Machine Learning Model Report: House Price Prediction

Zinchenko Taras
Burtsev Illia

1. Introduction

This report describes the results of the python scripts which we built and executed in Google Colab for better efficiency. The Jupyter Notebook (Colab) script preprocesses data trains and evaluates a ML model for predicting house prices. The goal was to develop a model that captures the relationships between house features and the price, while also identifying the potential prediction errors. The script loads preprocessed data, trains a RandomForestRegressor model, and evaluates its results using Root Mean Squared Error (RMSE) and R-squared (R2) score.

2. Data Overview

The RandomForestRegressor model was trained on normalized and preprocessed data loaded from prep-realtor-data.csv. More information about how we normalized and proposed the data set to prep-realtor-data.csv described below.

The dataset includes about 200,000 real estate records, containing both numerical and categorical features such as:

- Bedrooms
- Bathrooms
- Lot size (acre_lot)
- House size (house_size)
- Price (our target)

Key insights about the data set:

- Target Variable (price): Highly skewed, indicating the presence of very expensive properties that might affect prediction errors.
- Numerical Features: Varied scales, e.g., lot size in thousands, house size in hundreds.
- Categorical Features: Location-based features (cities, zip codes) were present and required encoding.

Data set example (.head())

status	price	bed	bath	street	city	state	house_size
for_sale	105000	3	2	Sector Yahueca s Titulo # V84	Adjuntas	Puerto Rico	920
for_sale	80000	4	2	Km 78 9 Carr # 135	Adjuntas	Puerto Rico	1527
for_sale	67000	2	1	556G 556-G 16 St	Juana Diaz	Puerto Rico	748
for_sale	145000	4	2	R5 Comunidad El Paraso Calle De Oro R-5 Ponce	Ponce	Puerto Rico	1800

3. Selected Preprocessing Methods

- **Imputation**: To handle missing values for both num and cat data SimpleImputer tool used with median and constant strategies.
- **Scaling**: StandardScaler applied to num features such as house size and lot size, to normalize their values.
- **Encoding**: Cat features like city, property type transformed using the OneHotEncoder.
- **Column Assignment**: ColumnTransformer was used to apply the correct num or cat transformations to combined columns.
- **Preprocessing**: To efficiently chain all these steps together into a single workflow a Pipeline was used.
- **Dropping Irrelevant Cols**: Unnecessary features were removed from the dataset using the DF.drop.

It handles missing data, ensures feature scaling and transforms cat data into num data where needed for better calculation for our ML model. OneHotEncoder was used to avoid ordinal assumptions in categorical data (though, we were aware of the potential size of the processed file).
StandardScaler used to prevent numeric features with large ranges affecting model training prediction accuracy.

From the basic dataset we run the data_preprocessor.py script to apply those methods, before adding them to our ML module.

(**in the final version** data_preprocessor.py and ml_module.py **were combined** into a single Jupyter Notebook (.ipynb) script for better efficiency (explained in the code))

4. Machine Learning Models Used

We initially explored:

• Multiple Linear Regression
• Random Forest Regression

And have selected Random Forest Regression because:

a. **It handles high-dimensional data** and automatically captures nonlinear relationships.
b. **Robust to outlier**s and require minimal parameter tuning.

Early testing showed a big performance improvement for **Random Forest (R^2 ~ 80%)** over **Linear Models (R^2 ~ 19%)**.

5. Module Building

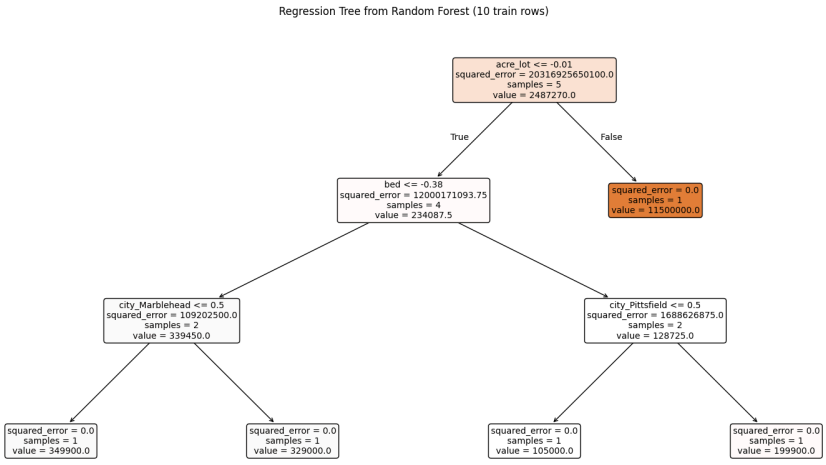
A RandomForestRegressor was implemented in a scikit-learn pipeline, using the preprocessed training data. Model parameters (like number of trees, maximum depth) were left at default values, with future potential for fine-tuning.

(You can also find all comments of each function within the code as well as Markdown cells)

Additionally, to not overload the main python script is running, we created python script for short dataset, to visualize a single decision tree from a RandomForestRegressor (eventually integrated in the ML_Final_Model.ipynb),

• Loads preprocessed data.
• Splits it into 6 rows for training and 6 for testing.
• Trains a RandomForestRegressor on those 6 training rows.

• Visualizes the first tree in the forest and saves it as regression_tree.png:



The initial process was built across three Python scripts:

- data_preprocessing.py: This script handled all data cleaning and feature normalization.
- ml_module.py: This was the main script used for training and evaluating the machine learning model.
- ml_module_visual.py: Was used to visualize a single decision tree, offering insight into the model's logic.

This modular approach ensured a structured workflow. These three scripts were combined into a single Jupyter Notebook cell to reduce the time needed to run the code and resolve runtime errors. Additionally metplotlib was used for visualistion to get more insights.

6. Evaluation Results

The dataset was divided into:

• Training set: 80% of the data

• Test set: 20% of the data

To evaluate the model's performance, we used two key metrics:

• Root Mean Squared Error: Measures the average size of prediction errors.

• R-squared (R^2): Provide the proportion of variance in the price of the house (target variable) explained by the model.

Results:

R-squared (R2 Score): 0.9724

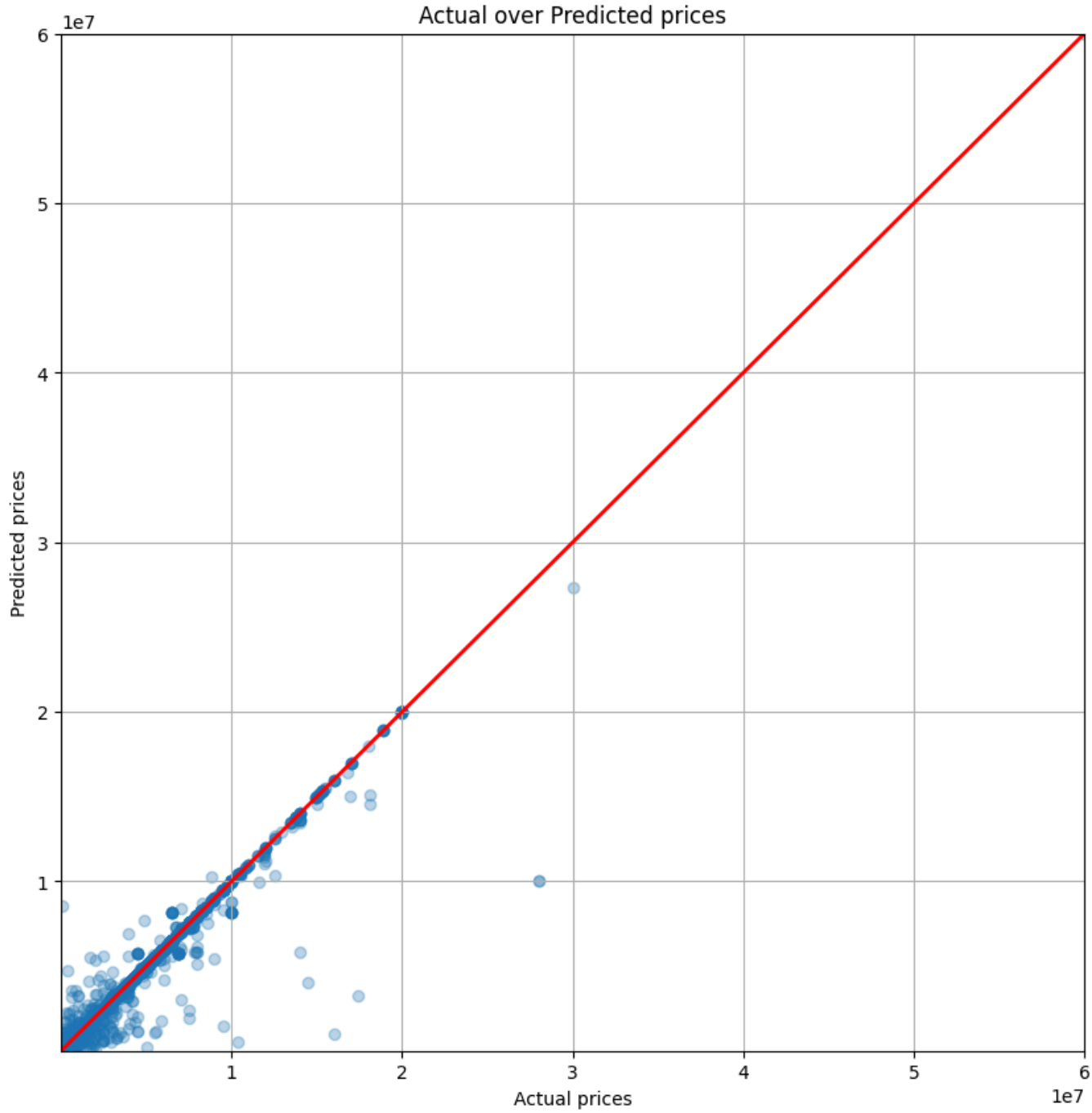
- An R2 score of 0.9724 is generally very high. It suggests that approximately 97.24% of the variability in the house prices in the test set can be explained by the features the model used. The model seems to capture a very large portion of the factors that influence house prices according to the data.

Root Mean Squared Error (RMSE): 235,396.7246

- It indicates the typical or average magnitude of the error in the model predictions, in the same units as the target variable (price). ~$200,000 seems like a big error but it is important to consider that the dataset contains a significant number of multi-million dollar properties.

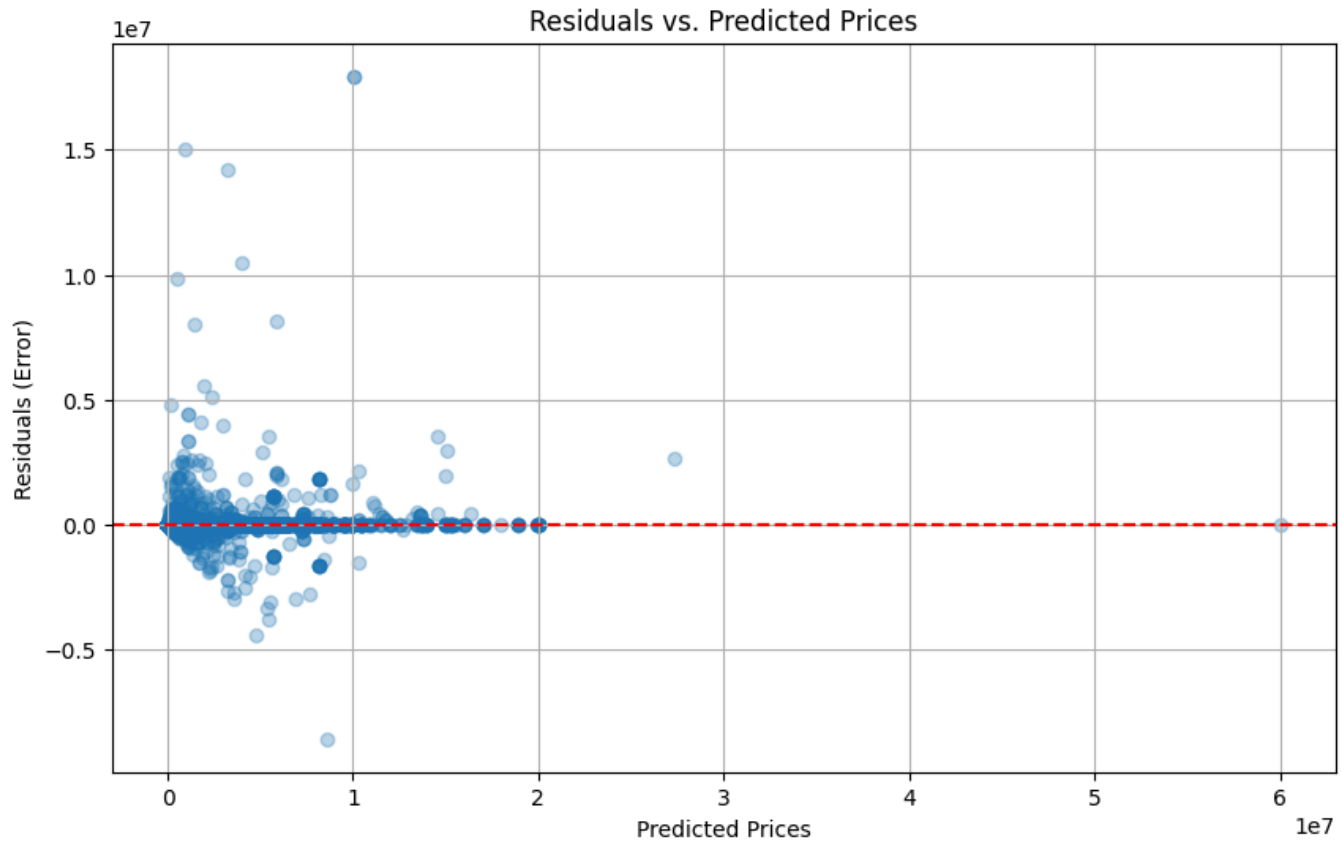
Conclusion: For these very expensive homes, the ~$235,000 error may represent a smaller percentage of the total property value. However, this is still a large absolute error that will be especially significant for any lower or mid-priced homes in the data set.

To gain a deeper understanding of the model behavior, the result was visualized with three key plots:



This scatter plot visually confirms the high R-squared value. A strong linear relationship is evident, with most data points clustered around the red 45 degree line, indicating accurate predictions for most properties.

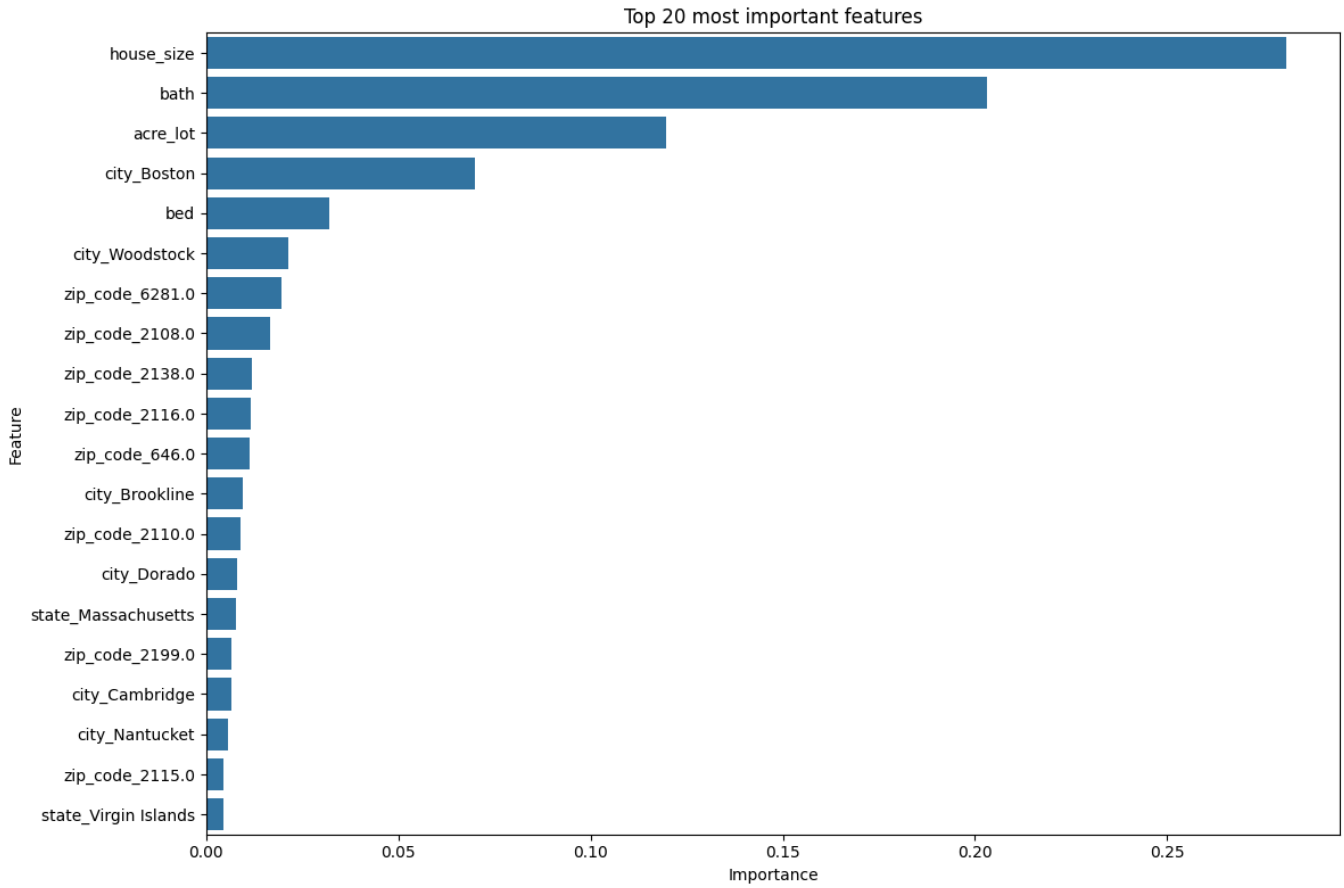
There are also some outliers that are visible, particularly in the higher price ranges, where the accuracy of the model declines.



This residual plot shows a pattern of heteroscedasticity. Although the forecast errors are centered around zero, their variance increases as the predicted price increases. This shows that model is very accurate for low-priced homes, but has a much larger error in predicting the value of more expensive properties.

This visual evidence supports the earlier analysis of the RMSE value. Average error is high not because the model is moderately inaccurate everywhere.

Instead, the RMSE value is being pulled up by the very large, though less frequent, errors on the most expensive properties.



The feature importance plot shows, as expected, physical attributes like house_size, the number of bath rooms, and acre_lot size are the most influential factors. Crucially, location-based features, including city_Boston and several specific zip codes, rank highly in the top 20.

This indicates that geographical location plays a big role in forming a price of a house and validates the use of one-hot encoding to capture this information effectively.

*8. Conclusion

We successfully developed a ML model for predicting house prices, achieving an **R-squared** of **0.9728** which is a very **good result**.

The model explains over **97% of the price variability,** but it also has a significant root mean square error of **~$233,442**.

Further **visual analysis confirmed** this two-sided performance. The **model** is very **accurate** for most properties, but loses accuracy on expensive homes, where errors can be significant. The most influential predictors turned out to be physical attributes such as house_size or geographic factors such as city_Boston. In summary, the model is effective at capturing overall market trends, but has clear limitations in its accuracy at the high end of the market.

Resources

"How does Random Forest work" :
https://builtin.com/data-science/random-forest-algorithm#:~:text=Random%20forest%20is%20an%20algorithm,and%20produce%20more%20accurate%20predictions.

"Decision Tree vs Random Forest"
https://www.geeksforgeeks.org/difference-between-random-forest-and-decision-tree/

Youtube tutorials and open sourced guides:

https://stackoverflow.com/questions/46137945/random-forest-classifier
https://www.youtube.com/watch?v=xlqX1dqcNbY
https://www.youtube.com/watch?v=J4Wdy0Wc_xQ

Machine Learning course