Predicting House Prices Using Machine Learning Algorithms and PandasAI

Anuj Bhandari
Longo Faculty of Business
Humber College
Lakeshore Campus
N01660244@humber.ca

Apurva Naringrekar
Longo Faculty of Business
Humber College
Lakeshore Campus
N01649220@humber.ca

Kathy Anusha Felix
Longo Faculty of Business
Humber College
Lakeshore Campus
N01674277@humber.ca

Manpreet Kaur
Longo Faculty of Business
Humber College
Lakeshore Campus
N01705619@humber.ca

Olga Ornek
Longo Faculty of Business
Humber College
Lakeshore Campus
N01644945@humber.ca

Roja Balarathinam
Longo Faculty of Business
Humber College
Lakeshore Campus
N01673008@humber.ca

Abstract

The accurate prediction of housing prices is a significant issue for buyers, sellers and investors in real estate. A good model can make sure people can be smarter about their decisions, reduce risk and increase confidence in the market. In this work, we implemented and compared some popular algorithms such as Linear Regression, Random Forest and K-Nearest Neighbors on the Kaggle housing dataset. We also talked about a data exploration and preprocessing helper, PandasAI, which is an AI assistant. Finally, to verify the model the ANN model is being tested with the five popular error metrics ME (Mean Error), RMSE (Root Mean Squared Error), MAE (Mean Absolute Error), MPE (Mean Percentage Error) & MAPE (Mean Absolute Percentage Error). The model was best (among our tested models) for a Random Forest model. This project showcases the implementation of ML and AI tools to estimate the correct price of real-estate property.

Keywords—Machine Learning, House Price Prediction, Pandas AI, Regression, Data Analytics, Feature Selection, Linear Regression, Random Forest and K-Nearest Neighbors (KNN, Decision Tree (PandasAI).

I. Introduction

The price of a home is largely affected by factors such as location, amenities within the property, and the building's facilities. As a result, efficient and accurate prediction of house prices can be a challenging problem. But it's very, very useful to be able to do so. For shoppers, sellers, developers, and investors, getting a good estimate of the value of property is useful in planning, negotiations, and investment decisions.

Machine learning is increasingly being used as a tool to tackle such issues. If enough of a historical record can be found, these models can detect behaviours that are recurring and anticipate which ones will recur in the immediate future. In this project, we employ linear regression, Random Forest and K-Nearest Neighbours for predicting the price of a house from some external attributes such as the age of the house, the distance to the nearest MRT station, the number of convenience stores that are nearby or the geographical coordinates. PandasAI is also a tool that we utilise for assistance in the cleaning and analysis of the dataset.

To evaluate our models fairly, we use metrics like: ME, RMSE, MAE, MPE and MAPE. We want to see how tools like PandasAI can be used for actual machine learning work.

II. Dataset Overview

The dataset is taken from Kaggle that contains the historical records of the houses sold in a metropolitan city. Each row in the data is information about a house transaction as well as some of that transaction's features which might influence the price of the house. These include the physical characteristics of the home, location-based features and amenities.

The dataset consists of 414 observations and seven variables. The most important feature predicted is the cost per unit area, which we use as response variable for all models.

Here is a quick summary of the dataset fields:

TABLE 1: Dataset Features and Description

Feature	Description
transaction_id	Unique identifier for each record
x1_transaction_date	Date of the house transaction
x2_house_age	Age of the property in years
x3_distance_to_the_nearest_mrt_station	Distance to nearest MRT station (meters)
x4_number_of_convenience_stores	Number of convenience stores within proximity
x5_latitude	Latitude coordinate of the property
x6_longitude	Longitude coordinate of the property
y_house_price_of_unit_area	Target variable: house price per unit area in dollars

III. Literature Review

For this project, we explored different machine learning models that performed well in predicting house prices. We went through the plethora of research papers, Kaggle notebooks, technical blogs, and case studies to see what works best for real estate prediction. Although several algorithms are possible, most references described only a few that have been found to be useful: Linear Regression, Random Forest, and K-Nearest Neighbours (KNN).

One of the most straightforward models suitable for price forecasting tasks is Linear Regression. It's easy to use, easy to understand, and a useful visualization of how each variable makes a difference in the end price point. However, multiple studies have indicated this approach lacks in the face of outliers and non-linear relationships between variables (Jha et al., 2020; Das et al., 2020).

We encountered Random Forest many times when analysing top models. Because it takes the average of predictions of many decision trees, it is more robust and less likely to overfit. Many studies indicated that Random Forest performed well with noisy housing data and was better than simpler models such as linear regression (Adetunji, et al., 2022; Mirbagherijam, 2021).

The K-Nearest Neighbours (KNN) was also presented in a few case studies. It operates on the principle that similar homes will generally have similar prices — a good assumption especially in real estate, where location is a big factor. However, KNN can be very sensitive the choice of k and needs feature scaling or may yield twisted results (Nivitha Shree et al., 2022; Intel AI Blog, 2022).

Beyond those conventional options, we found PandasAI, an AI-tool offering more traditional Exploratory Data Analysis and basic modelling through natural language. Although its use is not meant to replace model building manually, it is a useful aid in the process of data cleaning, summarizing, and plotting, especially at the beginning of an analysis (PandasAI, 2024).

In summary, we chose these models over those already in the literature and in community projects for their successful replication history, fit to the data we are using, and ease of communication.

IV. Methodology

This section presents a combined overview of data preprocessing, feature selection, modeling, and performance evaluation. In the interest of keeping a consistent process for all manual models, we used a common workflow which involved data cleaning, exploration, model training and performance evaluation. All steps were process optimized to guarantee comparability between the methods.

4.1 Workflow Description

This project followed a structured, consistent, and replicable process across all four models: Linear Regression, Random Forest, K-Nearest Neighbors, and the PandasAI-powered Decision Tree. The complete pipeline consisted of five key stages:

Data Cleaning and Preparation: Basic formatting, null/duplicate checks, outlier handling, and feature selection were done uniformly.

Exploratory Data Analysis: Important features were visualized and their relationships examined to guide future modelling steps.

Model training: All models were trained with 60:40 train-validation split to compare with each other.

Assessment: Model predictions were evaluated with standard error metrics and visualizations.

Configuration Management: The approach was similar for PandasAI, but the steps took place interactively using prompts, rather than manual coding.

The consistency of this pipeline allowed for fair comparison of performance, respecting the individual needs of each model (e.g., scaling for KNN or prompt-driven commands for PandasAI).

4.2 Data Cleaning & Preparation

To prepare the dataset for modeling the following preprocessing steps were executed consistently across both manual and AI assisted approaches:

Column Formatting and Renaming

- Whitespace in column names was replaced with underscores (_).
- All column names were converted to lowercase.
- The no column was renamed to transaction_id to provide semantic clarity and improve interpretability.

Missing Values

- A check for missing values was conducted across all columns.
- Result: No missing values were found in the dataset.

Duplicate Records

- The dataset was checked for duplicate rows using .duplicated().sum().
- Result: No duplicate records were present.

Data Types Validation

- Each column's data type was verified to ensure compatibility for numerical analysis and modeling.
- Result: All datatypes were appropriate and required no changes.

Outlier Detection and Removal

- Descriptive statistics was first calculated in. describe() to know the mean and variance of the variables.
- The shape of y_house_price_of_unit_area looked right skewed in the boxplot and there were some outliers separated from the upper whisker (~117.5). These have been manually removed after removal of 3 of the most extreme values 78.3, 78 and 117.5, leaving the dataset with 411 rows.

- Latitude and longitude also had some unusually high or low values, but since they represent actual geographic coordinates, we chose to leave them unchanged to preserve spatial integrity.

- Percentile capping was implemented for the variable x3_distance_to_the_nearest_MRT_station. Values above the 97th percentile (~2890.0) were replaced with the percentile threshold to limit the influence of distant properties.

- Note that the feature x4_number_of_convenience_stores had few high values. But they weren't too extreme or wrong, so we did not change anything.

- In the case of x2_house_age, no clear outliers were observed based on boxplot or statistical summaries, so no changes were made.

  The multi-aspect outlier treatment made sure that the variables were processed based on their distribution and significance, enhancing model stability without excessive distortion of realistic data patterns.

Irrelevant Column Removal

- The transaction_id column was dropped as it was simply a unique identifier with no predictive value.

- The x1_transaction_date column was excluded due to inconsistencies and limited variability. Instead, x2_house_age was retained as it provides a more stable temporal predictor.

Normalization (Applied Only for KNN Model)

- K-Nearest Neighbors is one of the distance-based models and scale-sensitive, hence works on inaccurate features for sure.

- Therefore, before training the KNN model, StandardScaler from sklearn.preprocessing was applied to rescale all input features to have zero mean and unit variance.

- This normalization step was only made for KNN because the other two models utilized (Linear Regression and Random Forest) are not influenced by feature scaling.

Final Columns Retained for Modeling

After cleaning, the following features were retained:

- x2_house_age
- x3_distance_to_the_nearest_mrt_station
- x4_number_of_convenience_stores
- x5_latitude
- x6_longitude
- y_house_price_of_unit_area (Target)

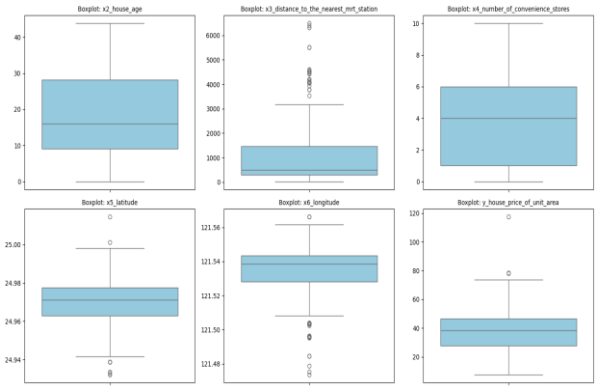

Fig.1. Boxplots of all numerical variables.

These plots reveal the distribution and presence of outliers in key features.

4.3 Exploratory Data Analysis (EDA)

Before modelling the data, we performed exploratory data analysis (EDA) to understand the general profile of each variable and connection with the price of the house. These intuitions guided the model design and were shared among all our models such as the PandasAI model (where the whole process has been conducted by means of natural language).

Distribution of Features

The descriptive statistics, histograms, and boxplots were used to check distribution of all numerical variables. These visualization techniques helped in getting a sense of how data is distributed and see where most observations fell and whether the distributions were symmetrical or skewed.

- The response, house price of unit area, had a long right skew and most of prices clustered between 20 to 50 and a few high-priced ones pushed the distribution towards the right. As can be seen in the boxplot, the lower 25% of the pair counts fell below approximately 27.5 and the upper 25% fell above around 46.3, with a mean of ~37.6. The highest observed value before topping was 117.5, showing a long tail to the left in prices.

- The distance from the nearest MRT station was another variable that revealed right-skewed distribution as most of them were situated at distance from housing to MRT between the ranges, but some were located at the longer distances.

- Distribution of house age was similar, with homes in all age groups.

- Number of convenience stores had a left-tailed distribution. The number of stores ranged from 0 to 3, and only a small number had more stores.

This distributional analysis informed our intuition about the min-max range, skewness and central tendency of the important features in a later model formulation.

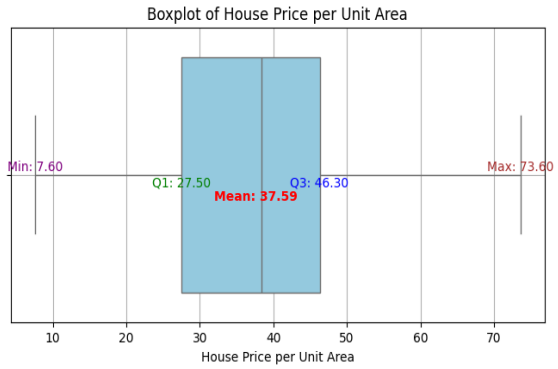

Fig.2. Boxplot of House Price per Unit Area

Correlation Analysis

We did this analysis by a correlation matrix which shows the strength of each feature correlation to the target variable. This has allowed us to better tell which predictors have more effect over house price.

- Accessorily, the distance to the nearest MRT station is strongly negatively correlated with price, indicating that houses can value more if they are near to an MRT station.
- There was a negative moderate correlation with age of house, whereby newer houses are relatively more expensive.
- Number of convenience stores showed a moderate positive correlation, suggests that houses located at places with more amenities nearby are likely to have higher prices.
- Latitude and longitude obtained weaker correlations but were found to be useful in the models as they were able to account for spatial effects not detectable with simple correlation coefficients.

The results of these were used as a guide in the a priori selection of variables in model building and served to make more evidence-based choices in later phases.

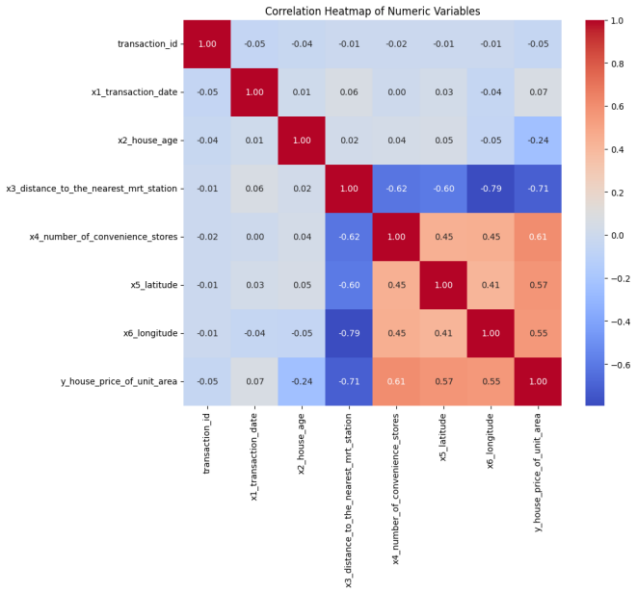

Fig.3. Correlation heatmap

This figure displays the pairwise correlation coefficients. The brighter regions indicate stronger relationships with house prices.

4.4 Modeling and Evaluation

Following EDA, features were separated from the target, the data was divided into training and validation (60/40) sets. Various models required various preprocessing (for instance, KNN and its sensitivity to feature scale). They have also used different variable selection approaches in each model. And predictivity of the used models were the mentioned measurement criteria (ME, RMSE, MAE, MPE and MAPE).

4.5 Pseudocode: House Price Prediction Pipeline

Step 1: Import required Python libraries

pandas, matplotlib, seaborn, sklearn, etc.

Step 2: Load the dataset and inspect its structure:

Use df.info() and df.head() to examine data types and columns.

Step 3: Perform data cleaning and preprocessing

- Rename all column names to lowercase with underscores
- Remove duplicate rows if any
- Confirm there are no missing values
- Drop unnecessary columns like transaction_id and transaction_date
- Handle outliers:
    - Cap values above 97th percentile in distance_to_the_nearest_MRT_station
    - Remove 3 extreme records in house_price_of_unit_area
    - Retain latitude and longitude as they represent real coordinates

Step 4: Perform Exploratory Data Analysis (EDA)

- Plot histograms and boxplots to inspect feature distributions
- Create a correlation heatmap to assess feature relationships
- Use scatter plots to explore interactions between predictors and target

Step 5: Split the dataset into features X and target y

- X = df.drop("y_house_price_of_unit_area", axis=1)
- y = df["y_house_price_of_unit_area"]

Step 6: Apply standard scaling (only for KNN model)

Use StandardScaler to normalize numerical features if KNN is selected

Step 7: Perform variable selection based on the model

- Linear Regression: Backward Elimination
- Random Forest: Feature Importance
- KNN: GridSearchCV for best k

Step 8: Split data into training and validation sets (60:40 ratio)

Step 9: Train the selected model on training data

- Linear Regression using LinearRegression()
- Random Forest Regressor using RandomForestRegressor()
- K-Nearest Neighbors (KNN) using KNeighborsRegressor()
- Decision Tree (via PandasAI) through AI prompt-based interaction

Step 10: Generate predictions on both training and validation sets

Step 11: Evaluate model performance using regression metrics:

- Mean Error (ME)
- Root Mean Squared Error (RMSE)
- Mean Absolute Error (MAE)
- Mean Percentage Error (MPE)
- Mean Absolute Percentage Error (MAPE)

Step 12: Visualization of predictions

- **Scatter Plots**: Actual vs Predicted plots are made on validation to check model calibration. Points that are closer to this diagonal line suggest a better prediction.
- **Histograms of Residuals**: For each model, generate histograms of residuals (difference between actual and predicted values). Such plots can be used to illustrate the distribution of the prediction errors and to detect the presence of bias or skewness.
- **Error Metrics**:
  - Mean Error (ME)
  - Root Mean Squared Error (RMSE)
  - Mean Absolute Error (MAE)
  - Mean Percentage Error (MPE)
  - Mean Absolute Percentage Error (MAPE)

This step provides both visual and statistical evidence of how each model performed and helps identify areas where predictions may be off.

V. MANUAL MODELS

5.1 Linear Regression

Variable Selection

For the Linear Regression model, we used backward elimination to select the most relevant variables. Starting with all available features, we gradually removed those that did not contribute significantly to the model based on statistical outputs. This helped reduce unnecessary complexity and retain only the predictors that had the most influence on house prices.

Model Training

The Linear Regression model is trained on 60% of the data, the training set. Following fitting, the remaining 40% of the data were used for the model to test how well it could generalise to unseen cases. The fact that in this model linear relationships between the inputs and the output are assumed, means that neither transformation nor interaction terms were introduced at this point.

Model Prediction

We then applied the trained Linear Regression model to make price predictions on the training and validation sets. The aim was to evaluate how well the model can generalize to the unknown data. Subsequently, the predicted drop volumes were compared with the real drop volumes for two data sets. Although the model did conform to the overall pricing trend, there were some discrepancies present there within a complex setting as we show further in performance.

Model Performance

We evaluated the effectiveness of Linear Regression using visual aids and statistics.

Most of the scatter plot of actual vs. predicted house prices lies on the diagonal, which would correspond to perfect predictions. For the more expensive houses however, the predictions were not only less accurate, but they also showed to be more fuzzy, showing the model struggled to grasp a more complex or non-linear relationship between the pricing and the input features.

We also formed histograms of residuals (actual - predicted prices) and found a not very large non-normality. This in addition to the above supported that, though the model caught general trends, there was still structure in the data it just could not explain.

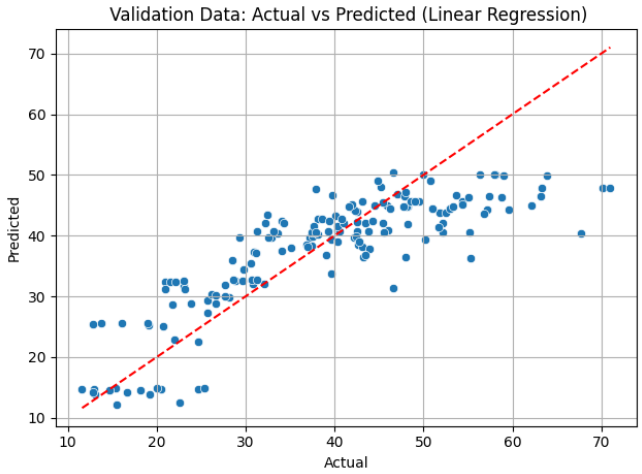

Fig.4. Actual vs Predicted Prices - Linear Regression

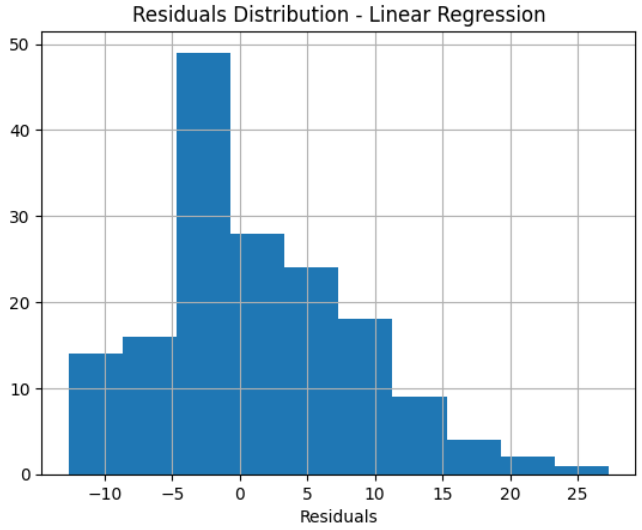

Fig.5. Residuals Distribution - Linear Regression

Finally, we evaluated the model using five standard regression metrics:

METRICS	VALUE
ME ($)	1.2256
RMSE ($)	7.6140
MAE ($)	5.8800
MPE (%)	-1.4676
MAPE (%)	16.5966

Those values indicate the model performed moderately, where the RMSE and MAPE are quite high meaning the predicted price can differ significantly especially in higher home's value.

5.2 Random Forest Regressor

Variable Selection

We employed the in-built feature importance method of the Random Forest model. This method prioritizes variables by their degree of impurity reduction between trees. Using these rankings, we singled out the features with the highest predictive capabilities. The significant variables were selected to contribute to the final modeling, which helped to enhance prediction performance and control overfitting.

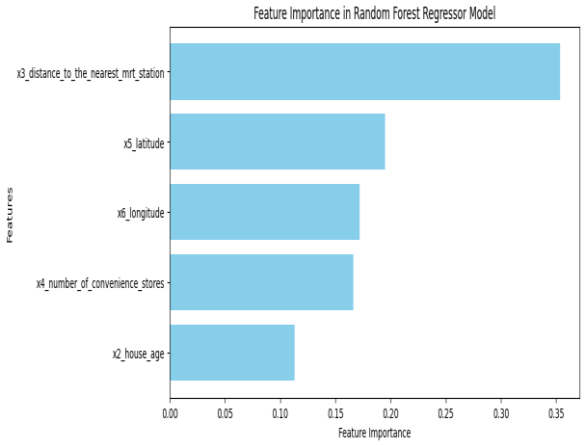

Fig.6. Feature Importance - Random Forest

Model Training

The Random Forest Regressor was fitted with the identical learning set (60% of samples). The model creates a group of decision trees, each of which is trained on a random fraction of the data. The final prediction is computed by taking the average of all the tree predictions and this can mitigate the variance and enhance robustness.

Model Prediction

Upon training the Random Forest model by training data, predictions were performed on the training and validation datasets. The ensemble aspect of Random Forest allowed it to pick up these patterns effectively and manage interactions between features nicely. With both seen and unseen data, prediction values shadowed the actual across the price spectrum and had solid agreement.

Model Performance

The predictions of the model were presented on a scatter plot comparing prices that were predicted and the prices that actual were in the validation. Random Forest's predicted values were closer to actual values than Linear Regression's in mid and much more in mid than in very low-midrange prices. While there was still some spread in the high value estimates, the model seemed to do much better at addressing non-linear relationships.

A histogram of the residuals became less skewed with less extreme values, indicating that the model was able to explain the patterns in the pricing data better.

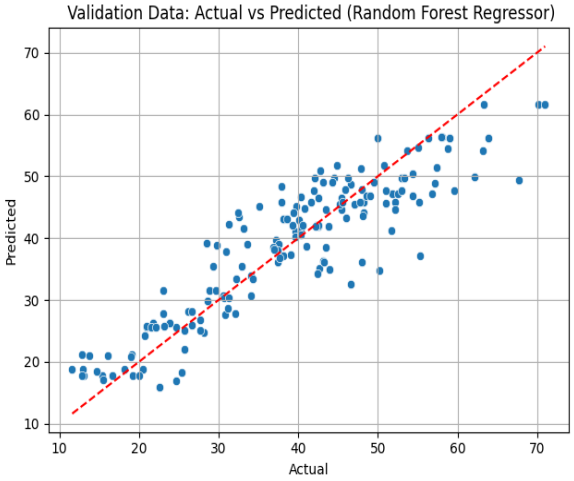

Fig.7. Actual vs Predicted Prices - Random Forest

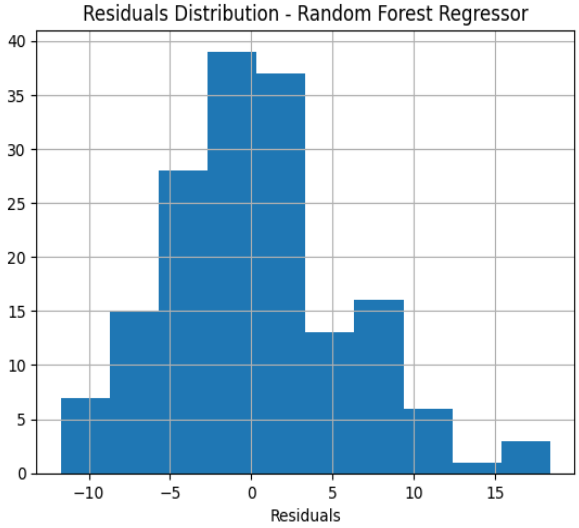

Fig.8. Residuals Distribution - Random Forest

Here are the regression metrics for Random Forest:

METRICS	VALUE
ME ($)	0.2008
RMSE ($)	5.7360
MAE ($)	4.4204
MPE (%)	-2.8545
MAPE (%)	12.9027

The lower the RMSE and MAPE are, the more consistent and accurate the performance of the model and that is what the Random Forest achieved particularly for properties with mid-to-high price.

5.3 K-Nearest Neighbors (KNN) Regressor

Variable Selection

For the KNN model, variable selection was carried out based on hyperparameter tuning through GridSearchCV. We were able to use the grid search to determine the best number of neighbors (k) for the model, by running it across several k values and choosing a k which provided the minimum mean absolute error while not overfitting.

Model Training

KNN is non-parametric and makes predictions based on the prices of the k most similar points to the given point in the training set. All numerical features were standardized with StandardScaler before model fitting (as distance-based models such as KNN are scale-sensitive).

The training data was then fit back on the model using optimal k but no assumptions about linearity at all, which makes KNN a versatile choice for structured data.

Model prediction

GridSearchCV was performed to find the best k value and then the model of KNN was constructed with both the training and validation data as input. Predictions were the average prices of the nearest neighbors in feature space. The model successfully imitated the behavior of the training data. The predictions were largely in line with the actual values on the validation set, there were some small margins where the neighborhood-based averaging could not account for the characteristic of property.

Model Performance

The scatter plot between the actual and predicted price looked somewhat in agreement, especially near prices of middle range properties. However, predictions were spread out a bit more, especially for inexpensive homes, than with Random Forest. This may be because the model captures local patterns that are noise dependent on the dataset.

A histogram of corrections for KNN also showed some spread (but less than Random Forest) with some over- and under- predictions visible at the edge of the range. Although the model worked fine in overall, it seemed to be less robust in the estimation of the outpost.

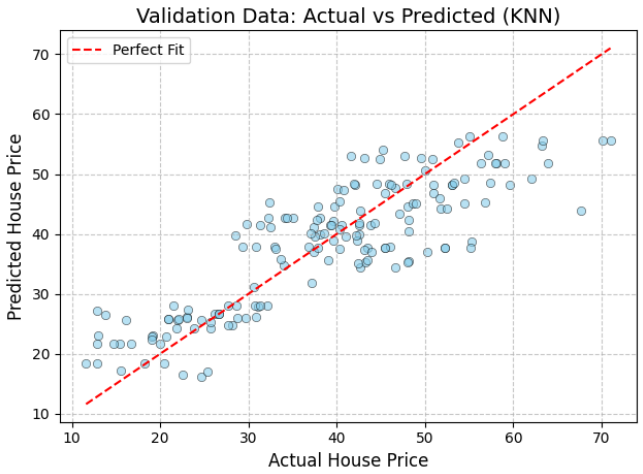

Fig.9. Actual vs Predicted Prices - K-Nearest Neighbors

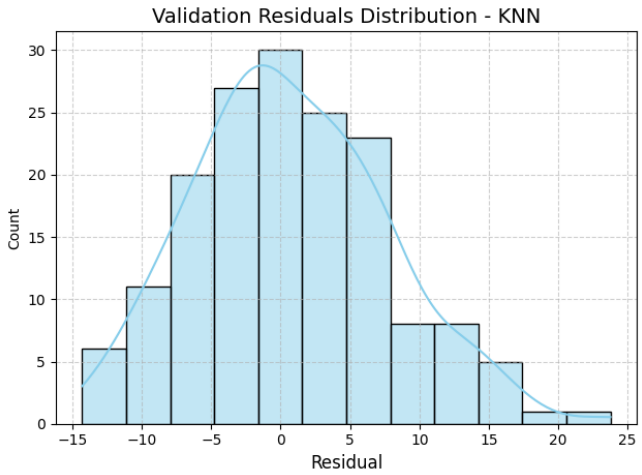

Fig.10. Actual vs Predicted Prices - K-Nearest Neighbors

Here are the regression metrics for KNN:

METRICS	VALUE
ME ($)	0.7322
RMSE ($)	6.9516
MAE ($)	5.5362
MPE (%)	-3.0110
MAPE (%)	16.2728

While KNN performed adequately, its slightly higher RMSE and MAPE suggest that it was less accurate than Random Forest in capturing the full pricing complexity.

VI. DECISION TREE (PANDAS AI MODEL)

Variable Selection

Model construction was guided by PandasAI on the conversational prompt-driven interface. The assistant loaded the clean dataset then called on PandasAI to identify the best features to predict house prices. We analysed the data and singled out house_age, number_of_convenience_stores and distance_to_the_nearest_MRT_station as the most important predictors. This action emulated a feature importance estimate through the natural language interface, as it is needed for manual processing.

Model Training

Using pandasAI, we built a Decision Tree Regressor by asking it to fit the model to the training data with the relevant features we discovered by the analysis. The code beneath the tool, including the parts splitting the training and validation sets, was all handled. Given that we did not tune the depth or any other hyperparameters manually, the decision trees were able to pick up on important splits in the data and were straightforward to interpret.

The use of this low-code interface facilitated rapid experimentation and visualization of results but restricted more fine-grained control over model complexity.

Model prediction

The machine learning PandasAI decision tree model is trained on the same cleaned dataset as the manual models. But rather than authoring the training and the prediction from scratch, we used playgrounds to guide PandasAI on training and testing using sample, step-by-step visualizations.

After training on 60% of the data, the model provided predictions over the training and validation datasets. Since PandasAI handles most of the underlying of the code, output can be interfaced directly to the assistant, which then produced predictions, for example house prices, in addition to visualizations.

The machine-learned predictions were automatically recognized and plotted with built-in charting provided by PandasAI. These outputs helped us see trends, understand model's response and validate the model's work as expected. For the lack of control over the predictions, they made manual models weaker with cruder use cases than AI models, and for the difficulty of interpreting the model behavior in connection with the AI's output, to a lesser extent.

Model Performance

The scatter plot of the true values with the predicted values showed that the Decision Tree model worked pretty good for the properties obtained at average price, but it appears that its predictions were quite unsystematic for high-priced houses. This indicates overfitting, which is typical of unpruned trees. There was also greater dispersion in residual plots, particularly at the upper and lower end of the price distribution.

Residuals' histograms showed that Decision Tree had the most dispersed errors. It appeared not to scale across all price points as well, as the extremes were working less well.

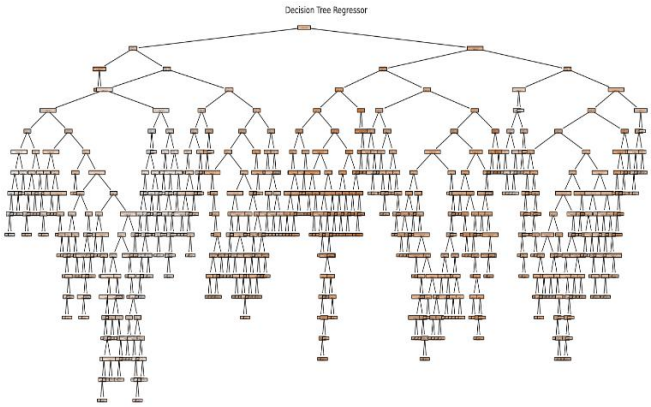

Fig.11. Decision Tree Structure Visualization

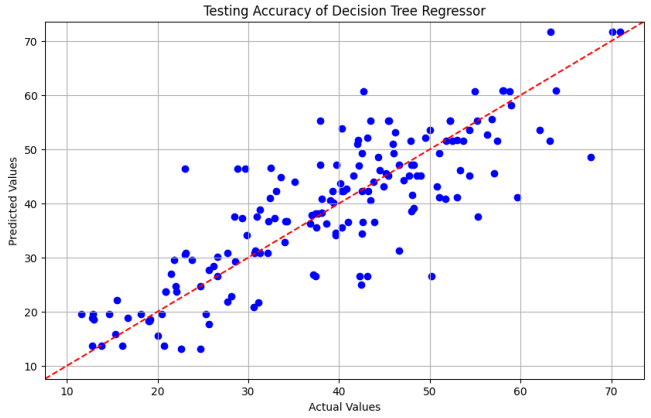

Fig.12. Actual vs Predicted Prices - Decision Tree

Here are the regression statistics for the PandasAI-based Decision Tree:

METRICS	VALUE
ME ($)	-3.7591
RMSE ($)	7.5989
MAE ($)	5.6497
MPE (%)	-10.0000
MAPE (%)	10.0000

Even though PandasAI allowed for a very rapid and hands-off learning of models, performance was weak compared to handmade models, and especially in the context of ME and variation across the validation set.

VII. MODEL COMPARISON

TABLE 2: Performance Metrics for all Models

Model	ME ($)	RMSE ($)	MAE ($)	MPE (%)	MAPE (%)
Linear Regression	1.2256	7.6140	5.8800	-1.4676	16.5966
Random Forest Regressor	0.2008	5.7360	4.4204	-2.8545	12.9027
KNN	0.7322	6.9516	5.5362	-3.0110	16.2728
Decision Tree-Pandas AI	-3.7591	7.5989	5.6497	-10.0000	10.0000

7.1 Error Metrics Selection Rationale

To compare our models for this project, we choose to use 3 error metrics. We do not use ME (Mean Error) and MPE (Mean Percentage Error) because the positive and negative values cancel out each other to give a misleading error metric that ignores large distortion in its computation.

Instead, we focus on the three-error metrics that account for all the variation in the models:

- RMSE: Accounts for huge errors, which makes it ideal for predicting price in our project.
- MAE: Less sensitive to outliers and very interpretable.
- MAPE: Shows the relative error as percentage, making it easier to understand.

7.2 Manual Models Comparison

In every error metrics the Random Forest Regressor model is the clear winner among the three manually built models. It obtains the best predictions among the three baseline models (RMSE = $5.7360; MAE = $4.4204; MAPE = 12.9027%). This result clearly points out that Random Forest is the most accurate predictor among all these three models. In contrast, Linear Regression was the worst across all indicators of error, and KNN was second. This indicates that the house price is non-linearly related to the variables.

Why Random Forest is so much better than other models lies in how it utilizes votes of multiple trees to lower both the variance and bias. This results in the model being robust to extreme data points and is able to capture intricate relationships between all variables.

7.3 Manual Models VS PandasAI

Due to the fact that Random Forest was the best model from all manual runs we did in our project it only makes sense to compare it with the Decision Tree model from the PandasAI framework. At closer examination, it could even be observed that against Decision Tree (RMSE: $7.5989, MAE: $5.6497, MAPE: 10%), Random Forest fares much better having the RMSE and MAE much lesser comparatively. Low value of MAPE of the Decision Tree means that it's better in handling percentage error than in fact all of our manual models. However, Random Forest's other error rate metrics show us that it is more accurate and stronger even than the Decision Tree.

VIII. CONCLUSION

We employed 4 machine learning models to predict the house price of the data we were provided. But we began with the data preparation for the modelling. We needed to address missing data, outliers and collinear features in the data. We also utilized or better asked our way to clean and prepared the data for the Decision Tree model with the help of the PandasAI library. Continuing to the next part of our own exploratory analysis of the data, we put faces of data variability on using e.g. distributions and correlation analysis. Finally, with KNN, we also scaled our dataset accordingly to run the model more effectively. We also discovered our best predictors in the data for each model prior to implementing the model.

For the model application, we executed three different ML models. First was Linear Regression and we looked at the co-efficient which would tell us about the potential biases in the data. Our second model was Random Forest Regressor, where we tried to tune the number of trees and the depth of the tree to acquire the best possible result. And finally, our third model, KNN, had parameter tuning via Cross-Validation and the use of fine tuning for scaling. We chose to compare the models using ME, RMSE, MAE, MPE and MAPE on test and validation data. Lastly, we applied a conversational AI wrapper in order to automatically compute most of our modeling over the Decision Tree while also calculating those error metrics.

Our model comparison revealed that the random forest regressor we have constructed manually performed the best out of all models. It achieved the minimum error, with RMSE, MAE and MAPE: $5.7360, $4.4204 and 12.9027%, respectively. This model, which aggregates predictions from a number of different decision trees, produced more accurate predictions than linear regression, KNN, and a single decision tree. Based on these findings, we suggest that the Random Forest Regressor should be employed for price prediction. 3 It's highly accurate, tells you which features you should be paying attention to and supports numerical, categorical and text data, all of which make it an excellent choice in practice for real-world pricing predictions for real estate.

IX. REFERENCES

[1] Adetunji, A. B., Akande, O. N., & Ajala, F. A. (2022). House price prediction using Random Forest machine learning technique. *Procedia Computer Science, 199*, 537–544. https://www.sciencedirect.com/science/article/pii/S1877050922001016?via%3Dihub

[2] Das, S. S. S., Ali, M. E., Li, Y.-F., Kang, Y.-B., & Sellis, T. (2020). Boosting house price predictions using geo-spatial network embedding. *arXiv preprint arXiv:2009.00254*. https://arxiv.org/abs/2009.00254

[3] R. H. Nivitha Shree, P. R. Rithick, & Mohan Kumar P. (2022). Price prediction of house using KNN based Lasso and Ridge Model. *2022 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS)*
https://ieeexplore.ieee.org/document/9760832

[4] Intel AI Blog. (2022). House price prediction using KNN regression – Intel OneAPI optimized. *Medium*.
https://medium.com/@soumyadeepdas295/house-price-prediction-using-knn-regression-intel-oneapi-optimized-867c6ce3fbb8

[5] Jha, S. B., Pandey, V., & Babiceanu, R. F. (2020). Machine learning approaches to real estate market prediction: A case study. *arXiv preprint arXiv:2008.09922*.
https://arxiv.org/abs/2008.09922

[6] Mirbagherijam, M. (2021). Housing price prediction model selection based on Lorenz and concentration curves: Empirical evidence from the Tehran housing market. *arXiv preprint arXiv:2112.06192*.
https://arxiv.org/abs/2112.06192

[7] Scikit-learn developers. (2023). Regression metrics — scikit-learn documentation. *scikit-learn.org*.
https://scikit-learn.org/stable/modules/model_evaluation.html#regression-metrics

[8] Zhang, H., & Li, Y.-B. (2020). Boosting house price predictions using geo-spatial network embedding. *arXiv preprint arXiv:2009.00254*.
https://arxiv.org/abs/2009.00254

[9] PandasAI. (2024). Conversational AI for pandas. *GitHub Repository*.
https://github.com/sinaptik-ai/pandas-ai

[10] International Journal of Research Publication and Reviews. (2025). House price prediction with Python. *IJRPR, 6*(4), 112–118.
learn.org/stable/modules/model_evaluation.html#regression-metrics