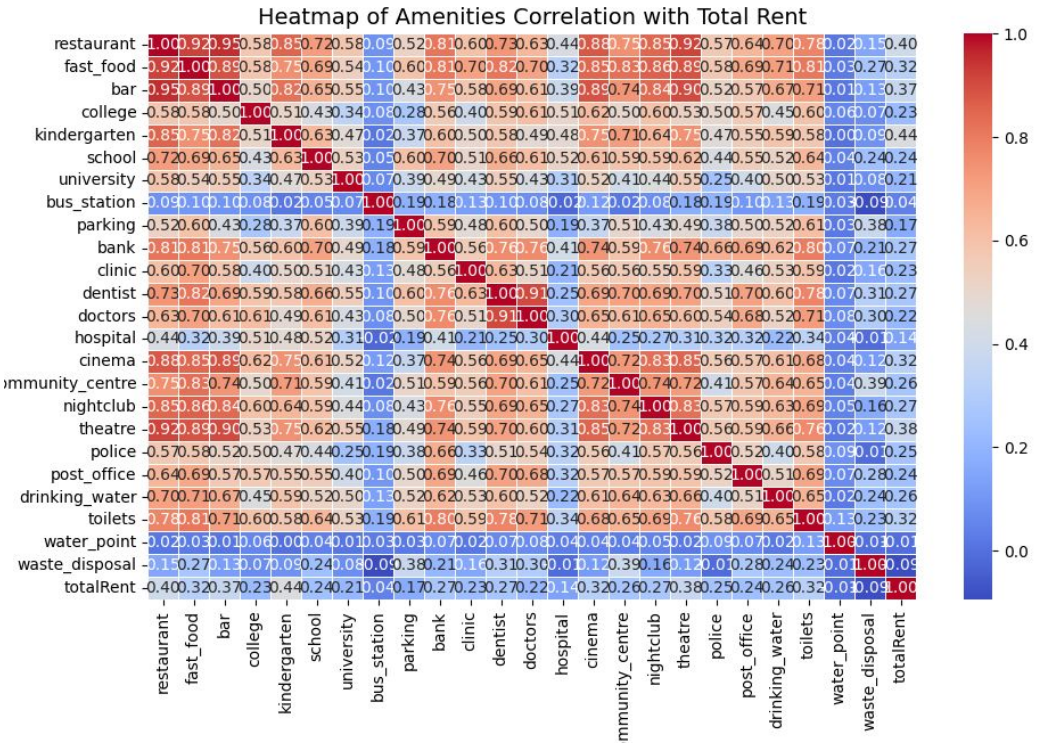


Figure 6. Property Condition vs Interior Quality

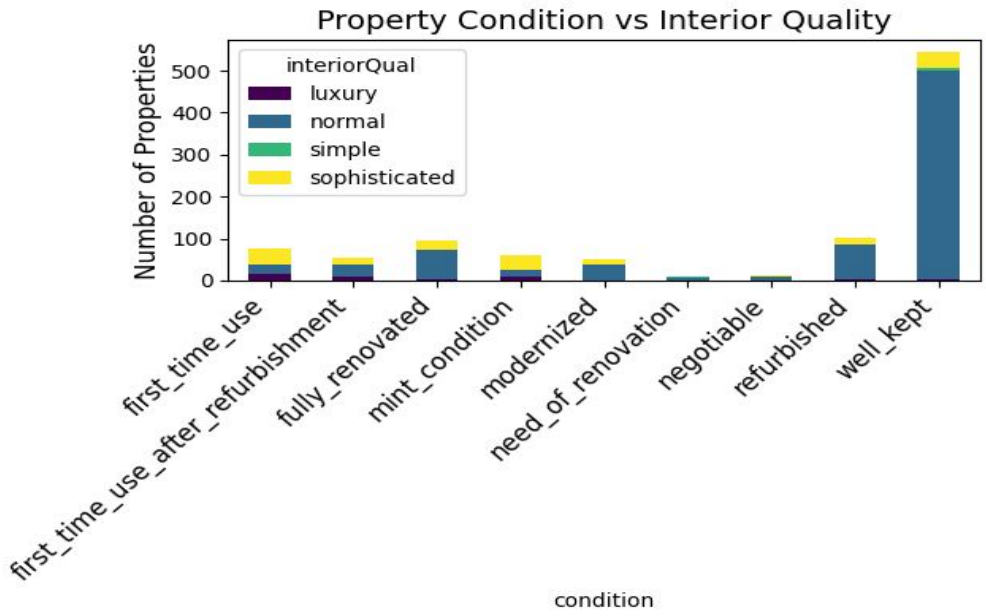


Figure 7. Rent Distribution by Living Space Range

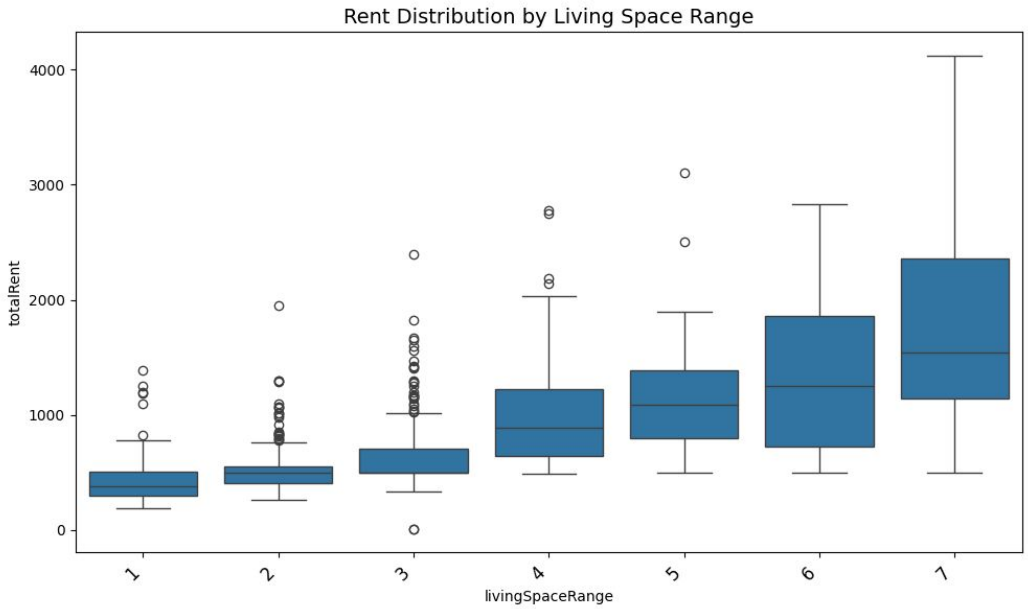


Figure 8. Total Number of Amenities within 2km (Normalized)

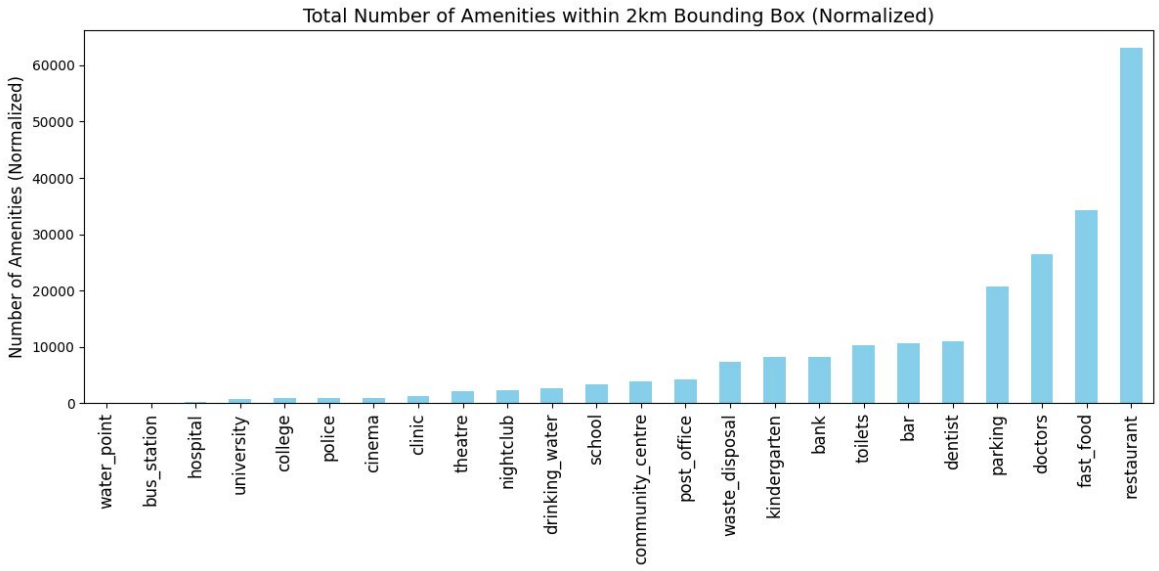


Figure 9. Total Rent vs Living Space by Type of Flat

Figure 10. Total Rent vs Total Amenities within 2km

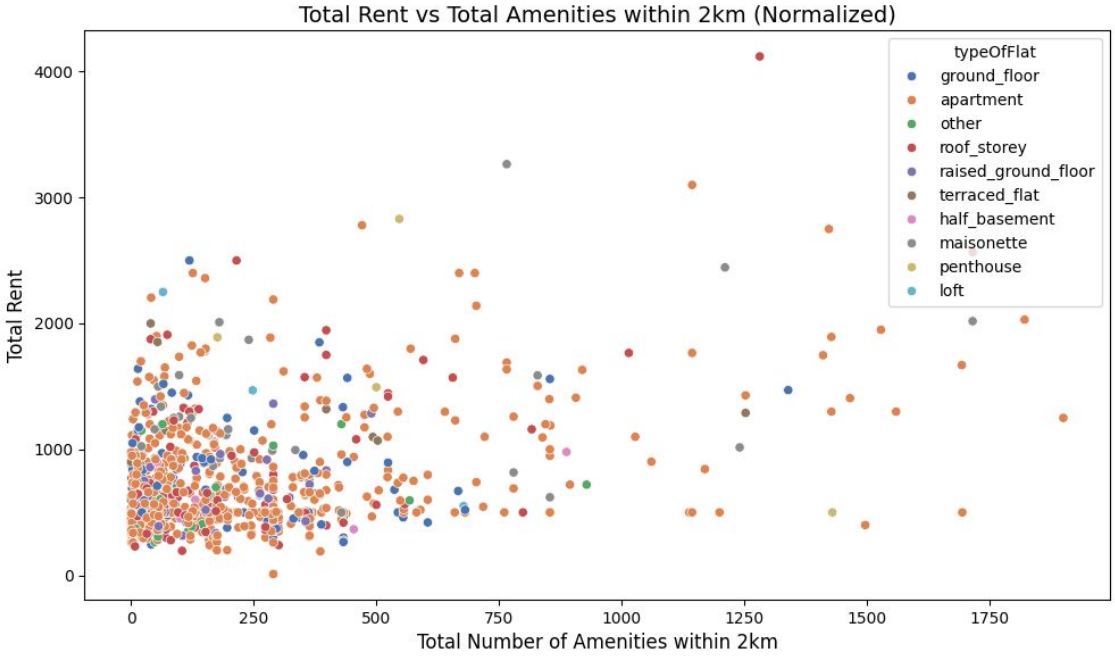


Figure 11. Bubble Chart: Rent vs Amenities (Bubble = Living Space)

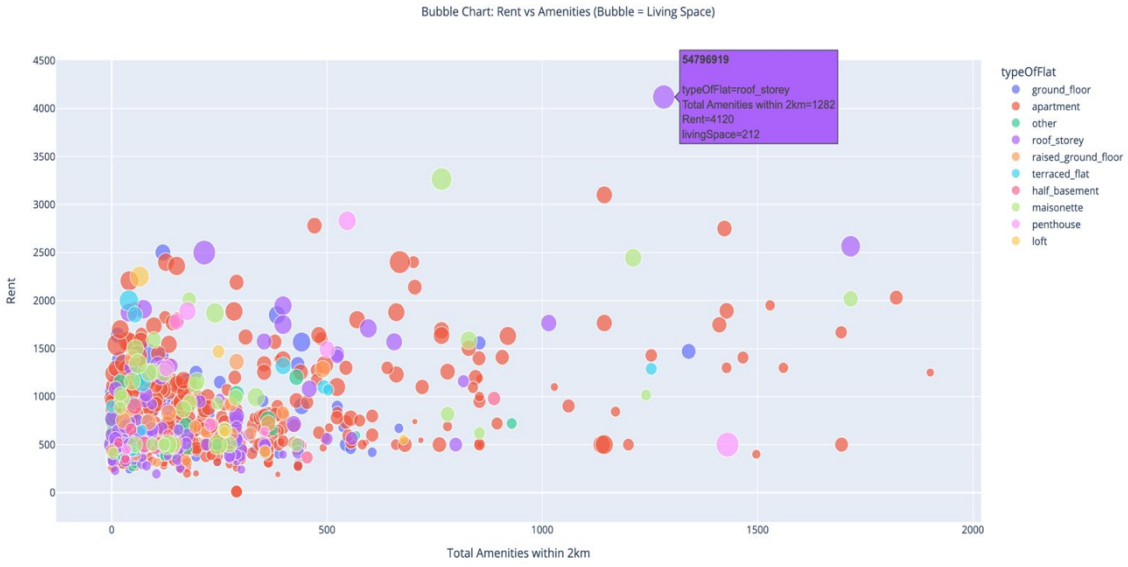
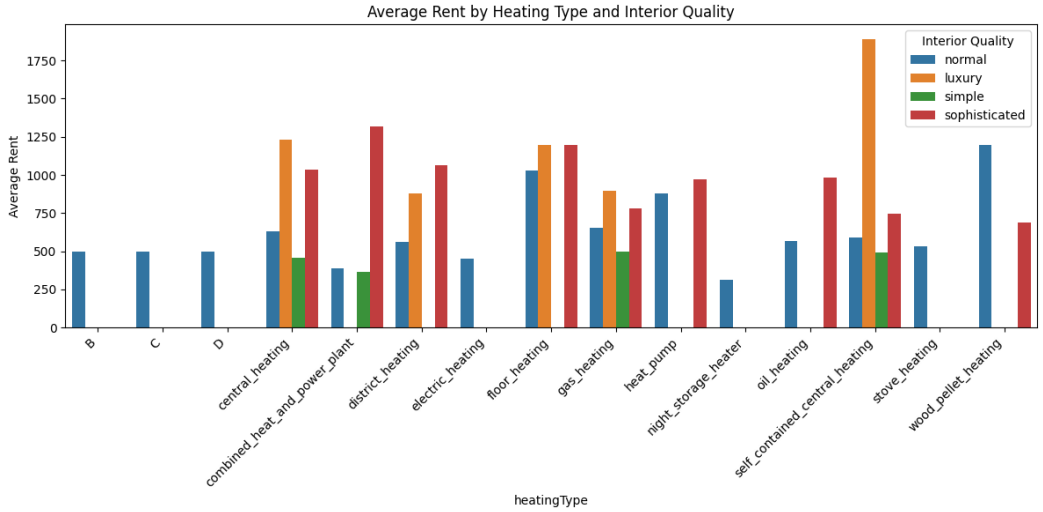


Figure 12. Correlation Between Boolean Features and Total Rent

Figure 13. Average Rent by Heating Type and Interior Quality



- Key Insights Derived from Analysis

1. Average Rent by Floor and Lift Availability

Figure 1 presents a bar chart illustrating the relationship between average rent, floor level, and lift availability. It is evident that apartments situated on higher floors generally command higher rents when a lift is available, suggesting a premium associated with accessibility and convenience. Conversely, properties on similar floors without lift access tend to be priced lower. This visualization underscores the role of vertical accessibility in influencing rental prices.

2. Balcony Distribution

Figure 2 shows a pie chart highlighting the distribution of properties based on the presence of a balcony. Approximately 60.5% of the listings include a balcony, while 38.5% do not. This indicates that balconies are a commonly available amenity, potentially contributing to higher tenant satisfaction and increased rental desirability.

3. Distribution of Heating Types

Figure 3 displays the frequency distribution of heating systems installed in the properties. Central heating is the most prevalent, followed by district and floor heating. Other systems, such as gas heating, oil heating, and heat pumps, appear with significantly lower frequencies. The visualization helps identify standard heating practices in the housing market and can inform infrastructure or energy policy considerations.

4. Garden Distribution

Figure 4 is a pie chart representing the availability of garden space across listings. Only 16.9% of properties offer garden access, while the majority (83.1%) do not. This suggests that garden space is a relatively scarce amenity, which could act as a differentiating factor for property valuation, particularly in suburban or family-oriented housing markets.

5. Heatmap of Amenities Correlation with Total Rent.

Figure 5 provides a heatmap that quantifies the correlation between proximity to various amenities and total rent. The analysis reveals that proximity to restaurants, fast food outlets, bars, banks, and clinics shows a relatively stronger correlation with rent, suggesting that convenience to lifestyle and health-related amenities positively influences rental values.

6. Property Condition vs Interior Quality

Figure 6 illustrates a grouped bar chart comparing property condition categories with interior quality ratings. The 'well-kept' condition paired with 'normal' interior quality is the most dominant combination. While luxury and sophisticated interiors appear less frequently, they are often associated with recently renovated or first-time use properties, reinforcing the link between renovation status and premium interior offerings.

7. Rent Distribution by Living Space Range

Figure 7 showcases a boxplot distribution of rents across different living space ranges. A clear trend of increasing rent with larger living spaces is observed, along with greater variability in rent for higher size categories. This supports the intuitive understanding that living space is a key determinant of rent, though variance in higher ranges indicates additional influencing factors such as location or amenities.

8. Total Number of Amenities within 2km (Normalized)

Figure 8 presents a normalized count of various amenities within a 2km radius of each property. Amenities such as restaurants, fast food, parking, and healthcare-related services are the most abundant. This distribution provides insight into neighborhood infrastructure and urban density, which may impact tenant preferences and rent.

9. Total Rent vs Living Space by Type of Flat

Figure 9 shows a scatter plot of total rent against living space, categorized by the type of flat. As expected, larger flats tend to have higher rent. The scatter also highlights

specific flat types, such as penthouses and maisonettes, which cluster at the higher end of both rent and living space, indicating their premium positioning in the rental market.

10. Total Rent vs Total Amenities within 2km

Figure 10 investigates the relationship between total rent and the number of nearby amenities. While some high-rent listings do correlate with a greater number of amenities, the overall trend remains weak. This suggests that while access to amenities plays a role, it is likely not the sole factor influencing rent.

11. Bubble Chart: Rent vs Amenities (Bubble = Living Space)

Figure 11 enhances the previous scatter plot by incorporating bubble size to represent living space. This multidimensional visualization reveals that properties with higher rents are often those with both more amenities nearby and greater living space. It offers a more comprehensive view of how multiple property attributes interact to influence rental pricing.

12. Correlation Between Boolean Features and Total Rent

Figure 12 presents a heatmap illustrating the correlation between various binary apartment features and total rent. Among the features considered, the presence of a balcony, guest toilet, and kitchen show a relatively stronger positive correlation with rental price. These amenities likely enhance the livability and functionality of a property, thereby justifying higher rents. Conversely, features such as pets allowed or handicap accessibility exhibit minimal correlation, suggesting that while these are important for inclusivity and personal preference, they may not significantly influence rent levels from a market pricing perspective.

13. Average Rent by Heating Type and Interior Quality

Figure 13 explores the combined influence of heating type and interior quality on average rent using a grouped bar chart. The results clearly indicate that luxury and sophisticated interiors are consistently associated with higher rents across all heating types, with floor heating + luxury quality fetching the highest values. On the other hand, normal or simple interiors paired with basic heating types, such as central or gas heating, show considerably lower average rents. This reinforces the notion that interior quality plays a substantial role in rental pricing, often outweighing the heating method in impact.

(9) Recommendations

- **Integrate Predictive Model into Pricing Workflow**
  EarlyBuild GmbH should incorporate the final Random Forest model into their pricing recommendation system to assist agents and property owners in setting competitive and fair rental prices.

- **Enhance Listings with Amenity Scores**
  Display an "Amenity Score" based on the number and type of nearby facilities to help tenants understand the location's advantages and justify rental prices.

- **Prioritize High-Impact Features**
  Features such as interior quality, presence of a lift, balcony, and heating type have a high influence on rental price. These should be emphasized in listings and potentially upgraded during renovations for value addition.

- **Customized Tenant Recommendations**
  Use clustering techniques (future suggestion) to group similar property preferences and recommend listings to users based on location and amenity proximity preferences.

(10) Conclusion

This project, conducted in collaboration with **Earlybuild GmbH**, successfully demonstrated the complete lifecycle of building and deploying a real estate price prediction system. From data pre-processing and feature engineering to training a robust regression model and deploying it via a Streamlit web application, the project provided valuable hands-on experience in applied machine learning. Integrating tools such as Scikit-learn, Pandas, and Streamlit within a virtual environment allowed for streamlined development and deployment. Despite encountering challenges with data inconsistencies and deployment configurations, we gained critical insights into the importance of data quality, user interface design, and reproducibility. Overall, this collaboration offered a real-world perspective on predictive analytics and has equipped us with the skills and knowledge to contribute meaningfully to future projects in the domain.

(11) References

Dataset

Corrieaar. (2018). *Apartment rental offers in Germany* [Dataset]. Kaggle. https://www.kaggle.com/datasets/corrieaar/apartment-rental-offers-in-germany

Online Resources

OpenStreetMap contributors. (n.d.). *OpenStreetMap* [Database]. https://www.openstreetmap.org/

OpenStreetMap Wiki. (n.d.). *Overpass API*. https://wiki.openstreetmap.org/wiki/Overpass_API

OpenStreetMap Wiki. (n.d.). *Key:amenity*. https://wiki.openstreetmap.org/wiki/Key:amenity

Software

Streamlit, Inc. (n.d.). *Streamlit* (Version 1.22.0) [Computer software]. https://streamlit.io/

Academic Materials

Mohawk College. (2025). *[Data Preparation]* MyCanvas. https://mycanvas.mohawkcollege.ca/courses/113934/assignments/1062209/submissions/152483

Mohawk College. (2025). *[Data Analysis (Evaluation and/or Deployment]* MyCanvas. https://mycanvas.mohawkcollege.ca/courses/113934/assignments/1062208/submissions/152483

Appendix

GitHub Repository - https://github.com/earlybuild/Mohawk-ML-Project-Q2

- DatasetPreparation
  https://github.com/earlybuild/Mohawk-ML-Project-Q2/blob/main/ml_model/DatasetPrep.py

- Data Clening and Preprocessing
  https://github.com/earlybuild/Mohawk-ML-Project-Q2/blob/main/ml_model/DataCleaningAndPreprocessing.py

  https://github.com/earlybuild/Mohawk-ML-Project-Q2/blob/main/ml_model/DataCleaningAndPreprocessing2.py

- Predictive Modeling

  https://github.com/earlybuild/Mohawk-ML-Project-Q2/blob/main/ml_model/predictiveModel.py

- Streamlit app(Deployment)

  https://github.com/earlybuild/Mohawk-ML-Project-Q2/blob/main/app/app.py