Predicting House Prices

Using Machine Learning Algorithms and PandasAI – Final Project

Anuj Bhandari
Manpreet Kaur

Apurva Naringrekar
Olga Ornek

Kathy Anusha Felix
Roja Balarathinam

CONTENTS

ABSTRACT

Accurate **prediction of housing prices** is a major challenge in real estate. This project builds **four machine learning models—Linear Regression, Random Forest, KNN and Decision Tree (through PandasAI)** using a Kaggle dataset and compares the house price prediction accuracy between them. Model performance was **evaluated using RMSE, MAE, and MAPE**. Through the comparison it became evident that **Random Forest performs better** than the other three models in predicting house prices.

INTRODUCTION

Objective

Develop a model to predict prices based on house age, MRT distance, and few other relevant features.

Significance

Predicting house prices is essential for informed real estate decisions.
Useful for buyers, sellers, and investors.
Linear Regression is simple and easy to interpret but struggles with outliers and non-linear data (Jha et al., 2020; Das et al., 2020).
Random Forest handles complex relationships better and performs well with noisy data (Adetunji et al., 2022; Mirbagherijam, 2021).
KNN assumes similar homes have similar prices but requires proper scaling and tuning to perform well (Nivitha Shree et al., 2022; Intel AI Blog, 2022).

Models Used

Linear Regression, Random Forest, K-Nearest Neighbors, and PandasAI-powered Decision Tree.

DATASET OVERVIEW

Source:
https://www.kaggle.com/code/sivakumarpradhan/price-prediction-multiple-linear-regression

Features:
No: Transaction ID
X1 transaction date: Date of the house purchase
X2 house age: The age of the house in months
X3 distance to the nearest MRT station: Distance to nearest MRT station in meters
X4 number of convenience stores: Number of convenience stores near the house
X5 latitude: Latitude of the house location
X6 longitude: Longitude of the house location

Target Variable: y_house_price_of_unit_area (Price per unit area in dollars)

Dataset Size: 414 observations, 7 variables

DATA PREPROCESSING

Basic Data Cleaning

- Formatted column names
- Handled missing values and duplicate records
- Removed irrelevant variables
- Validated data types

Irrelevant Column Removal

- Dropped transaction_id (non-predictive)
- Removed x1_transaction_date due to inconsistency and low variability

Outlier Handling

- Removed extreme price outliers (78, 78.3, 117.5)
- Capped MRT distance values above 97th percentile
- Retained latitude and longitude outliers to preserve location info

Normalization

- Applied only to KNN using StandardScaler

Note: Same steps were followed for both manual data preprocessing and preprocessing through PandasAI

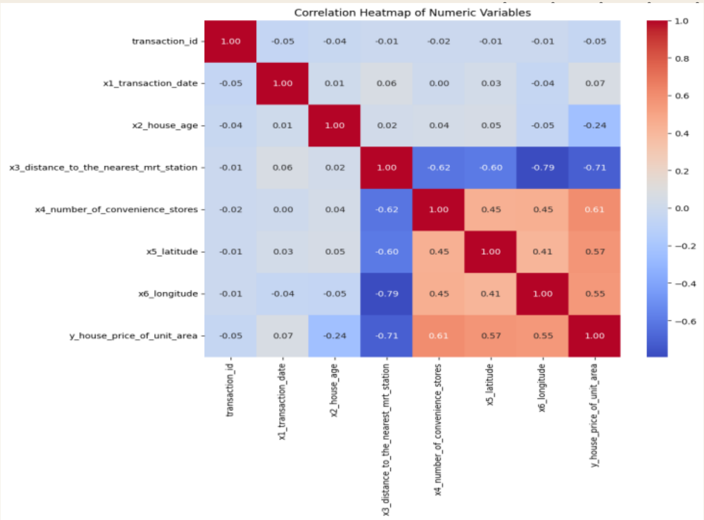
EXPLORATORY DATA ANALYSIS

Feature Distribution Highlights

- House Price per Unit Area: Right-skewed (Mean: 37.6, Q1: 27.5, Q3: 46.3)
- Distance to MRT: Right-skewed; most homes close, few far
- House Age: Spread across all age groups
- Convenience Stores: Left-skewed; mostly 0–1 nearby stores

Correlation Insights

- Distance to MRT: Strong negative correlation with price
- House Age: Moderate negative correlation
- Convenience Stores: Moderate positive correlation
- Latitude/Longitude: Weak correlation, but spatially informative



Note: Same steps were followed for both manual EDA and EDA through PandasAI

MACHINE LEARNING **MODELS**

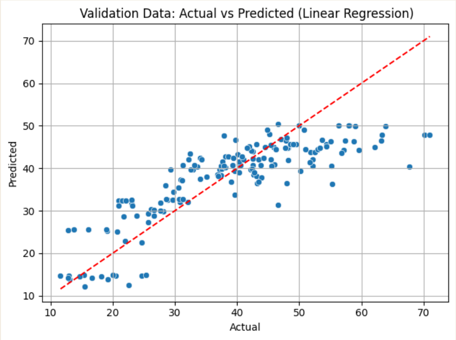
LINEAR REGRESSION

Model Overview

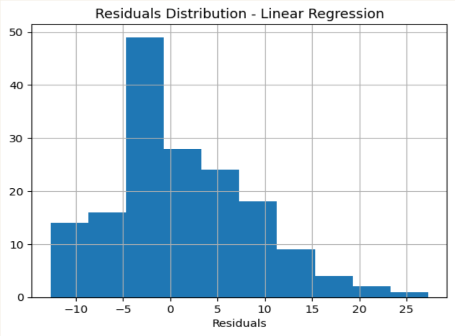
- Assumes a linear relationship between features and house price
- Feature Selection: Used Backward Elimination to retain only significant predictors
- Training Setup: 60% training, 40% validation
- Prediction: Generates a straight-line fit to generate predictions but struggles with complex or non-linear price variations.
- Strength: Captures overall trend
- Limitation: Performs poorly on expensive homes due to inability to capture non-linear patterns

LINEAR REGRESSION

Model Performance



Actual vs Predicted Prices - Linear Regression



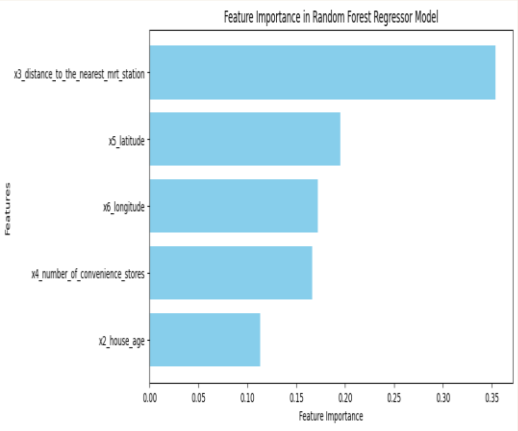
Validation Data Residuals Distribution - Linear Regression

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

RANDOM FOREST

Model Overview

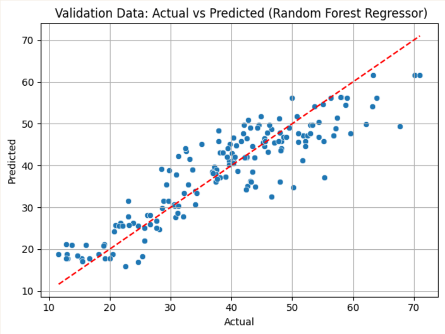
- Ensemble method combining multiple decision trees
- Feature Selection: Based on Feature Importance
- Training Setup: Same 60:40 split used to ensure fair comparison with other models.
- Prediction: Averages outputs from all trees to boost accuracy and stability
- Strength: Excellent at capturing complex, non-linear relationships
- Advantage: Effectively handles mixed data types and reduces overfitting



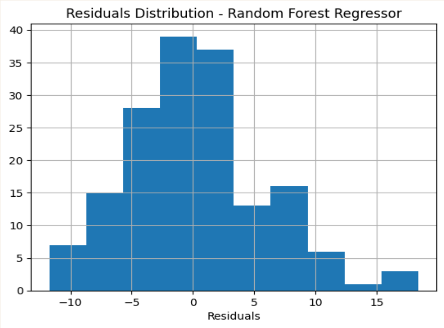
Feature Importance - Random Forest

RANDOM FOREST

Model Performance



Actual vs Predicted Prices – Random Forest



Validation Data Residuals Distribution – Random Forest

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

K – NEAREST NEIGHBOUR

Model Overview

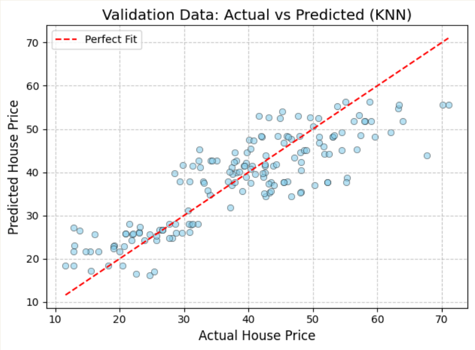
- Type: Distance-based algorithm
- Normalisation: Applied StandardScaler to scale all features
- Feature Selection: Used GridSearchCV to identify the optimal value for k and best features
- Training Setup: Model trained on 60% of data; validation performed on remaining 40%.
- Prediction: Computes Euclidean distance between data points and averages the prices of the k-nearest neighbors to generate the final prediction.
- Limitations:

    Sensitive to outliers and feature scale

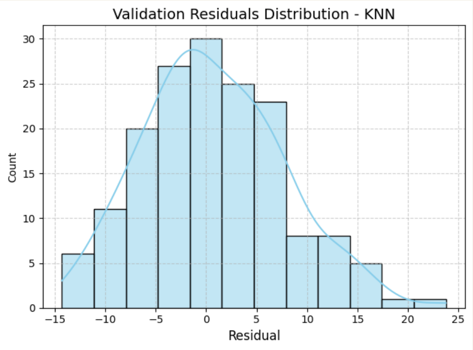
    Best suited for smaller datasets with local trends

K - NEAREST NEIGHBOUR

Model Performance



Actual vs Predicted Prices – KNN



Validation Data Residuals Distribution – KNN

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

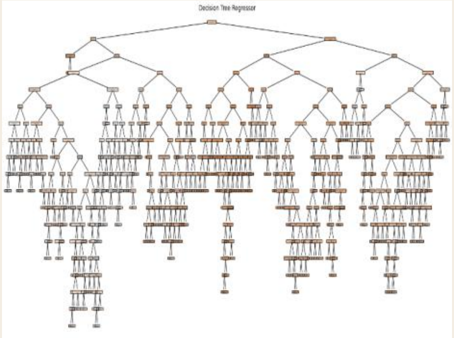
DECISION TREE – PANDASAI

Model Overview

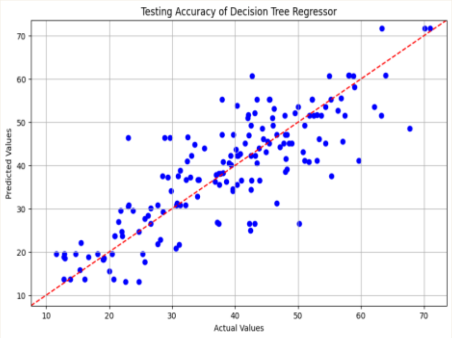
- Built using natural language prompts in PandasAI
- Enabled quick data processing and modeling without manual code
- Same data cleaning and preprocessing steps were followed through prompts
- Training Setup: Applied on the same 60:40 split, processed through natural language prompts within PandasAI.
- Prediction: Generate predictions based on a single decision tree developed during training process
- Limitations: Lacked parameter tuning controls, which limited flexibility. Further, model showed signs of overfitting - it captured both useful patterns and noise from the training data

DECISION TREE - PANDASAI

Model Performance



Decision Tree Structure Visualization



Actual vs Predicted Prices - Decision Tree

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

MODEL PERORMANCE COMPARISONS

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

- **For model performance comparison we are considering 3 robust error metrics – RMSE, MAE, and MAPE**
- **Linear Regression** performed poorly across all metrics when compared with other models due to its inability to model non-linear patterns.
- **Random Forest** performed best, with the lowest RMSE, MAE and second best MAPE. The success is due to the model using multiple tree votes to reduce both variance and bias. This makes the model being less sensitive to outliers while capturing complex interactions between all the variables.

MODEL PERORMANCE COMPARISONS

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

- **KNN** had moderate performance but was less accurate than Random Forest.
- **Decision Tree - PandasAI** Since Random Forest is clearly the best model across all the manual models developed in the project, lets compare that to the Decision Tree model powered by PandasAI. Even against Decision Tree, Random Forest performs better with its low RMSE and MAE. But decision tree comparatively, has the lowest MAPE implying lower error proportions when scaled by actual values

CONCLUSION

We used **four machine learning models** to predict house prices, starting with **data cleaning** and handling of missing values, outliers, and collinearity. We then performed **Exploratory Data Analysis** on the dataset. **PandasAI** helped prepare and explore the data for the Decision Tree model through **prompts**. Further, the data was normalized for the KNN model. **Linear Regression model** failed to adjust for non-linearity in the data, **Random Forest Regressor model** was capturing complex interactions between all the variables, **KNN model** showed good results but was sensitivity to noise and outliers, and **Decision Tree model** was overfitted by PandasAI. Models were **evaluated using RMSE, MAE, and MAPE**. **Random Forest Regressor performed best** with the lowest RMSE ($5.7360), MAE ($4.4204), and second best MAPE (12.90%), making it the most effective model for **accurate price prediction**.

THANK YOU!

REFERENCES

[1] House price prediction using Random Forest machine learning technique. https://www.sciencedirect.com/science/article/pii

[2] Boosting house price predictions using geo-spatial network embedding. https://arxiv.org/abs/2009.00254

[3] Price prediction of house using KNN https://ieeexplore.ieee.org/document/9760832

[4] Intel AI Blog. https://medium.com/@soumyadeepdas295

[5] Machine learning approaches to real estate market prediction: A case study https://arxiv.org/abs/2008.09922

[6] Housing price prediction model selection based on Lorenz and concentration curves https://arxiv.org/abs/2112.06192

[7] Regression metrics — scikit-learn documentation. https://scikitlearn.org/stable/modules/model_evaluation.html#regression-metrics

[8] Boosting house price predictions using geo-spatial network embedding. https://arxiv.org/abs/2009.00254

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

[10] House price prediction with Python learn.org/stable/modules/model_evaluation.html#regression-metrics