House Price Prediction Technical Task Report

Technical Task Submission

Submitted by: Hilal Ilcin
Date: 4 January 2025

Objective

To develop a robust predictive model for real estate prices ("House price of unit area") using machine learning techniques. The task involves data preprocessing, exploratory data analysis, training various models, hyperparameter optimization, and performance evaluation.

Exploratory Data Analysis (EDA) and Model Development Report

1. Data Overview and Initial Inspection

The dataset consists of 414 rows and 9 columns. Initial inspection using df.describe() provided summary statistics for the numerical columns, which helped in understanding the data's central tendencies and variability.

2. Missing Value Analysis

The dataset was checked for missing values using df.isnull().sum(). No missing data was detected, ensuring the completeness of the dataset.

3. Time Data Transformation

The **Transaction date** column was split into **Year** and **Fractional Year** to better understand time-related effects. A new **formatted date** column was created for improved time analysis. The transformation involved:

```
df['Year'] = df['Transaction date'].astype(int)

df['Fractional Year'] = df['Transaction date'] - df['Year']

df['Days'] = (df['Fractional Year'] * 365).round().astype(int)

df['Transaction date (formatted)'] = pd.to_datetime(df['Year'], format='%Y') + pd.to_timedelta(df['Days'],
unit='d')
```

4. Feature Engineering

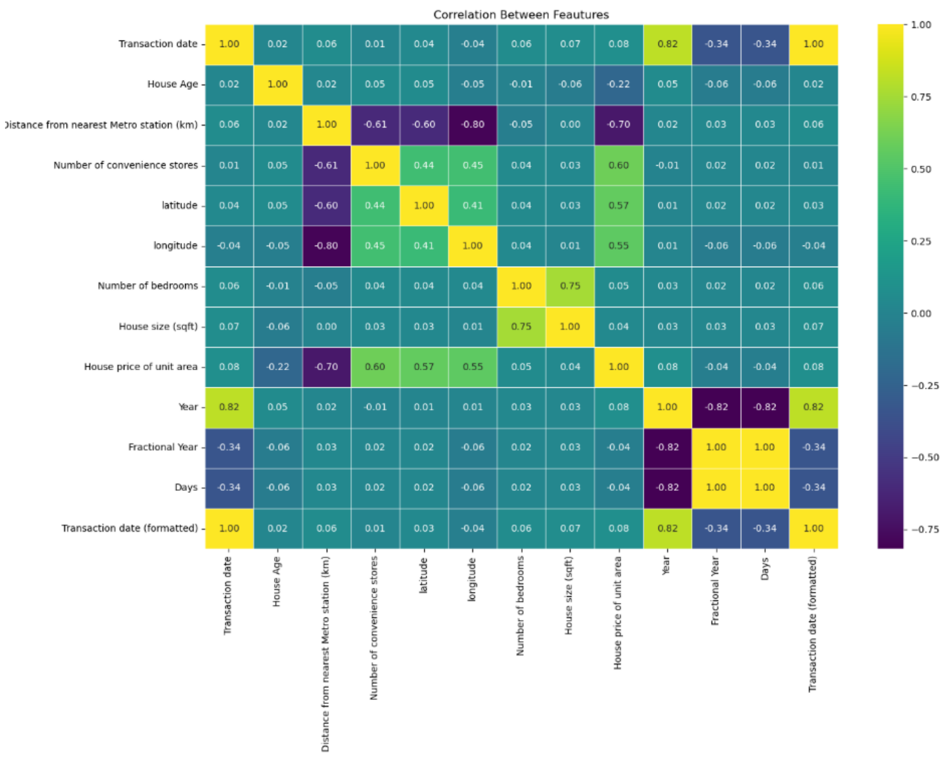
New features were introduced to capture the relationships between location and distance:

- **Distance * Latitude**
- **Distance * Longitude**

These transformations aimed to enhance the model's ability to understand spatial patterns.

5. **Correlation Matrix Visualization**

At this point, before you move to feature selection, it's helpful to visualize the correlation between features. This helps identify any strong relationships between the features and the target variable. It can also guide the feature selection process by highlighting highly correlated features that may be redundant.

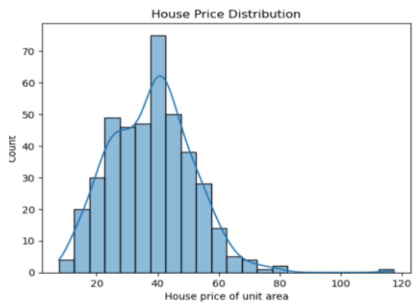


6. Target Variable Analysis

A histogram was plotted to examine the distribution of the target variable, **House price of unit area**. This provided insight into its spread and any potential skewness

7. Variable Relationships

Pairwise relationships were analyzed using **sns.pairplot(df)**. This visualization helped identify potential correlations between variables, informing feature selection decisions.



8. Categorical Variable Analysis

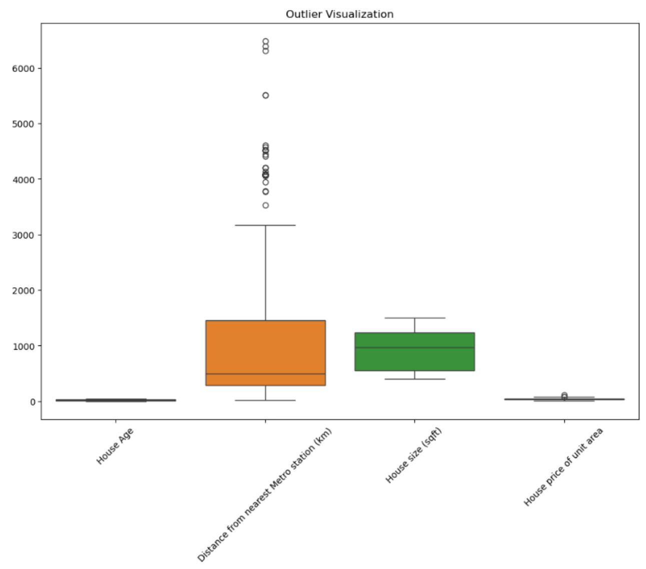
For categorical features such as **Number of convenience stores** and **Number of bedrooms**, we examined the unique values to understand their distributions and structure.

9. Data Preprocessing

- **One-Hot Encoding**: Applied to categorical variables like **Number of convenience stores** and **Number of bedrooms** to transform them into numeric form.
- **Standardization**: Continuous variables like **House Age**, **Distance from nearest Metro station (km)**, **House size (sqft)**, and **House price of unit area** were standardized to ensure they were on the same scale.

10. Outlier Detection and Handling

Outliers were detected using the **Z-score method** with a threshold of 3, and **boxplots** were used for visualization. Outliers were removed to ensure model accuracy and avoid distortion.



Model Building and Evaluation

1. Feature Selection and Correlation

- A correlation matrix was created to assess the relationship between features and the target variable, **House price of unit area**.
- Features with low correlation (below 0.1) were dropped, such as **Transaction date**, **Number of bedrooms**, **House size (sqft)**, and others.

2. Data Preprocessing for Model Building

- **Standardization** was applied to the remaining continuous variables.
- The dataset was split into training (80%) and testing (20%) sets using **train_test_split** to allow for model evaluation on unseen data.

3. Model Training and Evaluation

Several models were trained and evaluated:

- **Linear Regression**: As a baseline, it achieved an **R^2 = 0.68**, **MAE = 5.27**, and **MSE = 53.29**.
- **Random Forest Regressor**: Showed better performance with an **R^2 = 0.80**, **MAE = 4.17**, and **MSE = 33.01**.
- **XGBoost Regressor**: Achieved **R^2 = 0.72**, **MAE = 4.83**, and **MSE = 47.64**.
- **Support Vector Regressor (SVR)**: **R^2 = 0.69**, **MAE = 5.13**, and **MSE = 52.71**.

4. Hyperparameter Tuning

- **Grid Search** was employed to optimize **Random Forest** and **XGBoost** hyperparameters.
- **Random Forest**: Optimal parameters were **max_depth=30**, **min_samples_split=5**, and **n_estimators=300**.
- **XGBoost**: Optimal parameters were **learning_rate=0.01**, **max_depth=3**, **n_estimators=300**, and **subsample=0.7**.

After fine-tuning, the models showed improvement:

- **Optimized Random Forest**: **R^2 = 0.81**, **MAE = 4.03**, **MSE = 31.52**

- **Optimized XGBoost**: **R^2 = 0.81**, **MAE = 4.05**, **MSE = 32.39**

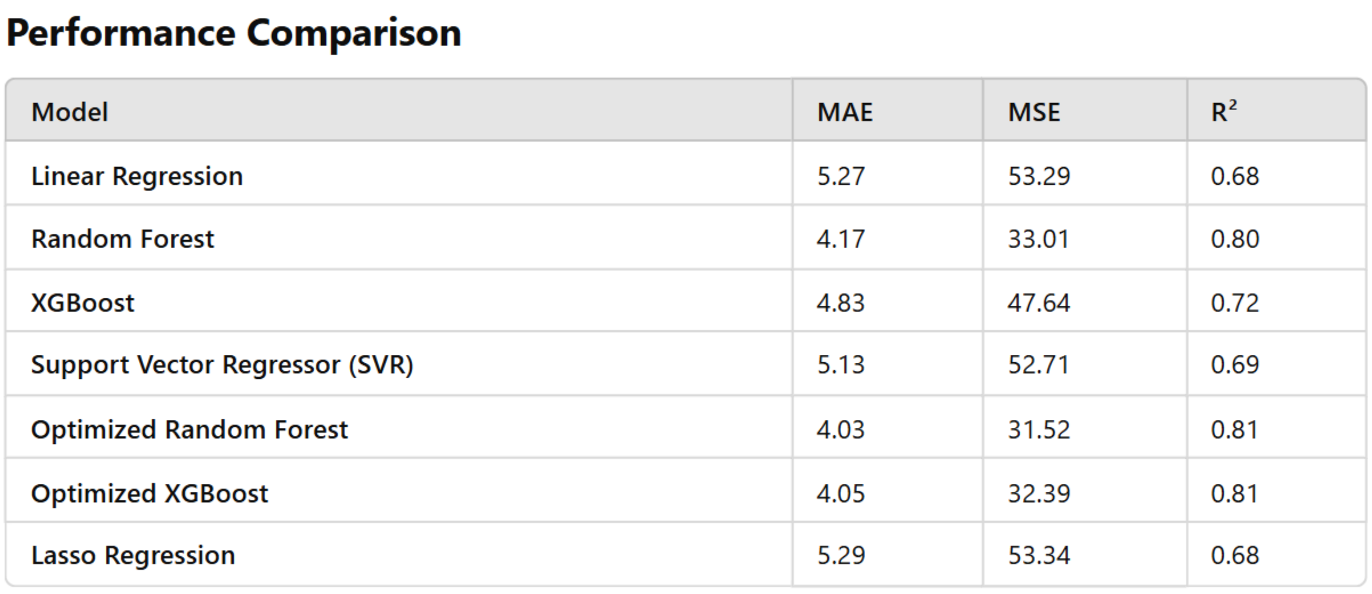
5. Cross-Validation

- **Random Forest** and **XGBoost** models were cross-validated, achieving consistent results across different data splits:
    - **Random Forest R^2 = 0.74**
    - **XGBoost R^2 = 0.75**

6. Lasso Regression

Lasso Regression, used for regularization and feature selection, achieved:

- **R^2 = 0.68**, **MAE = 5.29**, and **MSE = 53.34**, but did not outperform the tree-based models.



Performance Comparison

Model	MAE	MSE	R^2
Linear Regression	5.27	53.29	0.68
Random Forest	4.17	33.01	0.80
XGBoost	4.83	47.64	0.72
Support Vector Regressor (SVR)	5.13	52.71	0.69
Optimized Random Forest	4.03	31.52	0.81
Optimized XGBoost	4.05	32.39	0.81
Lasso Regression	5.29	53.34	0.68

Conclusion

- **Random Forest** and **XGBoost** are the best-performing models, with both achieving an **R^2** of 0.81 after optimization.
- **Random Forest** was slightly superior, with a lower **MAE** (4.03) and **MSE** (31.52).
- **Lasso Regression** and **SVR** provided valid alternatives but did not perform as well as the ensemble models.
- **Cross-validation results** confirm the robustness and generalizability of both **Random Forest** and **XGBoost**, indicating that these models are reliable for predicting house prices.