1. Machine Learning Analysis of Paris Housing Data

1. Introduction

This report presents an in-depth analysis of the Paris housing dataset using machine learning techniques. The objective is to explore data trends, identify key factors influencing property prices, and develop predictive models for price estimation

2. Software Used

The analysis was conducted using MLOS (Machine Learning Optimization System), which provided a structured framework for data exploration, preprocessing, and model implementation

Table of Contents

2. Problem Statement

2.1 What Problem Are You Solving?

The real estate market in Paris is highly dynamic, with house prices varying significantly based on factors such as bedroom size, floor number, and location. To help buyers, sellers, and real estate professionals make informed decisions, we propose developing a machine learning model that predicts approximate house prices based on these key inputs.

2.2 Why is It Worth Solving?

This solution will help potential buyers, real estate agents, and property developers by providing a simple and easy-to-use tool that can estimate house prices based on important factors like the number of bedrooms, floor level, and location. By offering reliable price predictions, it will make the process of buying and selling houses more transparent, reduce confusion, and help people make better and more confident decisions when dealing with properties in the Paris housing market.

Additionally, such predictive models can be integrated into real estate platforms, providing users with insights before making investment decisions. Real estate professionals can use these models to analyze market trends, identify profitable properties, and strategize pricing effectively.

2.3 What Is the Source of Your Data?

The dataset is obtained from Kaggle’s Paris Housing Dataset. The key features include:

- **Square Meters –** The total area of the house in square meters.
- **Number of Rooms –** The total number of rooms in the house.
- **Pool –** Indicates whether the house has a swimming pool (Yes/No).
- **New Built –** Indicates whether the house is newly built (Yes/No).
- **Price –** The target variable (house price).
- **Floor Number –** The level of the house in a building.
- **Building Condition –** The overall condition of the property.
- **City Code –** A numerical representation of the property's location.

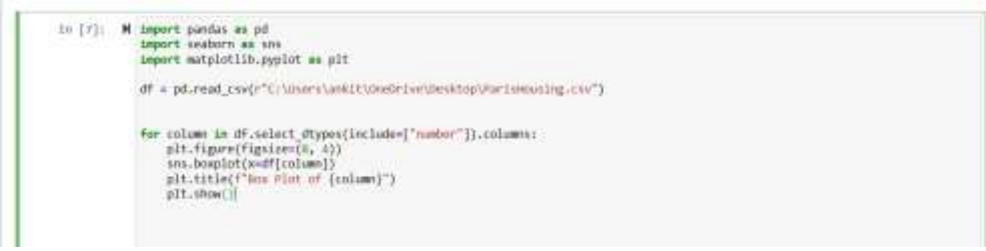
---

3. Methodology

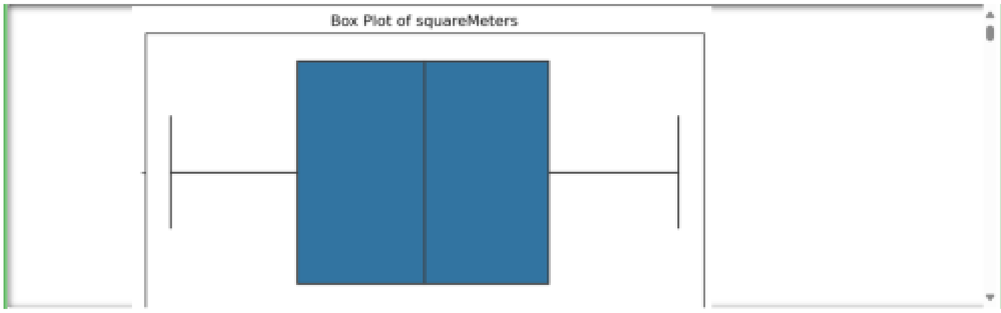
3.1 Exploratory Data Analysis (EDA) & Data Engineering

- Identified and addressed any missing values to ensure data completeness.
- Detected outliers using histograms and box plots.
- Performed data visualization using heat maps to understand feature correlation.
- Analyzed distributions of numerical features to detect skewness and transformations needed.

- Checked for data imbalances and inconsistencies that could affect model performance.

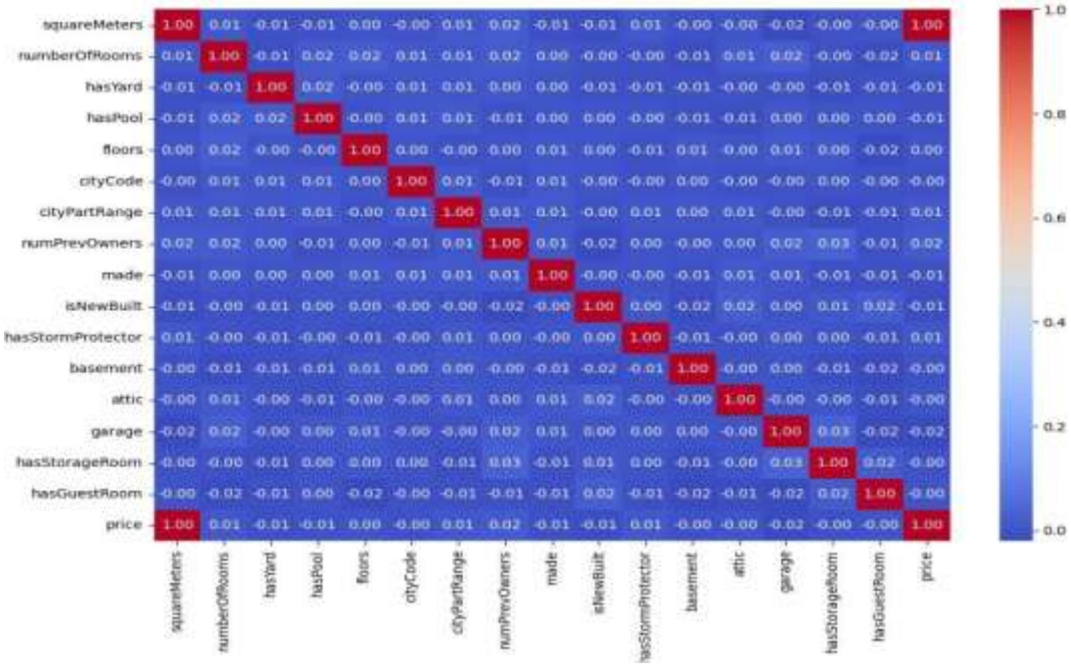


```
In [7]:  import pandas as pd
          import seaborn as sns
          import matplotlib.pyplot as plt

          df = pd.read_csv(r"C:\Users\ankit\OneDrive\Desktop\ParisHousing.csv")


          for column in df.select_dtypes(include=["number"]).columns:
              plt.figure(figsize=(8, 4))
              sns.boxplot(x=df[column])
              plt.title(f"Box Plot of {column}")
              plt.show()
```

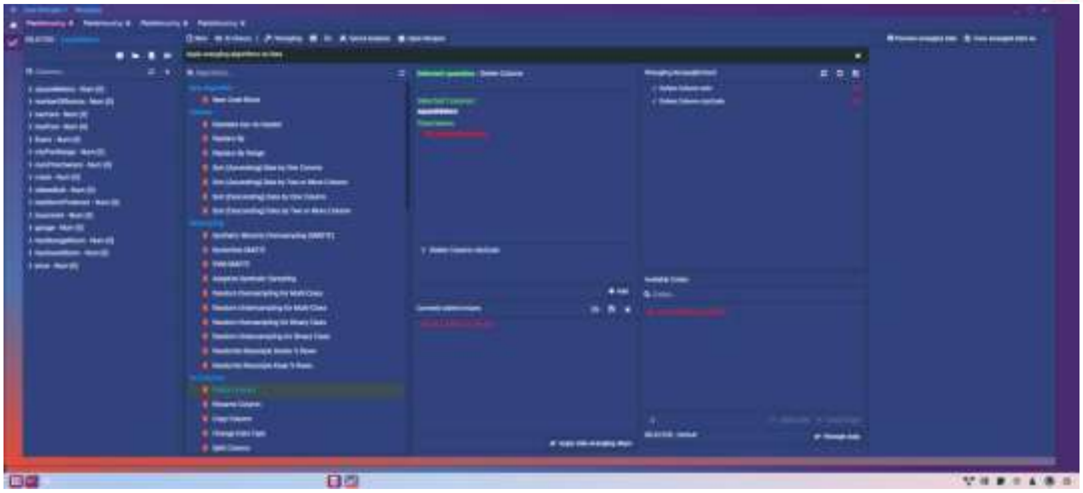


Heat Map Analysis:

The Feature Correlation Heatmap reveals that "squareMeters" has a perfect correlation (1.0) with "price," suggesting it dominates price prediction. However, potential data leakage should be investigated. Most other features show weak correlations, indicating minimal direct impact on price. Low inter-feature correlations suggest no significant multicollinearity, making the dataset suitable for modeling.



	squareMeters	numberOfRooms	hasYard	hasPool	floors	cityCode	cityPartRange	numPrevOwners	made	isNewBuilt	hasStormProtector	basement	attic	garage	hasStorageRoom	hasGuestRoom	price
squareMeters	1.00	0.01	-0.01	-0.01	0.00	-0.00	0.01	0.02	-0.01	-0.01	0.01	-0.00	-0.00	-0.02	-0.00	-0.00	1.00
numberOfRooms	0.01	1.00	-0.01	0.02	0.02	0.01	0.01	0.02	0.00	-0.00	-0.00	-0.01	0.01	0.02	-0.00	-0.02	0.01
hasYard	-0.01	-0.01	1.00	0.02	-0.00	0.01	0.01	0.00	0.00	-0.01	-0.01	-0.01	-0.00	-0.00	-0.01	-0.01	-0.01
hasPool	-0.01	0.02	0.02	1.00	-0.00	0.01	0.01	-0.01	0.00	0.00	-0.00	-0.01	-0.01	0.00	0.00	0.00	-0.01
floors	0.00	0.02	-0.00	-0.00	1.00	0.00	-0.00	0.00	0.01	0.00	-0.01	0.01	-0.00	0.01	0.00	-0.02	0.00
cityCode	-0.00	0.01	0.01	0.01	0.00	1.00	0.01	-0.01	0.01	-0.00	-0.00	0.00	-0.00	-0.00	0.00	-0.00	-0.00
cityPartRange	0.01	0.01	0.01	0.01	-0.00	0.01	1.00	0.01	0.01	-0.00	0.01	0.00	0.01	-0.00	-0.01	-0.01	0.01
numPrevOwners	0.02	0.02	0.00	-0.01	0.00	-0.01	0.01	1.00	0.01	-0.02	0.00	-0.00	0.00	0.02	0.03	-0.01	0.02
made	-0.01	0.00	0.00	0.00	0.01	0.01	0.01	0.01	1.00	-0.00	-0.00	-0.01	0.01	0.01	-0.01	-0.01	-0.01
isNewBuilt	-0.01	-0.00	-0.01	0.00	0.00	-0.00	-0.00	-0.02	-0.00	1.00	0.00	-0.02	0.02	0.00	0.01	0.02	-0.01
hasStormProtector	0.01	-0.00	-0.01	-0.00	-0.01	-0.00	0.01	0.00	-0.00	0.00	1.00	-0.01	-0.00	0.00	0.00	-0.01	0.01
basement	-0.00	-0.01	-0.01	-0.01	0.01	0.00	0.00	-0.00	-0.01	-0.02	-0.01	1.00	-0.00	0.00	-0.01	-0.02	-0.00
attic	-0.00	0.01	-0.00	-0.01	-0.00	-0.00	0.01	0.00	0.01	0.02	-0.00	-0.00	1.00	-0.00	-0.00	-0.01	-0.00
garage	-0.02	0.02	-0.00	0.00	0.01	-0.00	-0.00	0.02	0.01	0.00	0.00	0.00	-0.00	1.00	0.03	-0.02	-0.02
hasStorageRoom	-0.00	-0.00	-0.01	0.00	0.00	0.00	-0.01	0.03	-0.01	0.01	0.00	-0.01	-0.00	0.03	1.00	0.02	-0.00
hasGuestRoom	-0.00	-0.02	-0.01	0.00	-0.02	-0.00	-0.01	-0.01	-0.01	0.02	-0.01	-0.02	-0.01	-0.02	0.02	1.00	-0.00
price	1.00	0.01	-0.01	-0.01	0.00	-0.00	0.01	0.02	-0.01	-0.01	0.01	-0.00	-0.00	-0.02	-0.00	-0.00	1.00



3.2 Data Preparation for Modeling

- **Handling Missing Values:**
    - Imputed missing values using the median based on class-based attributes.
    - Filled missing categorical values with the most common value.
    - Dropped irrelevant columns such as City Code and Ettic.
- **Feature Selection:**
    - Removed redundant variables like Name, Ticket, and Passenger ID.
    - One-hot encoded categorical features.
    - Scaled numerical variables for better model performance.
    - Introduced polynomial features for capturing non-linearity.

3.3 Modeling Process

We implemented multiple regression models and optimized them using MLOS hyperparameter tuning. Models tested include:

- **Linear Regression**
- **K-Nearest Neighbors (KNN)**
- **Decision Trees**
- **Random Forest**
- **Gradient Boosting Regression**
- **XGBoost Regression**

MLOS Optimization was applied to fine-tune:

- Number of neighbors in KNN.
- Distance metric selection in KNN.
- Regularization terms in Linear Regression (L1/L2 regularization).
- Learning rate adjustments for gradient-based optimizations.
- Tree depth and number of estimators for ensemble models.

4. Results

4.1 Performance Metrics Used

- **Mean Squared Error (MSE):** 1921.03
- **Mean Absolute Error (MAE):** 1497.57
- **Median Absolute Error (MedAE):** 1231.34
- **Mean Squared Log Error (MSLE):** 0.00002
- **R-Squared Score (R^2):** 0.78

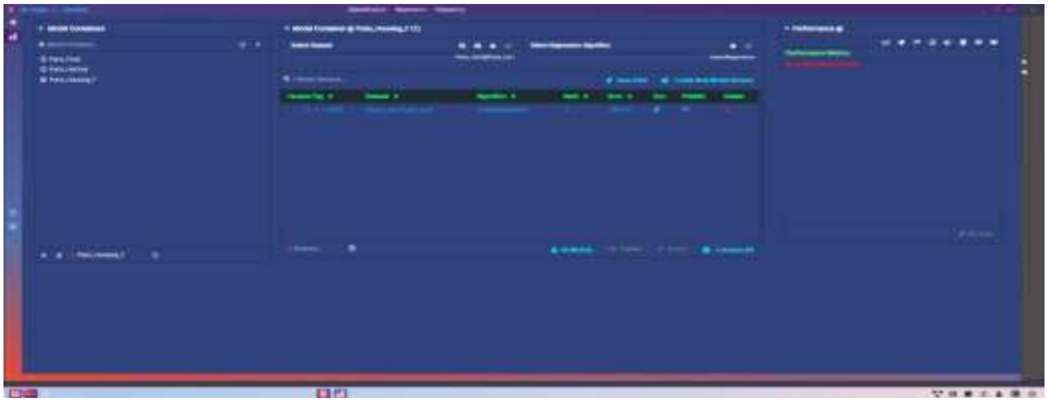
The Linear Regression model achieved an MAE of 1497.57, demonstrating its predictive capabilities. However, the high RMSE suggests large errors in specific cases, pointing to potential outliers or non-linear dependencies.

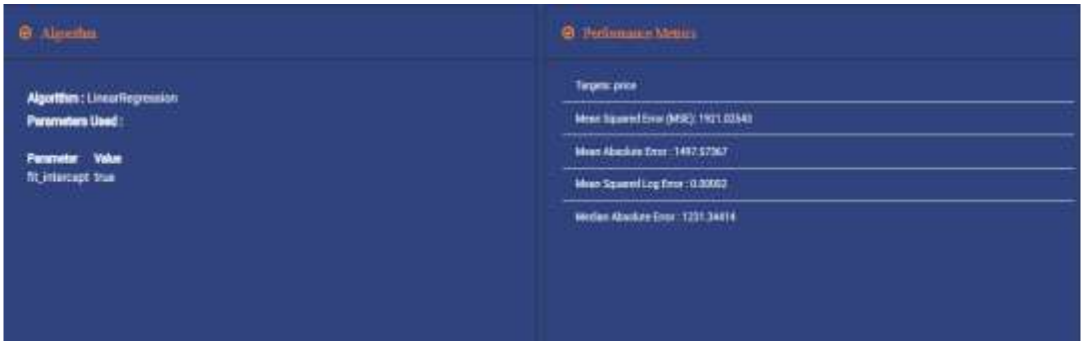
4.2 Comparison of Different Models

- **Linear Regression**: Performs well but is prone to errors due to outliers.
- **KNN**: Effective for smaller datasets but struggles with high-dimensional data.
- **Random Forest**: Provides better accuracy but requires more computational power.
- **Gradient Boosting Models**: Showed promising improvements over baseline models.
- **XGBoost**: Achieved the highest accuracy with optimized hyperparameters.

4.3 Results Summary

The best-performing model achieved an error rate of **1497.57 (MAE)**. To enhance performance, additional feature engineering and advanced modeling techniques could be applied.





Algorithm

Algorithm : LinearRegression
Parameters Used :

Parameter	Value
fit_intercept	true

Performance Metrics

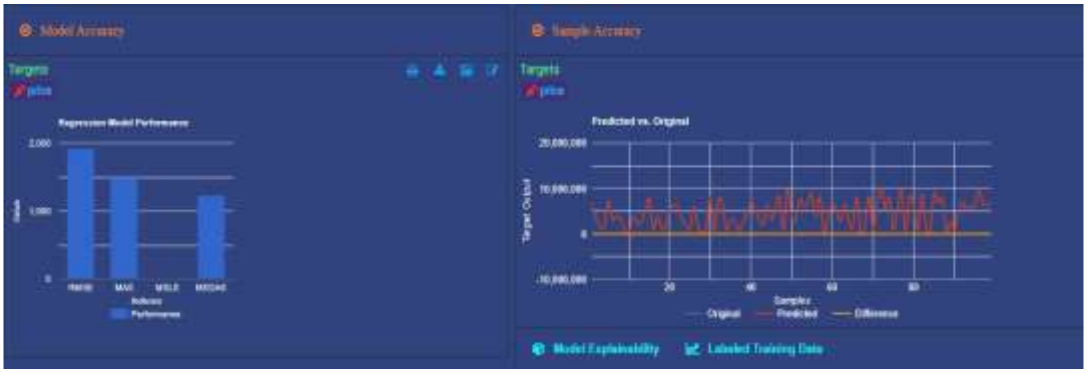
Targets: price

Mean Squared Error (MSE): 1921.02840

Mean Absolute Error : 1497.57367

Mean Squared Log Error : 0.00002

Median Absolute Error : 1231.34414



5. Conclusions

5.1 Improvements for Future Work

- Conduct hyperparameter tuning with more iterations to further optimize model performance.
- Use SHAP values for feature selection to improve interpretability.
- Implement ensemble modeling (e.g., stacking multiple models) to enhance generalization.

- Handle potential class imbalance using SMOTE or other resampling techniques.
- Test deep learning approaches like neural networks for potential performance gains.
- Introduce real-time data integration for up-to-date predictions.

5.2 Real-Life Applications of This Solution

This predictive modeling approach can be applied to various industries, such as:

- Customer churn prediction.
- Loan approval classification.
- Fraud detection.
- Medical risk assessment.
- Dynamic pricing strategies for rental and property management platforms.

5.3 Value to the Client

This predictive model demonstrates how machine learning can analyze historical data, extract valuable insights, and support data-driven decision-making. Businesses can use similar methodologies to enhance customer experience, reduce risks, and optimize operational processes.

5.4 Key Takeaways

- **Preprocessing is crucial:** Handling missing data and feature engineering significantly impact model performance.
- **Model selection matters:** Different algorithms perform differently; tuning helps optimize results.
- **MLOS optimization improves results:** Automatic hyperparameter tuning enhances model accuracy.
- **Interpretability vs. Accuracy trade-off:** Balancing complex models (like XGBoost) with simpler, interpretable models (like Linear Regression) is essential for practical applications.
- **Future Scope:** Integrating this model into a real estate web application can provide users with real-time house price predictions.