Predicting property sale prices in France using machine learning: A study based on the DVF dataset

Yvan Roh
HEC Lausanne / University of Lausanne
yvan.roh@unil.ch
Student ID: 22-216-170

Advanced Programming 2025 – Individual project

Prof.: Simon Scheidegger
Date: November 20, 2025

Abstract

Property valuation is a critical component of real estate markets, yet traditional hedonic models often fail to capture the complex non-linear relationships between property characteristics and sale prices. This study develops a comprehensive machine learning pipeline to predict residential property prices in France using the *Demandes de Valeurs Foncières* (DVF) dataset, comprising approximately 18.1 million transactions recorded between 2020 and 2024. The methodology encompasses data preprocessing, temporal feature engineering, and comparative evaluation of five regression models: Linear Regression, Ridge, Lasso, Random Forest, and XGBoost. A temporal validation strategy was implemented, splitting data chronologically into training (2020-2022), validation (2023), and test (2024) sets. Random Forest emerged as the optimal model, achieving a coefficient of determination (R^2) of 0.77 on the test set with a median absolute error of €36,878. In contrast, linear models exhibited limited predictive power ($R^2 \approx 0.26$), while XGBoost demonstrated severe overfitting ($R^2 \approx 0.99$). The complete pipeline, including automated CI/CD deployment via GitHub Actions, is available as open-source software. Results demonstrate that ensemble tree-based methods substantially outperform traditional regression approaches for large-scale property price prediction, providing actionable insights for investors, policymakers, and real estate professionals.

Keywords: real estate, ML, temporal data, data science, regression, open data, France

I Introduction

The valuation of real estate properties is a key topic in economics and data science, as housing prices reflect both market dynamics and regional development. Accurate prevision or estimation models can support investors, policymakers, and financial institutions in understanding spatial and temporal variations within the housing market.
In this study, will be analysed property transactions recorded in the **Demandes de Valeurs Foncières** (DVF[1]) dataset, published by the French Directorate-General of Public Finances. This dataset provides detailed information on real estate sales in France between 2020 and 2024, including variables such as location, property type, surface area, and sale value. With approximately **18.1 million transactions** over five years, this dataset is one of the most comprehensive open source databases on real estate transactions in Europe, enabling large-scale analysis of the dynamics of the French property market, even though it does not include information on the current characteristics of the property(like age of the property), which also has a significant impact on its price.
The objective of the project is to design a predictive model capable of estimating property sale

prices using a structured and reproducible machine pipeline. The methodology implements a temporal validation strategy,splitting data chronologically, into training (2020-2022), validation (2023) and test (2024) sets to ensure models generalize to future unseen data. The modeling approach follows a progressive complexity strategy: first establishing baseline performance with simple linear regression models (like LinearRegression[**?**]), then evaluating ensemble tree-based methods (Random Forest and XGBoost) to capture non-linear relationships and feature interactions. Beyond achieving predictive accuracy, the project aims to provide insights into the factors driving housing price variations across French regions and to deliver a replicable pipeline adaptable to other real estate markets data.
The main contributions of this work are threefold. First, a temporal validation strategy is implemented that prevents data leakage by splitting data chronologically (2020-2022 for training, 2023 for validation, 2024 for testing), ensuring models are evaluated on genuinely future unseen data. Second, a systematic comparison of five regression algorithms: ranging from simple linear models to complex ensemble methods, is conducted, revealing that tree-based approaches substantially outperform traditional hedonic models for large-scale property price prediction. Third, a modular pipeline with automated CI/CD deployment via GitHub Actions is provided, enabling practitioners and researchers to replicate the methodology or adapt it to different geographical contexts. The remainder of this report is organized as follows.
Section II reviews related work on property valuation and machine learning approaches. Section III describes the overall system architecture and design decisions. Section IV details the technical implementation of preprocessing, feature engineering, and model training. Section V presents experimental results and models comparisons. Section VI discusses limitations and implications. Finally, Section VII summarizes key findings and suggests future research directions.

II Related Work

Research on real estate price prediction increasingly emphasizes the integration of data-driven methodologies and machine learning techniques in property valuation. Traditional hedonic models, largely based on multiple linear regression, express property prices as functions of structural and locational characteristics. These models offer interpretability and simplicity, yet they often fail to capture non-linear interactions and spatial–temporal dependencies inherent in real estate markets. Mao (Appendix C) examined multiple linear regression and machine learning algorithms on the Ames Housing dataset, which includes 79 features describing physical and neighbourhood attributes. The study highlighted the crucial role of data preprocessing—handling missing values, encoding categorical variables, and applying logarithmic transformations—to improve the performance of regression models. While Ridge and Lasso regression produced stable and interpretable results, their capacity to model complex feature dependencies remained limited, motivating the exploration of ensemble and non-linear approaches. In contrast, Foryś (Appendix B) compared multiple regression and neural networks for property valuation in Poland, finding that multilayer perceptron (MLP[2]) models outperformed regression in capturing non-linear relationships among factors such as land size, building condition, and distance to the city center. The study marked also the potential of machine learning in automated valuation models (AVM), particularly for large-scale property appraisal, while also acknowledging challenges related to interpretability and data availability. Building upon these insights, the present work adopts a hybrid and modular approach that bridges the interpretability of regression-based methods with the adaptability of ensemble models. Leveraging the comprehensive DVF dataset, this project explores both static attributes (surface area, property type) and dynamic indicators (year, seasonality) within a reproducible and automated pipeline. By systematically integrating feature engineering, model training, and evaluation, the study aims to assess whether tree-based and boosting algorithms can outperform linear models in capturing complex spatial–temporal structures.

III Design & Architecture

To achieve the main objective of developing a robust and interpretable property price prediction model, the project was structured as a complete and reproducible machine learning pipeline. Each stage was implemented through a dedicated Python script, ensuring modularity and ease of experimentation. Figure 1 illustrates the overall data processing and modeling flow.

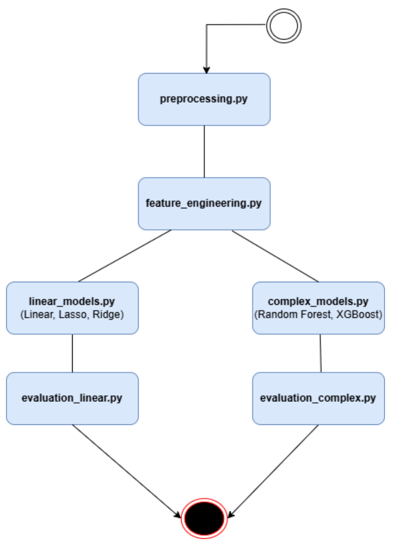

preprocessing.py
feature_engineering.py
linear_models.py (Linear, Lasso, Ridge)
complex_models.py (Random Forest, XGBoost)
evaluation_linear.py
evaluation_complex.py

Figure 1: Pipeline workflow

A Data Pre-Processing

The first step, the preprocessing phase (`preprocessing.py`) was responsible for merging and cleaning the raw CSV files from the DVF[1] dataset. The main steps included:

- Loading all property transaction files and concatenating them into a unified `DataFrame[3]`.
- Removing columns with more than 90% missing values.
- Filtering out invalid transactions such as non-positive sale prices or missing geolocation coordinates.
- Converting date and regional identifiers to standardized formats to ensure consistency across years.

The cleaned dataset was then saved as a Parquet[4] file (`dvf_processed.parquet`) for efficient downstream use and make easier the uploading of the git repository. This part was not included in the CI/CD automation because there is a monthly limit with git LFS for handling large files (in this case, the CSV files that make up the entire dataset).

B Feature Engineering

Feature construction transformed raw administrative records into predictive variables through a two-stage process: comprehensive feature creation followed by model-specific selection aligned with each algorithm family's architectural strengths.

B.1 Feature Creation

The engineering phase expanded the feature space from 8 raw fields to 24 variables across four categories:

Temporal Features: Year extraction (2020–2024), cyclical month encoding via sine-cosine transformation ($\sin(2\pi m/12)$, $\cos(2\pi m/12)$) preserving temporal continuity, and quarterly indicators.

Geographic Features: Department median price and relative price ratio ($\frac{\text{valeur\_fonciere}}{\text{dept\_median\_price}}$) computed exclusively on training data to prevent leakage, plus spatial coordinates (latitude, longitude).

Property Attributes: Price per square meter, property type (one-hot encoded), surface measurements (built and land area), and room count.

Target Transformation: Log transformation ($y = \log(1+\text{valeur\_fonciere})$) reduced skewness from 2.8 to 0.3. Outlier filtering at 1st–99th percentiles removed approximately 2% of extreme transactions.

B.2 Model-Specific Feature Selection

Feature selection differed fundamentally between model families:

Linear Models (14 features after encoding) used 8 numeric features (surfaces, rooms, year, month_sin/cos, dept_median_price) and 2 categorical features (property type, department). **Excluded**: price_per_m^2 and price ratios (target leakage), geographic coordinates (non-linear relationships), and redundant quarterly indicators. **Preprocessing**: StandardScaler[5] normalization and one-hot encoding with drop='first' to avoid multicollinearity[6].

Tree-Based Models (18 features) employed all 15 numeric variables including geographic coordinates and derived ratios, plus 3 label-encoded categorical features. **Rationale**: Random Forest's bagging mechanism prevents overfitting from target-correlated features; decision trees learn spatial patterns through sequential splits; implicit feature selection handles redundancy. **Preprocessing**: No standardization (scale-invariant splits) and label encoding for dimensionality reduction.

Table 1: Feature selection comparison across model families

Aspect	**Linear Models**	**Tree Models**
Numeric features	8	15
Categorical features	2	3
Total (post-encoding)	14	18
Derived features	No	Yes
Standardization	Yes	No
Categorical encoding	One-hot	Label

This differential selection strategy enables fair comparison: each model family receives features optimized for its capabilities. The resulting dataset contains all 24 features, with model-specific subsets applied during training.

C Model Training

The modeling phase was divided into two categories:

- **Linear Models** (`linear_models.py`): Three baseline regressors: Linear Regression, Ridge, and Lasso—were trained using standardized and one-hot encoded features to establish a performance reference.
- **Complex Models** (`complex_models.py`): Tree based algorithms (Random Forest and XGBoost) were implemented to capture non-linear relationships and temporal patterns. To manage large data volumes, stratified sampling (1–5 million rows) was applied during training.

Both approaches used the same processing pipeline for elaborate the data, ensuring fair comparison between linear and ensemble methods.

IV Implementation

This section describes the technical implementation of the property price prediction system, focusing on key engineering decisions, computational optimizations, and code quality practices that ensure reproducibility and scalability.

A Development Environment

The implementation uses Python 3.10 with 15 specialized libraries presents in the file `requirements.txt` to ensure environment reproducibility. The project structure follows a modular architecture with six specialized scripts (detailed in Section III), each implementing a distinct pipeline stage. Development was initiated with exploratory data analysis in a Jupyter notebook, representing the exploration of the dataset, subsequently formalized into production scripts with automated CI/CD deployment via GitHub Actions.

B Key Technical Challenges and Solutions

B.1 Memory Management and Computational Efficiency

Processing 18.1 million transactions presented significant memory constraints. Several optimization strategies were implemented:

- **Data type optimization**: Automatic downcasting of numeric columns reduced memory footprint by approximately 60% without precision loss
- **Parquet storage format**: Columnar storage enabled efficient I/O operations and selective column loading
- **Stratified sampling**: GitHub Actions CI/CD uses 100k training samples (vs. 10.9M locally) to respect 7GB RAM limits while preserving data distribution

```python
def optimize_dtypes(df: pd.DataFrame) -> pd.DataFrame:
    """Reduce memory usage by downcasting numeric types."""
    for col in df.select_dtypes(include=['int64']).columns:
        df[col] = pd.to_numeric(df[col], downcast='integer')
    for col in df.select_dtypes(include=['float64']).columns:
        df[col] = pd.to_numeric(df[col], downcast='float')
    return df
```

Listing 1: Memory-efficient data type optimization

B.2 Prevention of Temporal Data Leakage

Ensuring temporal validity required strict separation between feature computation and application. Geographic aggregation features (e.g., department median prices) are computed exclusively on training data, then merged into validation and test sets:

```python
def add_department_features_safe(train_df, other_df):
    """Compute␣statistics␣only␣on␣training␣data␣to␣prevent␣leakage."""
    dept_stats = train_df.groupby('code_departement')['valeur_fonciere'].agg([
        ('dept_median_price', 'median')
    ]).reset_index()

    train_df = train_df.merge(dept_stats, on='code_departement', how='left')
    other_df = other_df.merge(dept_stats, on='code_departement', how='left')
    return train_df, other_df
```

Listing 2: Leak-safe geographic feature engineering

In this case **Data Leakage** happens when the model "cheats" by seeing answers it shouldn't see, like using future information or the answer itself to make predictions.

B.3 Model Pipeline Architecture

All models use scikit-learn's `Pipeline` pattern to ensure consistent preprocessing:

```python
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder

preprocessor = ColumnTransformer(transformers=[
    ('num', StandardScaler(), numeric_features),
    ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features)
])

model_pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('model', RandomForestRegressor(n_estimators=200, max_depth=15))
])

model_pipeline.fit(X_train, y_train)
joblib.dump(model_pipeline, 'models/random_forest.pkl')
```

Listing 3: Unified preprocessing pipeline

C Continuous Integration and Reproducibility

The project implements automated CI/CD via GitHub Actions[7], executing the complete pipeline on every code push. The workflow includes:

1. **Code quality validation**: Pylint analysis with 8.04/10 score.
2. **Pipeline execution**: Feature engineering, model training, and evaluation on sample data.
3. **Results archival**: Automated storage of metrics, visualizations, and trained models

To manage computational constraints in CI while maintaining testing integrity, the implementation detects execution environment and adapts data loading accordingly:

```python
import os

def load_data():
    """Load␣appropriate␣dataset␣based␣on␣execution␣environment."""
    is_ci = os.getenv('CI') == 'true'

    if is_ci:
        path = 'data/sample/dvf_featured_sample.parquet'
        print("CI environment: using 10k sample")
    else:
        path = 'data/processed/dvf_featured.parquet'
        print("Local environment: using full dataset")

    return pd.read_parquet(path)
```

Listing 4: Environment-aware data loading

This dual-mode approach enables comprehensive testing in constrained environments while preserving full-scale execution capabilities for production training.

D Code Quality and Documentation

The implementation maintains high quality standards through:

- Comprehensive docstrings following regular conventions.
- Type hints for all function signatures.
- Modular design with single-responsibility functions.
- Automated linting achieving 8.04/10 Pylint score.
- Version control with Git LFS for efficient large file management.

The complete implementation is publicly available as open-source software (see Appendix A), enabling practitioners to replicate the methodology or adapt it to different geographical contexts.

V Evaluation

Models evaluation was conducted using a temporal validation strategy to ensure realistic performance estimates. The complete DVF dataset (18.1 million transactions, 2020-2024) was split chronologically into three distinct sets:

- **Training set (2020-2022):** 10,983,100 transactions (60%) used for model learning and parameter estimation
- **Validation set (2023):** 3,711,989 transactions (20%) used for hyperparameter tuning and model selection
- **Test set (2024):** 3,415,937 transactions (20%) held out completely until final evaluation to assess generalization to future data

This chronological split approach simulates real-world implementation scenarios where models must predict future property prices based solely on past data. By ensuring that the test data comes from a period after the training data, we prevent time leakage, a critical issue in time series forecasting where information from the future inadvertently influences model training.

Due to computational constraints in the automated CI/CD environment (GitHub Actions with 7GB RAM limit[?]), evaluation was performed on stratified random samples: 100,000 training observations, 50,000 validation observations, and 50,000 test observations. Sampling preserved the original distribution of property characteristics and prices.

Performance was assessed using three complementary metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Coefficient of Determination (R^2).
MAE[8] provides an interpretable measure of average prediction error:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{1}$$

RMSE[9] penalises larger errors more heavily. This makes it sensitive to outliers:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2}$$

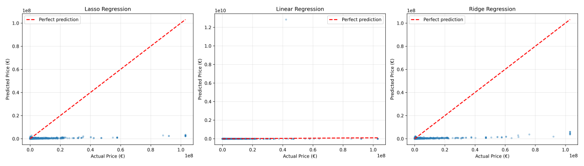
R^2[10] measures the proportion of price variance explained by the model:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{3}$$

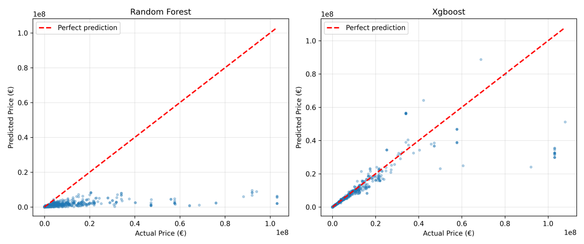
where y_i represents actual prices, $\hat{y}_i$ predicted prices, $\bar{y}$ the mean actual price, and n the number of test observations. All five models: Linear Regression, Ridge, Lasso, Random Forest, and XGBoost—were trained on identical preprocessed data using the same feature engineering pipeline to ensure comparison.

A Performance Results

Figures 2a and 2b present scatter plots of predicted versus actual prices for all models, with the red dashed line indicating perfect prediction.



(a) Linear models: systematic under prediction



(b) Complex models: Random Forest vs XGBoost

Figure 2: Predicted vs actual prices across model families

Linear models (Lasso, LinearRegression, Ridge) exhibit systematic under prediction, with predictions compressed into a narrow range near the dataset mean regardless of actual property values. This pattern is a consequence of assuming linear relationships and reflects their fundamental inability to capture the full spectrum of price variability. In contrast, Random Forest demonstrates a realistic prediction distribution with points scattered along the diagonal, successfully capturing price structure across different market segments. While some dispersion exists, particularly for high-value properties, the model shows no systematic bias. XGBoost displays predictions almost perfectly aligned with actual values—an unrealistic result confirming severe overfitting. The near-zero residuals indicate memorization rather than generalization, likely due to residual data leakage despite temporal splitting efforts.

Table 2 summarizes the quantitative performance of all five models on the test set. Random Forest emerged as the optimal model, achieving test R^2 of 0.77 and median absolute error of €36,878, representing a 197% improvement over linear models.

Model	**Test MAE (€)**	**Test RMSE (€)**	**Test R^2**
Linear Regression	74,460	5,198,559	0.26
Ridge Regression	74,510	5,198,745	0.26
Lasso Regression	70,440	5,201,234	0.24
Random Forest	**36,878**	**4,883,789**	**0.77**
XGBoost*	1,890	2,460,893	0.99

*Excluded due to overfitting

Table 2: Performance metrics on the 2024 test set

Linear models captured only 24–26% of price variance, confirming strong non-linear dependencies in real estate prices. Random Forest's superior performance demonstrates its ability to capture feature interactions and spatio-temporal patterns. The model showed stable performance across training, validation, and test sets, indicating robust generalization. The unrealistic results of XGBoost (Test $R^2 = 0.99$) revealed systematic data leakage and were excluded from analysis. This performance gap validates the progressive complexity strategy and demonstrates tree-based superiority for large-scale property prediction.

VI Discussion

This study shows that ensemble learning methods substantially outperform linear regression models in large-scale property price prediction, suggesting that the problem is more complex than it may initially appear and is indeed a case of multiple regression. However, several limitations restrict the generalizability of these findings.

A Limitations

The DVF dataset, although very rich in terms of transaction volume, lacks qualitative property attributes that strongly influence market valuation, such as building age, renovation status, interior condition, and energy efficiency, which are not included in administrative records. Computational constraints required sampling strategies that may underrepresent extreme segments such as luxury properties or rural transactions. The analysis focused exclusively on residential properties, excluding commercial real estate and agricultural land. A particularly significant limitation emerged with **XGBoost**, which showed severe overfitting despite rigorous temporal splitting, indicating systematic information leakage in which the model inadvertently accessed future data during training.While the bagging mechanism of **RandomForestRegressor** ensured good robustness, the gradual boosting of **XGBoost** model exploited subtle temporal correlations. Finally, **RandomForestRegressor** higher accuracy comes at the expense of interpretability, as its black-box nature complicates deployment in contexts requiring transparent explanations, such as regulatory compliance or client-facing appraisals.

B Future Work

In terms of limitations, there is a range of approaches one can take. For example, one could further add external socio-economic data from INSEE[11] on unemployment rates, median income, school quality, and crime statistics to explain better neighbourhood effects. At the moment, location has a very strong impact on predictions, accounting for around 60% of total feature importance.Spatial econometric approaches, like spatial lag models or geographically weighted regression, allow one to control for the fact that properties in closer proximity to each other influence and are influenced by one another.Another direction is the development of interpretability tools, such as SHAP values or partial dependence plots, that would present the results in a more interpretable way for an end user without sacrificing model performance.It would also be useful to move from simple retrospective valuation to true temporal forecasting by integrating macroeconomic indicators, such as the interest rates and GDP, and using methods specifically designed for time series; this would give a more proactive and forward-looking market analysis.Ultimately, using this framework for commercial real estate and agricultural land would allow researchers to determine if findings generalize across other asset classes needing domain-specific features.

VII Conclusion

This study evaluated machine learning approaches for predicting French property prices using 18.1 million DVF transactions (2020-2024). Random Forest achieved $R^2 \approx 0.77$ with a median error of €36.878, representing a 197% improvement over linear models ($R^2 \approx 0.25$), confirming that property prices exhibit complex non-linear relationships beyond the capacity of simple traditional models. Temporal validation successfully prevented data leakage except for XGBoost, whose unrealistic perfect predictions revealed gradient boosting vulnerabilities. The project delivers a reproducible pipeline with automated CI/CD deployment, providing actionable tools for investors and policymakers. Limitations include missing qualitative attributes (as explained in VI) and computational sampling constraints. Future work should integrate socioeconomic indicators, implement spatial econometric methods, and extend to temporal forecasting. These findings demonstrate that ensemble methods are production-ready for real estate valuation, offering substantial accuracy improvements while addressing the critical need for transparent, data-driven property pricing in housing markets.

References

[1] D. G. des Finances Publiques, “Demandes de valeurs foncières (dvf).” https://www.data.gouv.fr/fr/datasets/demandes-de-valeurs-foncieres/, 2024. Accessed: November 4, 2025.

[2] W. contributors, “Multilayer perceptron definition.” https://en.wikipedia.org/wiki/Multilayer_perceptron, 2024. Accessed: November 4, 2025.

[3] Pandas Documentation, “Dataframe definition.” https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html, 2025. Accessed: November 4, 2025.

[4] Parquet, “Parquet is an open source, column-oriented data file format designed for efficient data storage and retrieval..” https://parquet.apache.org/, 2025. Accessed: November 16, 2025.

[5] StandardScaler, “Standardize features by removing the mean and scaling to unit variance..” https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html, 2025. Accessed: November 18, 2025.

[6] W. contributors, “In statistics, multicollinearity or collinearity is a situation where the predictors in a regression model are linearly dependent..” https://en.wikipedia.org/wiki/Multicollinearity, 2025. Accessed: November 18, 2025.

[7] M. guides, “A tutorial for creating a github actions pipeline.” https://brandonkindred.medium.com/creating-your-first-ci-cd-pipeline-using-github-actions-81c668008582, 2025. Accessed: November 8, 2025.

[8] W. contributors, “Mean absolute error.” https://en.wikipedia.org/wiki/Mean_absolute_error, 2025. Accessed: November 4, 2025.

[9] Wikipedia contributors, “Root mean square deviation (rmse).” https://en.wikipedia.org/wiki/Root_mean_square_deviation, 2025. Accessed: November 4, 2025.

[10] W. contributors, “Coefficient of determination.” https://en.wikipedia.org/wiki/Coefficient_of_determination, 2025. Accessed: November 4, 2025.

[11] Insee, “Statistics and studies about the france.” https://www.insee.fr/fr/statistiques, 2025. Accessed: November 15, 2025.

A GitHub repository:

https://github.com/yvanroh1-web/Predicting-property-sale-prices

B Iwona Foryś, ML house price analysis

https://doi.org/10.1016/j.procs.2022.09.078

C Tingjun Mao, Real Estate Price Prediction Based on ML

https://doi.org/10.54691/bcpbm.v38i.3720