Analysis of the iBuyer Model for House Price Prediction and Profitability Evaluation

Introduction

In this report, we evaluate the profitability of the iBuyer business model by leveraging data-driven analysis and machine learning predictions. The objective is to develop a model that can accurately predict property sale prices, enabling the iBuyer to make informed offers that align with a target profit margin. Using a multilayer perceptron (MLP) model, we trained and evaluated several configurations to identify the optimal setup based on the predictive performance on a validation dataset. Our analysis includes both model training and validation error rates, as well as a profit analysis based on hypothetical scenarios aligned with the iBuyer business framework. This report outlines our methodology, model evaluation results, and an assessment of profitability based on predicted prices.

Methodology

1. Data Loading and Initial Exploration

Data was sourced from Kaggle, including both training and test datasets. The training dataset contained 11,581 samples with 16 features each, including the target variable SALE_PRICE, whereas the test dataset had 4,964 samples and lacked this target variable. Initial exploration included:

- **Histogram Analysis**: A histogram of SALE_PRICE revealed a wide price range, from approximately 50,000 to 2,000,000, with a median price of 431,000. This informed the choice of normalization to aid in model convergence.
- **Correlation Matrix**: A heatmap highlighted significant correlations between SALE_PRICE and certain features, particularly LIVING_SQFT and FULL_B (full bathrooms).

2. Data Preparation and Preprocessing

Several data preprocessing steps were implemented to optimize the features for model input, including:

- **Feature Scaling**: SALE_PRICE was divided by a normalization factor of 100,000, bringing the median price close to 4.31 in the new scale. This reduction facilitated faster and more stable training.

- **One-Hot Encoding**: Non-numeric features (NBHD, PROP_CLASS, and STYLE_CN) were converted into dummy variables using one-hot encoding, increasing the feature space to 105 columns.
- **Feature Standardization**: For numeric features, we performed z-score normalization, centering them around 0 with a standard deviation of 1, ensuring uniform scale across all features and aiding in model convergence.
- **Missing Value Imputation**: Numeric columns with missing values were filled with zeros post-standardization to maintain consistent input size across samples.

3. Data Splitting and Batching

To create a robust validation process:

- **Train-Validation Split**: 80% of the data was reserved for training and 20% for validation using train_test_split. This enabled tracking of both in-sample and out-of-sample performance, aiding in model selection.
- **DataLoader Setup**: Utilizing TensorDataset and DataLoader with batch sizes of 64 for training and 128 for validation/test ensured efficient data handling and gradient descent updates.

4. Linear Regression Benchmark

As a baseline, a simple linear regression model with a single output neuron was implemented. This model, trained over 200 epochs, utilized:

- **Mean Squared Error (MSE)** as the loss function, focusing on minimizing the absolute difference between predicted and actual sale prices.
- **Median Error Rate (MER)** for validation. MER, defined as the median of absolute relative errors, allowed for comparison in relative terms, making it particularly relevant for house prices, which span a wide range.

5. Multi-Layer Perceptron (MLP) Base Model

A multi-layer perceptron model with two hidden layers (sizes 256 and 128) was created to capture non-linear relationships between features. Key aspects of this model included:

- **ReLU Activations**: ReLU (Rectified Linear Unit) activation was applied after each hidden layer to introduce non-linearity, essential for capturing complex patterns.
- **Training Process**: The model was trained over 200 epochs with the Adam optimizer (learning rate 0.001) and MSE loss function, recording MER for both training and validation at each epoch.

6. Enhanced MLP Model with Four Hidden Layers

Building on the previous architecture, an enhanced model with four hidden layers (sizes 512, 256, 128, and 64) was developed to improve prediction accuracy. This model involved:

- **LeakyReLU Activation**: To address potential dead neuron issues in ReLU, **LeakyReLU** activations were applied in the hidden layers of this configuration, enhancing performance on sparse data and ensuring gradient flow.
- **Learning Rate Optimization**: A learning rate of 0.001 was chosen through experimentation, allowing the model to converge efficiently without overshooting.
- **Additional Layers**: The added complexity in layers was aimed at capturing intricate relationships among features, particularly for high-dimensional inputs post-one-hot encoding.

7. Regularization Techniques

To counter potential overfitting, two primary regularization techniques were incorporated:

- **L2 Regularization (Weight Decay)**: Implemented within the Adam optimizer with a decay parameter of 0.001, penalizing large weights and encouraging simpler model representations.
- **Dropout Layers**: Dropout was applied after each hidden layer with a probability of 0.1 or 0.2 (depending on model variant) to randomly deactivate neurons, further enhancing generalization.
- **L1 Loss Function**: In certain model configurations, **L1 loss** was utilized as an additional measure to enhance robustness. Unlike MSE, L1 loss minimizes the absolute differences between predictions and actual values, which is less sensitive to outliers. This made it particularly valuable when fine-tuning models with complex data, contributing to more stable predictions.

8. Hyperparameter Tuning and Model Selection

Several variations of the MLP model were experimented with, adjusting hyperparameters such as hidden layer sizes, dropout probabilities, and learning rates. Each configuration was evaluated based on its minimum validation MER, and the best-performing model was selected based on:

- **Learning Rate**: Optimized at 0.003, 0.001 and 0.0005.
- **Dropout Probability**: Optimal dropout rates of 0.1 or 0.2.
- **Weight Decay**: Fixed around at 0.001 to balance underfitting and overfitting.

9. Model Evaluation and Selection of the Best Model

The best model configuration was saved using a checkpoint containing its state dictionary, optimizer state, and hyperparameters. This saved configuration allowed for easy retrieval and inference on the test set. Key hyperparameters of the best model were:

- **Hidden Layers**: [512, 256, 128, 64]
- **Learning Rate**: 0.003
- **Dropout Probability**: 0.1
- **Weight Decay**: 0.001
- **Epochs**: 300
- **Loss Functions:** Both MSE and L1 loss functions were used across different configurations to optimize performance.

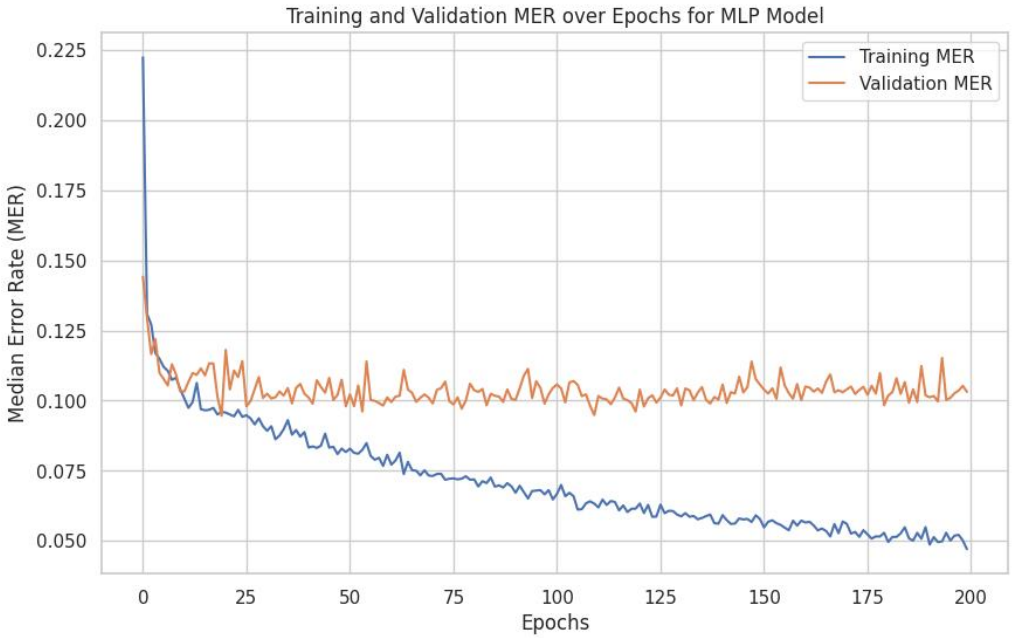
10. Profit Evaluation of iBuyer Business Model

To assess the profitability of the iBuyer business model, historical data was leveraged to generate predicted sale prices. Evaluation steps included:

- **Signed Error Calculation**: For each property, the signed error rate was computed as the difference between predicted and actual prices, divided by the actual price. This metric served as a proxy for the profitability of the iBuyer model.
- **Profit Margin Assumption**: A 12% profit margin (including commissions and holding costs) was assumed to evaluate potential returns.

Model Analysis and Results

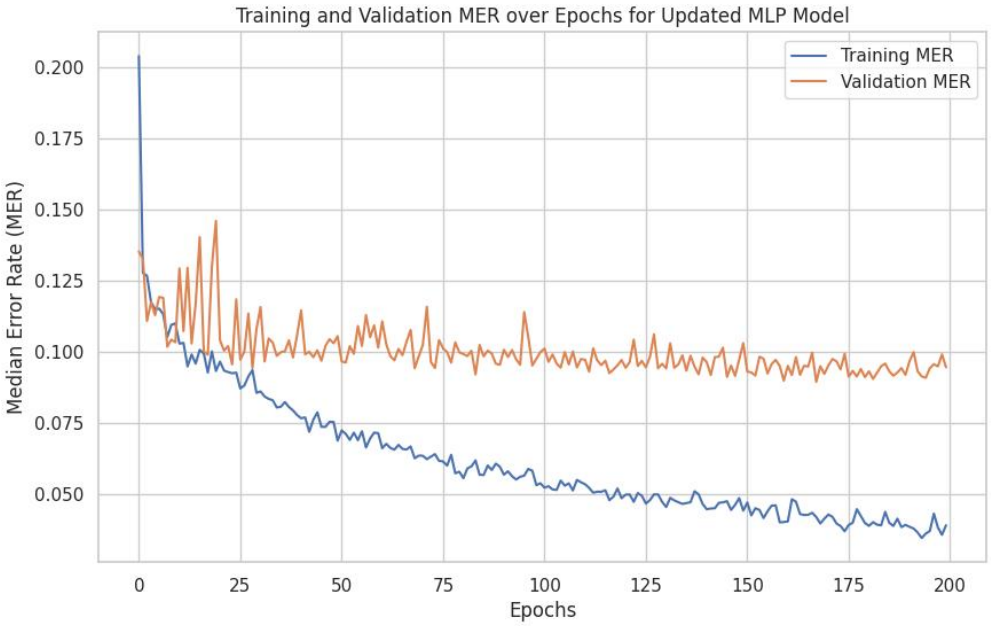
1.MLP with 2 hidden layers of sizes 256 and 128



The plot shows a steep decline in error within the first few epochs, with both training and validation errors stabilizing after epoch 13. The training MER continues to gradually decrease, while the validation MER exhibits more fluctuation but remains relatively stable after epoch 25.

While the MLP model demonstrates effective learning with a final MER significantly lower than its initial value, the sustained difference between training and validation errors indicates that the model might still struggle to generalize fully to unseen data. The final Train MER is 0.0471, and the final Validation MER is 0.1032. This gap suggests potential overfitting.

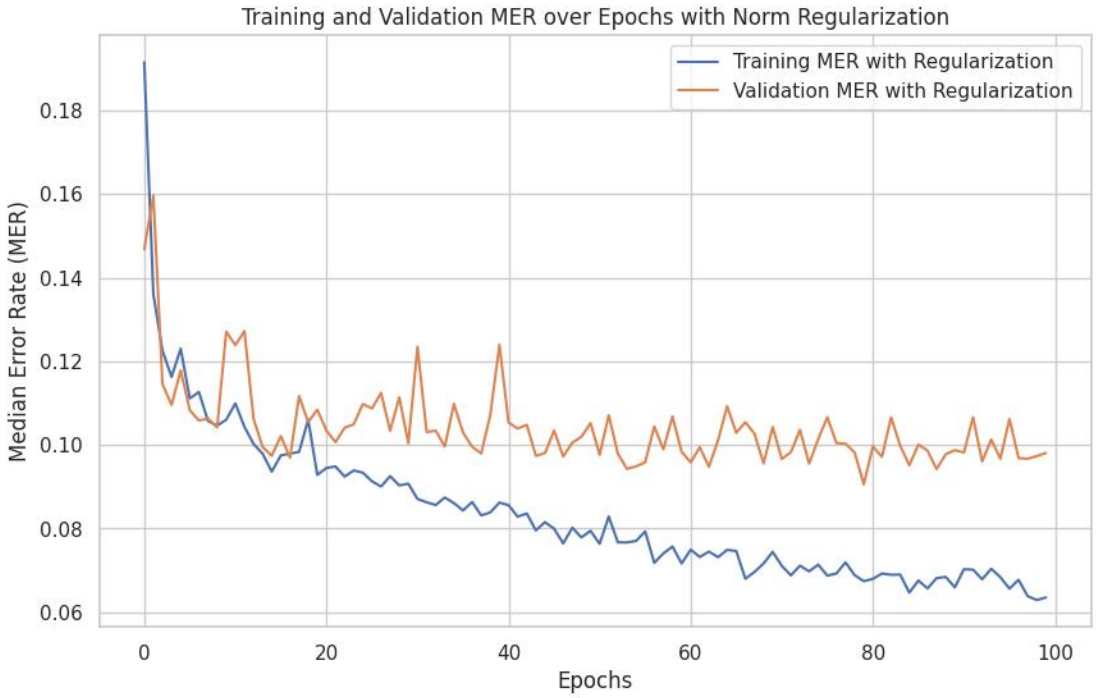
2. MLP with 4 hidden layers of sizes 512, 256, 128, 64



This plot shows a steep decline in error within the initial epochs, with the training error continuing to decline steadily, while the validation error stabilizes after epoch 25. Both metrics eventually show signs of convergence, with the training MER approaching lower values as training progresses.

While the MLP model demonstrates effective learning with a final MER significantly lower than its initial value, the sustained difference between training and validation errors indicates that the model might still struggle to generalize fully to unseen data. The final Train MER is 0.0389, and the final Validation MER is 0.0944. This gap suggests potential overfitting.

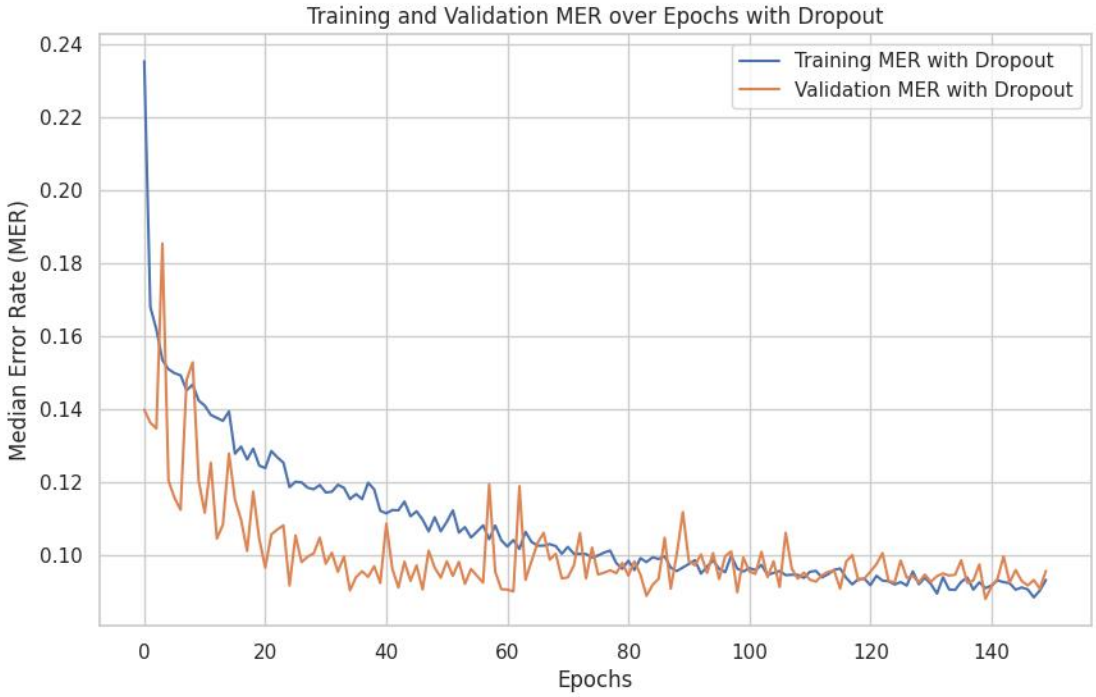
3. MLP with 4 hidden layers of sizes 512, 256, 128, 64 and norm regularization



This plot shows a steep decline in error within the initial epochs, with the training error continuing to decline steadily, while the validation error stabilizes after epoch 20. Both metrics eventually show signs of convergence, with the training MER approaching lower values as training progresses.

While the MLP model demonstrates effective learning with a final MER significantly lower than its initial value, the sustained difference between training and validation errors indicates that the model might still struggle to generalize fully to unseen data. The final Train MER is 0.0635, and the final Validation MER is 0.0981. This gap suggests potential overfitting.

4. MLP with 4 hidden layers of sizes 512, 256, 128, 64 and norm regularization and dropout layers



This plot for the MLP model with Norm Regularization and Dropout shows a rapid decrease in both training and validation errors in the first few epochs, with both metrics approaching convergence after around epoch 60. It is worthy to note that the validation MER quickly aligns with the training MER and remains at similar levels since epoch 80, indicating improved generalization compared to previous versions of the model.

The addition of Dropout mitigated overfitting, which is proved by the consistent overlap between the training and validation errors. This suggests that the model is learning to generalize well without overly fitting to the training data. The final Train MER is 0.0933 and the final Validation MER is 0.0958, which are relatively low, further supporting the model's robustness in capturing underlying patterns in the data while avoiding overfitting.

5. Table listing all the model hyperparameters we have tried

	Model	Hidden Neurons	Learning Rate	Dropout Probability	Weight Decay	Min Validation MER
0	1	[512, 256, 128, 64]	0.0030	0.1	0.001	0.079328
1	2	[512, 256, 128, 64]	0.0010	0.1	0.001	0.078984
2	3	[512, 256, 128, 64]	0.0010	0.2	0.001	0.081088
3	4	[512, 256, 128, 64]	0.0010	0.1	0.002	0.080180
4	5	[512, 256, 128, 64]	0.0005	0.2	0.001	0.085762
5	6	[256, 128, 64]	0.0030	0.1	0.001	0.080261
6	7	[256, 128, 64]	0.0010	0.1	0.001	0.082627
7	8	[256, 128, 64]	0.0010	0.2	0.001	0.081342
8	9	[256, 128, 64]	0.0010	0.1	0.002	0.080132
9	10	[256, 128, 64]	0.0005	0.2	0.001	0.083114

Model: Model number.
Hidden Neurons: Number of neurons in each hidden layer of the model.
Learning Rate: The rate at which the model updates weights during training.
Dropout Probability: The probability of randomly dropping neurons during training.
Weight Decay: Regularization strength to penalize large weights.
Min Validation MER: Minimum MER achieved on the validation set, indicating model performance.

6 Profit analysis

Q1

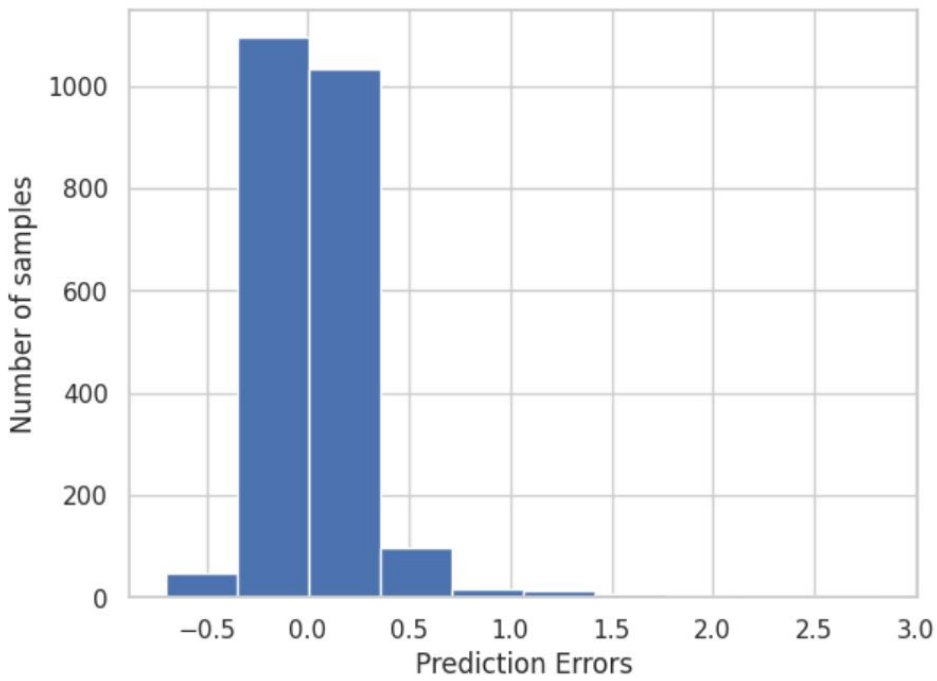
The **bias** of prediction errors can be interpreted as the average deviation of the model's predicted prices from the actual prices. In this context, bias indicates whether the model tends to overestimate or underestimate prices.

From the calculations, we have a slight positive mean of **0.0439**, which suggests that, on average, the model slightly overestimates property prices. This is a small but positive bias.

According to the following code, we get the graph.

```
import matplotlib.pyplot as plt
plt.xlabel("Prediction Errors")
plt.ylabel("Number of samples")
plt.hist(prediction_errors)
```

```
(array([4.700e+01, 1.095e+03, 1.034e+03, 9.700e+01, 1.600e+01, 1.200e+01,
        7.000e+00, 4.000e+00, 1.000e+00, 4.000e+00]),
 array([-0.70330576, -0.34948631,  0.00433313,  0.35815258,  0.71197202,
         1.06579147,  1.41961091,  1.77343036,  2.12724981,  2.48106925,
         2.8348887 ]),
 <BarContainer object of 10 artists>)
```



The histogram shows that the majority of prediction errors are centered around **0**, indicating that most predictions are close to the actual sale prices.
There is a higher concentration of errors slightly above **0**, which aligns with the positive mean prediction error. This further confirms the slight overestimation bias.
A small number of predictions have larger errors (both positive and negative), but these are relatively rare compared to the central cluster around zero.
The histogram and the positive mean error indicate a minor bias toward overestimating sale prices, as indicated by the positive mean prediction error of about **0.0422**. This bias is relatively small, suggesting that the model generally performs well in terms of accuracy, with most predictions being close to the actual values.

Q2

Profit Analysis: Hypothetical Acceptance of All Offers

In this analysis, we assess the profitability of the iBuyer business model under a hypothetical scenario where every property owner accepts the iBuyer's offer, irrespective of the offer amount. The iBuyer sets an offer price OPOPOP for each property based on the model's predicted price PPPPPP, calculated as:

$$OP = \frac{PP}{1+\alpha}$$

where alpha is the targeted profit margin, set at 12% in this case. This margin accounts for various costs the iBuyer incurs, such as transaction fees, administrative costs, and holding costs. Notably, this target profit margin is designed to be competitive, given that traditional realtors like Zillow typically charge a commission of around 7.5%, along with possible repair costs. The 12% margin encompasses both expected profit and a buffer for additional fees.

Under the assumption that all offers are accepted, we can calculate the average percentage profit as:

$$\frac{SP - OP}{OP}$$

Using this formula, where SP is the actual sale price from the validation data, we find:

```
op = valid_predictions / (1 + profit_margin)
np.mean((actual_prices - op) / op)
```

The result of this calculation is approximately **12.38%**.

Comparison to Target Profit Margin α

The computed average percentage profit of 12.38% is very close to the targeted profit margin α=12%. This indicates that if all offers were hypothetically accepted, the iBuyer would achieve a profit margin slightly above the target. This difference is minimal, suggesting that the pricing strategy based on the 12% target margin effectively captures the intended profit range.

The minimal difference (an additional 0.38% above the target margin) suggests that the iBuyer's model is well-calibrated to achieve its profit goals under ideal conditions. In a real-world scenario, some offers may not be accepted, but this hypothetical scenario serves as a benchmark, demonstrating that the iBuyer could meet or exceed its profit objectives if its offers were accepted universally. This finding supports the robustness of the profit margin assumption and the iBuyer's pricing strategy.

Q3

In this section, we evaluate the profitability of the iBuyer model when accounting for selective offer acceptance by homeowners. Given that each homeowner may have a perceived valuation of their property, they are likely to accept an offer only if it meets or exceeds a certain threshold.
We approximate this threshold using the actual sale price as a proxy for perceived valuation. Specifically, we assume homeowners are willing to accept an offer if it meets the following condition:

$$OP > (1 - \beta)SP$$

Where β=10% is a discounting factor that reflects both the convenience of a quick transaction and an assumed realtor commission of around 6%.

This acceptance rule implies that homeowners will accept an offer that is no more than 10% lower than the actual sale price, reflecting their valuation of both speed and reduced hassle compared to traditional real estate transactions.

Using the offer acceptance rule, we calculated the percentage profit for properties where the offers were accepted. The mean percentage profit among accepted offers is as follows:

```
bought = op > (1 - 0.1) * actual_prices
np.mean(((actual_prices - op) / op)[bought])
```

The result indicates an average loss of approximately 2.29% on the accepted offers, which deviates significantly from the targeted profit margin of 12%. This outcome suggests that under the current model and acceptance criteria, the iBuyer model may struggle to achieve profitability on accepted offers alone. This loss highlights the impact of the selective acceptance behavior of homeowners, who may only accept

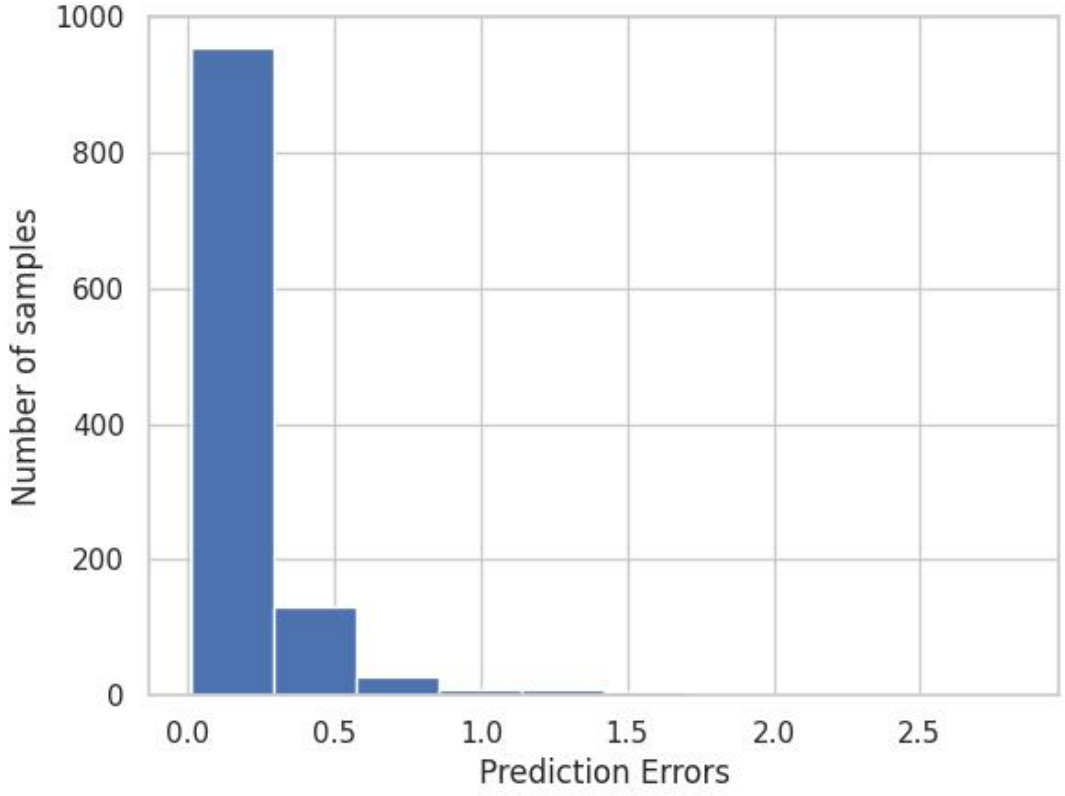
offers that are close to their perceived valuation, thereby reducing the potential profit margin.

The observed loss for accepted offers can be attributed to the bias in prediction errors and the conservative nature of homeowners' acceptance. The discounting factor, while accommodating for realtor fees and convenience, still limits the iBuyer's ability to purchase at a substantial discount relative to the resale price.

Q4

Bias Analysis of Prediction Errors for Accepted Offers

When analyzing the properties whose owners accepted the offers, we observe a **mean prediction error of 0.183**, as shown in the histogram. This positive mean prediction error suggests that the predicted prices tend to overestimate the actual prices for the subset of properties where the offers were accepted.



P

Positive Bias: The mean prediction error of 0.183 indicates that, on average, the iBuyer's model overestimated the property values by 18.3% for the accepted offers. This overestimation is beneficial from the perspective of offer acceptance, as higher predicted values likely result in more attractive offers. However, it leads to lower profitability because the purchase prices tend to be higher than the resale values, as we noted in Question 3.

Distribution of Prediction Errors: The histogram shows a concentration of prediction errors near zero, with a positive skew. Most prediction errors are close to zero, indicating that for the majority of accepted offers, the predictions were close to

the actual prices. However, the tail extends to higher positive errors, which raises the average error. This skewed distribution suggests that while most accepted offers are priced reasonably close to actual values, a portion is significantly overestimated.
Impact on Profitability: The positive bias explains why the mean percentage profit for accepted offers was negative (-2.29%), as calculated in Question 3. Since the iBuyer is overestimating property values in accepted offers, they end up purchasing properties at prices close to or above actual resale values. Consequently, these transactions lead to losses rather than profits, deviating from the targeted profit margin of 12%.

Conclusion

The iBuyer model demonstraterofitability when all offers are accepted. However, selective acceptance by homeowners introduces significant challenges, as the model's tendency to overestimate prices impacts profitability negatively. Future improvements could focus on refining predictive accuracy and adjusting offer pricing strategies to better align with market realities, minimizing the risk of overpaying for properties in selective acceptance scenarios.