

## 1. Real Estate Price Prediction using Artificial Neural Networks

### 1.1. Objective

The primary objective of this project was to implement and evaluate an Artificial Neural Network (ANN) for a regression task: predicting real estate prices based on historical transaction records. The project involved preprocessing data, an implementation of a Multi-Layer Perceptron (MLP), justifying the final model architecture by experimenting with different hyperparameters, and its evaluation on an unseen test set.

### 1.2. Data Preprocessing

The process began with the "Apartment for Rent Classified" dataset from the UCI Machine Learning Repository. The raw data required several cleaning and preprocessing steps to be suitable for a machine learning model.

#### 1.2.1. Initial Cleaning and Feature Selection

1. **Data Loading:** The dataset was loaded directly from the UCI repository.
2. **Type Conversion:** Columns expected to be numerical (e.g., price, bathrooms, bedrooms) were converted to a numeric data type. Any values that could not be converted were set to NaN.
3. **Column Removal:** Several columns were dropped due to being irrelevant for this regression task or having a high percentage of missing values. This included: `pets_allowed`, `address`, `title`, `body`, `source`, `cityname`, `state`, and `price_display`. The currency column was also dropped as it contained only a single value ("USD").
4. **Handling Missing Rows:** Rows with missing values in critical columns such as price, bathrooms, and location data (latitude, longitude) were removed to ensure data integrity.

#### 1.2.2. Data Splitting and Preprocessing Pipeline

To prevent data leakage, cleaned data was split into a training set (80%) and a test set (20%) before any feature scaling or encoding was applied.

A ColumnTransformer pipeline was established to simplify preprocessing for numerical and categorical features separately:

- **Numerical Features:** `bathrooms`, `bedrooms`, `square_feet`, `latitude`, `longitude`, `time`
  - ❖ **Imputation:** Imputed missing values using the median of each column.
  - ❖ **Scaling:** The features were scaled using StandardScaler to a mean of 0 and a standard deviation of 1, which is essential for the proper training of neural networks.
- **Categorical Features:** `category`, `amenities`, `fee`, `has_photo`, `price_type`
  - ❖ **Imputation:** Imputed missing values with the most frequent value in each column.

- ❖ **Encoding:** Features was converted to numeric using one-hot encoding through OneHotEncoder. This makes binary columns out of each category. So, it will allow the model to read them not presuming any ordinal relationship.

Training data was in shape of (79686, 8920) and test data in shape of (19922, 8920) after passing through this pipeline. The high number of columns was due to one-hot encoding on the high cardinality amenities feature.

### 1.3. ANN Implementation and Architecture Justification

A Multi-Layer Perceptron (MLP), a type of feedforward ANN, was implemented using TensorFlow and Keras. To justify the final model architecture, a comparative analysis of different hyperparameter combinations was conducted.

#### 1.3.1. Hyperparameter Comparison

Three distinct model configurations were trained and evaluated based on their performance on a validation set (20% of the training data, handled internally by Keras's `validation_split`):

1. **Baseline Model:** 2 hidden layers with 64 neurons each, Adam optimizer, and ReLU activation.
2. **Wider Model (A):** 2 hidden layers with 128 neurons each, RMSprop optimizer, and ReLU activation.
3. **Wider Model (B):** 2 hidden layers with 128 neurons each, Adam optimizer, and LeakyReLU activation.

The results of the comparison, based on the minimum validation loss achieved during training, are as follows:

**Table 1:** Results of the comparison between model architecture

Model Architecture	Minimum Validation Loss
Baseline (64x2, Adam, ReLU)	561,755.44
Wider (128x2, RMSprop, ReLU)	572,974.94
Wider (128x2, Adam, LeakyReLU)	575,334.38

#### 1.3.2. Final Model Selection

Based on the empirical results, the Baseline model (2 hidden layers, 64 neurons, Adam optimizer, ReLU activation) was selected as the final model. It achieved the lowest validation loss, indicating the best ability to generalize to unseen data among the tested configurations.

#### 1.4. Performance Evaluation

The final, selected model was evaluated on the completely unseen test set. The performance metrics are summarized below:

- **Mean Absolute Error (MAE):** \$300.75
- **Mean Absolute Percentage Error (MAPE):** 20.93%
- **Root Mean Squared Error (RMSE):** \$561.95
- **R-squared ( $R^2$ ):** 0.6118

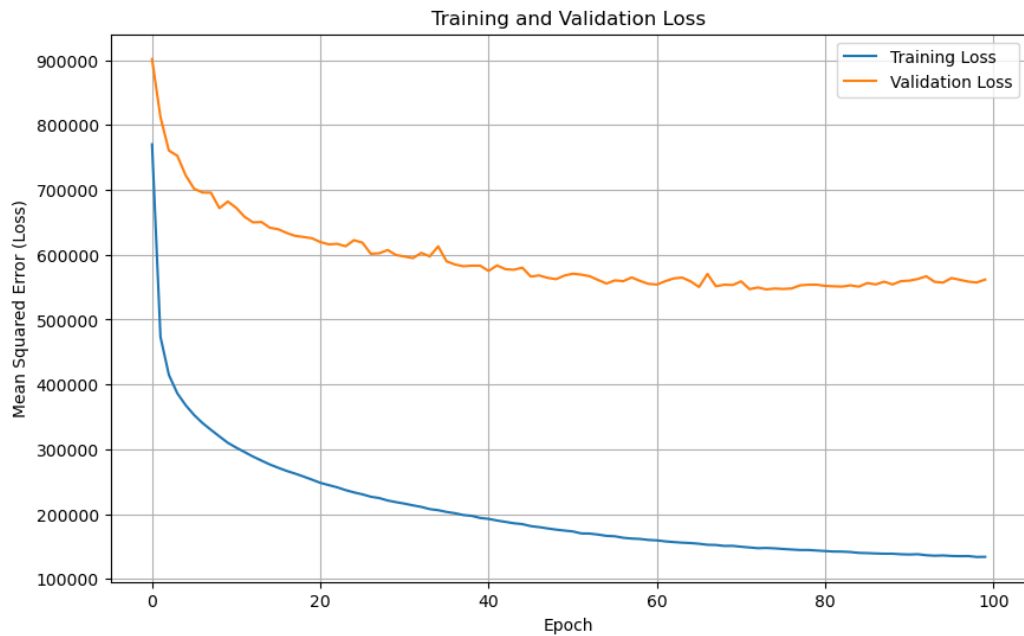


Fig. 1 Training & validation loss of the baseline model

## 1.5. Visualization of Results

To better understand the model's behavior, two key visualizations were generated.

### 1.5.1. Actual vs. Predicted Prices

This scatter plot compares the actual prices from the test set (x-axis) against the prices predicted by the model (y-axis). A perfect model would have all points lying on the 45-degree red dashed line. The plot shows a clear positive correlation, with most predictions clustering around the line.

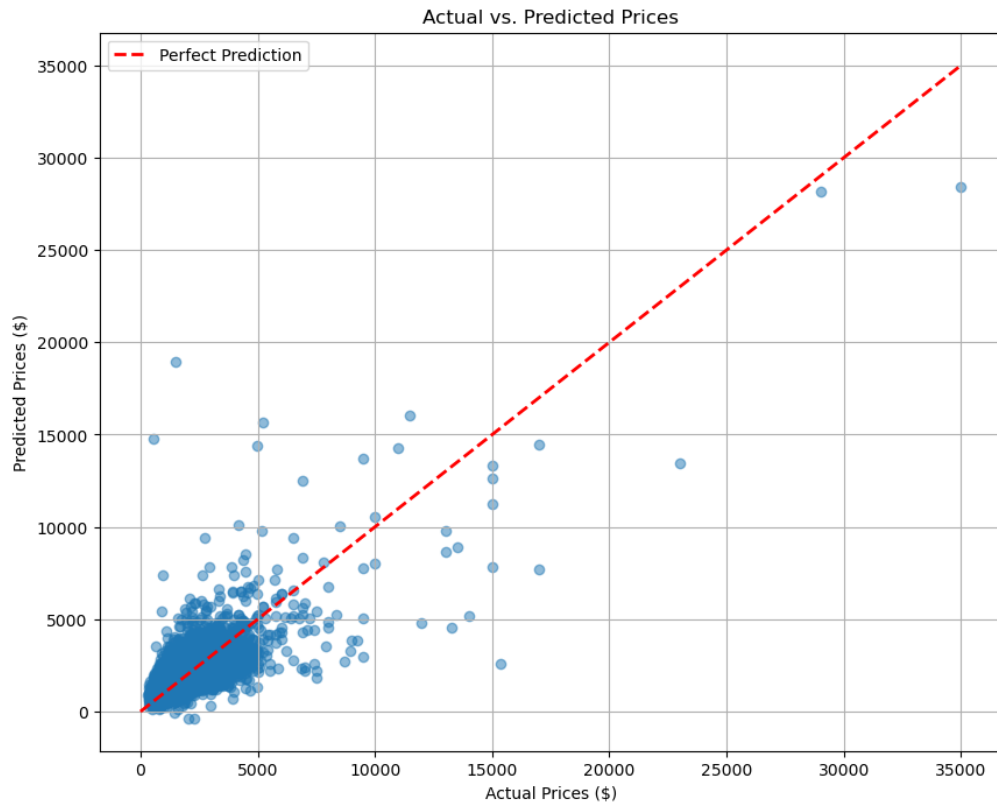


Fig. 2 Scatter plot of actual vs. predicted values

### 1.5.2. Distribution of Prediction Errors (Residuals)

This histogram displays the distribution of the residuals (Actual Price - Predicted Price). An ideal, unbiased model would show a normal distribution centered at zero. The generated plot is roughly bell-shaped and centered close to zero, which is a good sign.

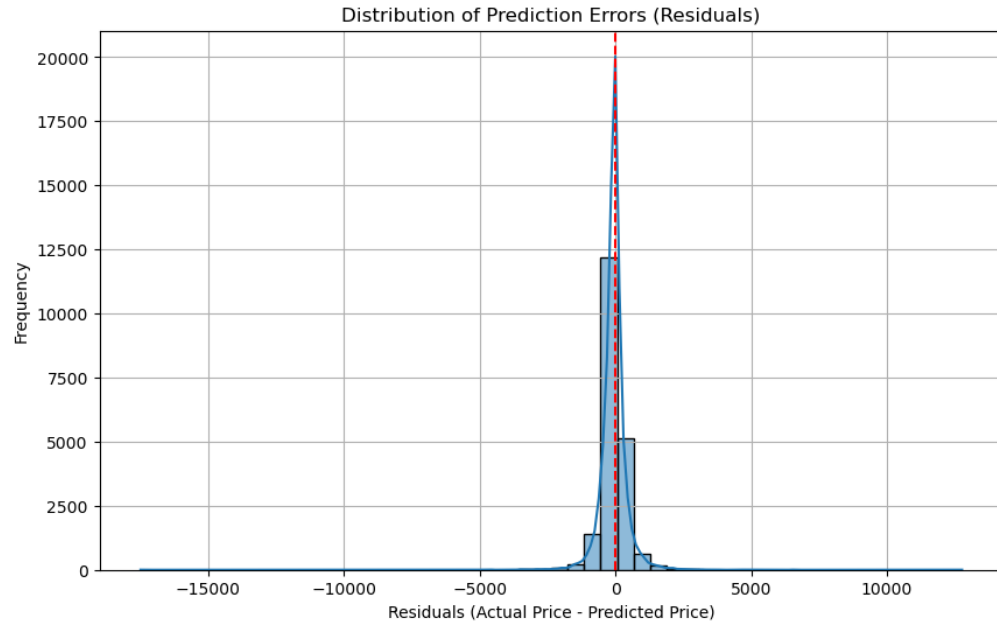


Fig. 3 Histogram of residuals

### 1.6. Conclusion and Comparative Insights

The Artificial Neural Network demonstrated a strong ability to predict real estate prices, explaining over 61% of the price variance. To provide a comprehensive analysis, its performance was compared against the results from Lab 1, which used K-Nearest Neighbors, Decision Tree, and Random Forest models on the same dataset.

### 1.6.1. Model Performance Comparison

The table below presents the key performance metrics for all four models on the test set.

**Table 2:** Key performance metrics

Model	Test	Test RMSE	Test $R^2$	Performance Notes
Random Forest	272.51	455.20	0.61	Best Performer (MAE/RMSE)
Artificial Neural Network	300.75	561.95	0.61	Excellent Performer
K-Nearest Neighbors	395.42	622.72	0.28	Moderate Performer
Decision Tree	368.15	802.28	-0.20	Poor (Overfit)

### 1.6.2. Performance Insights and Discussion

- **ANN vs. Random Forest:** Both the ANN and Random Forest models proved to be the top performers, with nearly identical  $R^2$  scores of 0.61. This indicates that both were equally effective at explaining the variance in the data. However, the Random Forest achieved a lower MAE (\$272.51 vs. \$300.75) and a significantly lower RMSE (\$455.20 vs. \$561.95).
  - **Reason:** As an ensemble method, Random Forest is inherently robust against overfitting and excels at handling high-dimensional data with many features, as was the case after one-hot encoding. The ANN, while powerful, likely requires further tuning (e.g., adding regularization like Dropout, or testing different architectures) to minimize its larger prediction errors and match the Random Forest's lower RMSE.
- **ANN vs. Decision Tree:** The ANN was vastly superior. The single Decision Tree model performed poorly, with a negative  $R^2$  score, confirming that it severely overfit the training data.
  - **Reason:** Single decision trees are prone to memorizing the training data, including its noise. The ANN, through its architectural design and optimization algorithm (Adam), was much better at generalizing the learned patterns to the unseen test set.
- **ANN vs. K-Nearest Neighbors (KNN):** The ANN also significantly outperformed the KNN model. The KNN model's  $R^2$  score of 0.28 shows it captured some patterns but was far less effective than the ANN or Random Forest.
  - **Reason:** KNN performance degrades in high-dimensional spaces (the "curse of dimensionality"). With 8,920 features, the concept of "nearness" becomes less meaningful, making it difficult for the model to find truly similar data points for making predictions. The ANN is better suited for identifying the most important features and learning complex relationships in such a space.

### **1.6.3. Conclusion**

While the developed ANN is a strong and effective model, the comparative analysis reveals that the Random Forest model from Lab 1 is the overall best performer for this specific task, primarily due to its lower average and large prediction errors (MAE and RMSE). The ANN's performance is on par with the Random Forest in terms of explanatory power ( $R^2$ ) and could potentially surpass it with additional hyperparameter tuning and regularization.