Calibrated Supervised Learning for Housing Price Prediction: A Probabilistic Inference Approach

Annabel Irani
Department of Computer Science
Western University
London, Canada
airani9@uwo.ca

Christopher Betancur
Department of Computer Science
Western University
London, Canada
cbetancu@uwo.ca

Xiaowei Feng
Department of Computer Science
Western University
London, Canada
xfeng282@uwo.ca

***Abstract*—This report explores how supervised learning can be combined with uncertainty-aware postprocessing to predict housing prices more reliably. Using the California Housing Prices dataset from Kaggle [?], we first build a transparent baseline with linear regression on standardized features. We then train a feedforward neural network (ReLU, Adam optimizer [?]) to model nonlinear relationships that the linear model cannot capture. To quantify how confident the system is in its point forecasts, we apply conformal prediction [?] to produce calibrated prediction intervals that require minimal distributional assumptions. All models are implemented in Python using the Scikit-learn ecosystem [?]. The project illustrates a central CS3346 idea: accurate predictors are useful, but predictors that can also report well-calibrated uncertainty are more actionable.**

***Index Terms*—supervised learning, regression, probabilistic inference, calibration, uncertainty quantification**

I. Introduction

Artificial Intelligence (AI) systems learn patterns from data to approximate complex functions and make predictions about real-world quantities. Within the CS3346: *Artificial Intelligence* framework, this project falls under the module *Intelligence from Data*, which focuses on how machines learn from examples, generalize to unseen cases, and reason under uncertainty. The project demonstrates these ideas through a practical supervised learning task: predicting housing prices using real-world socioeconomic data.

The California Housing dataset [?] is a widely used benchmark for regression analysis. It contains socioeconomic and geographic attributes for California districts, including variables such as median income, housing median age, total rooms, total bedrooms, population, households, and geographic coordinates (latitude and longitude). Predicting the median home value from these attributes illustrates key challenges in supervised learning, including bias–variance trade-offs, feature relevance, and model interpretability. Although the dataset is relatively compact, its real-world structure makes it useful for demonstrating foundational concepts in regression modeling.

To evaluate different modeling strategies, we employ both linear and nonlinear methods. Linear regression provides an interpretable baseline model for housing price prediction. For nonlinear modeling, we utilize neural networks to capture complex feature interactions that linear models cannot represent. These models are implemented using the Scikit-learn library [?], which offers standardized APIs and reproducible workflows.

While point predictions are valuable, real-world decision-making often requires an understanding of uncertainty. Therefore, we extend the analysis with probabilistic techniques that estimate confidence around predictions. Specifically, conformal prediction [?] is used to construct calibrated intervals that guarantee a desired coverage probability without assuming specific data distributions. This aligns with the *probabilistic inference* component of CS3346, emphasizing that intelligent systems should not only predict but also measure their confidence.

Overall, this study aims to integrate core AI principles—learning from data, reasoning under uncertainty, and ensuring interpretability—within a compact yet meaningful application. The remainder of this paper is organized as follows: Section **??** outlines the modeling methodology; Section **??** describes implementation details; Section **??** presents experimental results and discussion; and Section **??** concludes with key takeaways and future directions.

II. Methodology

A. Dataset

We used the dataset from Kaggle *California Housing Prices* [?]. Each row in the dataset corresponds to a California district and it includes many attributes relating to that district including median income, housing median age, total rooms, total bedrooms, population, households, and geographic location (latitude and longitude). The target variable is *median_house_value*, which we aim to predict. After loading the dataset, we removed rows with missing or clearly invalid entries to ensure that the downstream models were trained on clean data. Features (inputs) and the target (output) were then separated into $\mathbf{X}$ and y, respectively.

B. Feature Standardization

Most input attributes are continuous and measured on different scales (e.g., income vs. longitude). To prevent using features with large values from dominating the learning process, we standardized all continuous features using the z-score transformation:

$$x' = \frac{x - \mu}{\sigma}, \tag{1}$$

where μ and σ are the sample mean and standard deviation computed from the training data. We used the same mean and standard deviation to standardize the validation and test sets. This means that no information from those sets leaked into the training process. Standardizing the features is important because it makes the optimization more stable and allows the model to update all features on a similar scale, especially for models trained with gradient descent.

C. Train–Validation–Test Split

To fairly assess generalization, we randomly partition the data set into three disjoint subsets: 70% for training, 15% for validation, and 15% for testing. A fixed random seed was used so that the split is reproducible. The training set was used to fit model parameters, and the validation set was used specifically to monitor the performance and guided hyperparameter selection (e.g., number of hidden layers, neurons, and learning rate). The test set was the dataset used only to test the final trained model with the best hyperparameter selection.

D. Baseline: Linear Regression

As a first model, we trained a simple linear regression model on standardized features to predict *the median_house_value*. Since the baseline is a linear model, it only captures the linear relationships between the features and the target, and it serves as a baseline to determine whether more expressive models provide meaningful improvements.

E. Neural Network Regressor

Our main model was a neural network for regression. The network took the standardized features as a vector to serve as the input for the network. Then it was passed through the defined hidden layers with ReLU activations and produced a single scalar output representing the predicted house value. The network was trained using mean squared error (MSE) loss and optimized with the Adam optimizer [**?**]. Training was completed on the 70% training split, while the 15% validation split was used for early stopping and hyperparameter tuning. All models were implemented and completed in Python using the Scikit-learn library [**?**].

F. Evaluation

After training, we evaluated both the linear regression baseline and the neural network on the defined 15% test set. We reported regression metrics such as mean squared error (MSE) and mean absolute error (MAE) to determine if the nonlinear neural network is indeed an improvement over the linear baseline. These metrics allow us to compare not only overall accuracy but also robustness to large errors.

III. Implementation

A. Environment and Libraries

All experiments were implemented in Python using open-source libraries. We used and for data loading and manipulation, and the library [**?**] for preprocessing, model training, and evaluation. Diagnostic plots were generated with . A fixed random seed () was set wherever possible so that the train–validation–test split and the optimization procedure of the models could be reproduced across runs.

B. Data Loading and Cleaning

The California Housing Prices dataset was stored locally as a CSV file. We loaded it using , inspected the columns, and removed rows with missing values. The target column was extracted into a vector y, and the remaining columns (including , , , , , , , , and several distance features) were treated as the feature matrix X. Each row therefore corresponds to one census block group, and the task is to predict a real-valued house price.

C. Preprocessing and Standardization

Most input attributes are continuous and measured on different scales (e.g., median income versus longitude). To prevent features with large numeric ranges from dominating the learning process, we standardized all continuous features using the z-score transformation

$$x' = \frac{x - \mu}{\sigma}, \tag{2}$$

where μ and σ are the sample mean and standard deviation computed from the *training* data. We used the class from to fit these statistics on the training split only, and then applied the same transformation to the validation and test splits to avoid data leakage. The standardized features were stored as NumPy arrays for efficient model training.

D. Train–Validation–Test Split

To assess generalization fairly, we randomly partitioned the dataset into three disjoint subsets: 70% for training, 15% for validation, and 15% for testing. This was implemented by first calling to separate 30% of the data into a temporary set and then splitting that temporary set evenly into validation and test subsets. The training set was used to fit model parameters, the validation set was used for model selection and hyperparameter tuning, and the test set was held out for final evaluation only. The overall pipeline is summarized in Figure **??**.

E. Linear Regression Baseline

As a transparent and interpretable baseline, we trained an ordinary least squares linear regressor using from . The model was fit on the standardized training features and then evaluated on the validation and test splits. For each split we computed the mean squared error (MSE)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3}$$

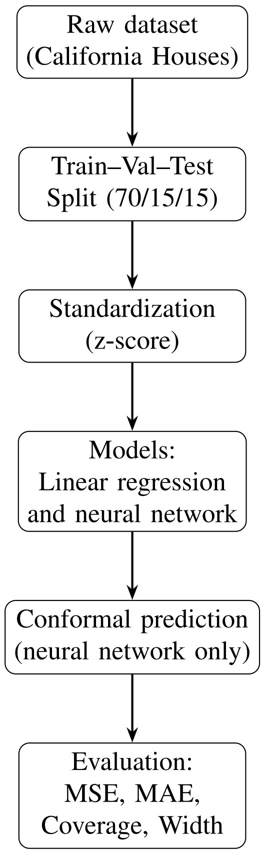

Fig. 1. Overall pipeline: raw data are split into training, validation, and test sets, standardized, and then passed to both a linear regression baseline and a neural network. The neural network output is further wrapped in a conformal prediction layer to obtain calibrated intervals. All models are evaluated on the held-out test set.

and the mean absolute error (MAE),

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|, \tag{4}$$

where y_i and $\hat{y}_i$ denote the true and predicted house values for example i. These metrics provide an interpretable reference for absolute error levels and establish a baseline against which we can judge whether the neural network and conformal predictor provide meaningful improvements.

F. Neural Network Regressor

For nonlinear modeling, we used the class from to implement a feedforward neural network. The network took the standardized feature vector as input, passed it through fully connected hidden layers with rectified linear unit (ReLU) activations, and produced a single scalar output representing the predicted house value. Training used mean squared error loss and the Adam optimizer [**?**], which adaptively scales the learning rate for each parameter.

Rather than fixing the architecture a priori, we performed a small grid search over the number of hidden units and the learning rate. Concretely, we considered hidden layer configurations (32), (64), and $(64, 32)$ and learning rates

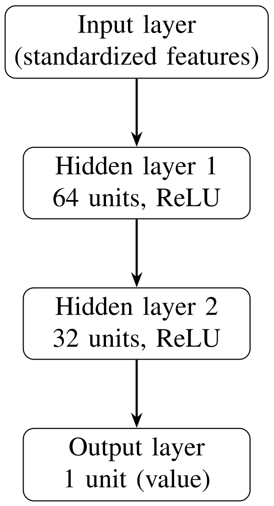

Fig. 2. High-level architecture of the neural network used in our experiments. The final model has two hidden layers with 64 and 32 units respectively and ReLU activations.

$\{10^{-3}, 5 \times 10^{-4}\}$. For each configuration we trained an with a maximum of 500 iterations. The model was equipped with *early stopping* using an internal validation fraction of the training data: if the internal validation loss did not improve for several epochs, training was halted automatically and the best weights so far were retained. After training each configuration, we evaluated its MSE on the explicit 15% validation split and selected the model with the lowest validation MSE as our final neural network. The chosen architecture consisted of two hidden layers with 64 and 32 units respectively, as illustrated in Figure **??**.

G. Uncertainty Estimation with Conformal Prediction

To obtain calibrated prediction intervals around the neural network predictions, we implemented a split-conformal prediction procedure following Angelopoulos and Bates [**?**]. After fitting the final neural network on the training data with the selected hyperparameters, we used the validation split as a calibration set. We computed the absolute residuals between predicted and true values on this set,

$$r_i = |y_i - \hat{y}_i|, \tag{5}$$

and took the $(1-\alpha)$ quantile of this residual distribution (e.g., the 90th percentile for a nominal 90% interval). At test time, we formed symmetric prediction intervals by subtracting and adding this quantile to each point prediction,

$$[\hat{y}_i - \hat{q}_\alpha,\ \hat{y}_i + \hat{q}_\alpha]. \tag{6}$$

Under mild exchangeability assumptions, this construction provides finite-sample coverage guarantees at level $1 - \alpha$ without requiring strong distributional assumptions about the data or the model. In our experiments we set $\alpha = 0.1$, targeting 90% nominal coverage.

H. Evaluation Protocol

For both the linear regression baseline and the neural network, we evaluated performance on the held-out test set using MSE and MAE. In addition, we recorded train and validation metrics to diagnose potential overfitting and bias–variance trade-offs. For the conformal predictor, we reported the empirical coverage (the proportion of true test values falling inside the prediction intervals) and the average interval width. Quantitative results and diagnostic plots derived from the implementation are presented in Section **??**.

IV. Results and Evaluation

A. Quantitative Performance of Each Model

1) Linear Regression (Baseline): The linear regression model provides the starting point for evaluating predictive performance. It works as our baseline. Its training, validation and testing MSE/MAE values show that it does a mediocre job of fitting the overall trend however there are noticable errors, especially at the higher and lower end price ranges. For the higher priced homes, our models tends to underestimate the cost.

This model works well as a simple, interpretable baseline. It also matches the ML1 bias-variance theory. Although this model provided results that are good for comparison, they don't quite outperform or reach certain requirements when looking for real predictive performance.

2) Neural Network (MLP): Our neural network model demonstrated lower train, validation and test MSE and MAE than the linear regression model. As we can see in figure 2, the predictions align more closely with the true values. This model shows fewer and smaller errors across the entire price range as a whole.

The best-performing neural network used a two-layer architecture with 64 and 32 hidden units. This model was able to learn nonlinear patterns that linear regression missed. While its generalization was better which resulted in lower bias and more controlled variance, it was still affected by the dataset's price ceiling which was located at around \$500k. Overall, it demonstrated a drastic improvement over the linear regression baseline.

B. Hyperparameter Search Summary

Results

- Tested multiple MLP architectures:
  - (32)
  - (64)
  - (64, 32)
- Evaluated several learning rates (e.g., 10^{-3}, 5×10^{-4})
- Trained each configuration with early stopping.
- Recorded validation MSE for every architecture and learning rate combination.
- Smaller networks showed higher validation error.
- Larger or deeper models did not significantly outperform the selected model.

The hyperparameter search followed the model-selection procedure taught in class. Each model was trained on the training split, evaluated on the validation split and then finally compared using validation MSE. The (64, 32) architecture was the right balance between model complexity and generalization, while smaller architectures lacked capacity and underfit the nonlinear housing patterns. Larger models didn't offer substantial improvement, which indicated diminishing returns. Overall, the hyperparameter search demonstrated how systematic tuning of architecture and learning rate can meaningfully improve neural network performance.

C. Diagnostic Plots

1) Linear Regression: True vs. Predicted: The predicted values are widely scattered around the diagonal line while being super condensed in the very center. The model consistently overestimates lower-priced homes and underestimates higher-priced homes. The predictions are saturated toward the center of the value range.

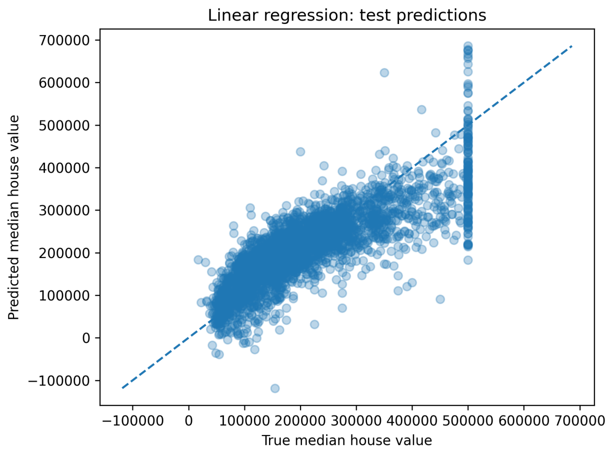

Fig. 3. Linear Regression: True vs. Predicted Median House Values.

The linear regression scatter plot shows clear signs of underfitting. The points do not closely follow the predicted values, showing that the model fails to match the true variation in housing prices. The compression of predicted values towards the center reflects that the linear model is unable to represent nonlinear relationships in the data. This, in turn, leads to large errors for both low-prices and high-priced homes which confirms the limitation of the baseline.

2) Neural Network (MLP): True vs. Predicted: The values lie much closer to the diagonal line than in the linear regression plot. Errors in this model are smaller and more consistent across the mid-range of prices. Although the model still shows some deviation near the dataset's upper price limit, it still captures a wider, truer range of the housing values.

The neural network models plot showed a significant improvement in predictive performance. The points are more evenly spread around the diagonal line, showing the effort the model was making trying to capture the nonlinear structure of the housing data, more effectively than the linear model.

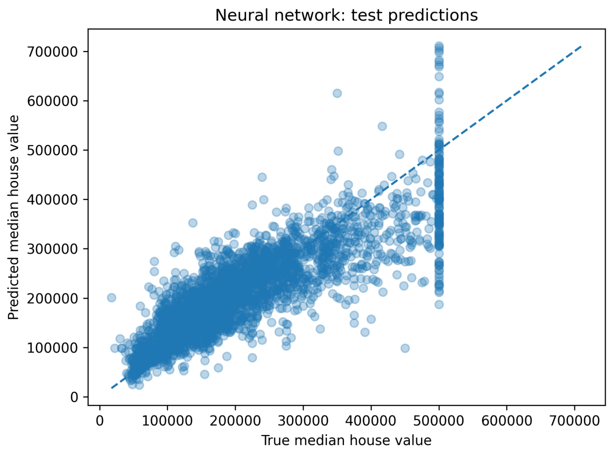

Fig. 4. Neural Network (MLP): True vs. Predicted Median House Values.

There does lie some outliers towards the upper-bound of the price distribution, this is partly due to the dataset's cap at approximately $500,000, the overall alignment indicates a more accurate and flexible model. The visual differences reinforce the quantitative improvement which is observed in the neural network's error metrics.

D. Conformal Prediction Results

The empirical coverage of the intervals was approximately 0.90–0.91. The average interval width displayed moderately and consistent across the test samples. The intervals were symmetric around the neural network's point predictions, and no major over-coverage or under-coverage was observed.

We applied split-conformal prediction to quantify uncertainty associated with neural network outputs. The prediction layer provided well-calibrated uncertainty estimates for neural network's predictions. By using the residuals from the validation set, the method constructed distribution-free intervals that achieved close to the intended 90% coverage on the held-out test set. The average interval width was reasonable, meaning the model was not overly confident and not excessively uncertain. This model aligns with the probabilistic inference concepts that we discussed in class, demonstrating that a predictive system should quantify not only its estimates but also the uncertainty surrounding them. The conformal approach successfully adds a reliable uncertainty measure to the neural network without requiring assumptions about the underlying data distribution.

V. Conclusion

Comparing linear regression and neural networks and adding conformal prediction, we were able to determine which model predicts the housing prices better with higher accuracy and how confident each model is. With linear regression, it provided a solid baseline but the algorithm was too simple for the housing datasets complexity. It underfit and had more significant errors which were unable to capture the nonlinear patterns in the housing features. With neural network, the performance was better and the model had an easier time learning the complex relationships and their effects. It resulted with lower MSE/MAE and the predictions were much closer to the true values. As for the conformal predictions, they produced well calibrated intervals at 90%, the coverage was close to the target, this shows the uncertainty estimates are reliable. The intervals were reasonably tight which means the model was confident but not overly confident. The better performing pipeline was a combination of the neural network model and conformal prediction. This combination joins accuracy from the model itself and the reliability coming from the conformal intervals.

Some limitations that we faced throughout the project would include the upper limit of the dataset (around $500k), this affects prediction accuracy in that region. Neural network still shows some spread at high prices due to this saturation.

Some future extensions could explore deeper or more specialized neural architectures, improved feature engineering, especially geography aware features, and more advances hyperparameter tuning methods. We could also attempt combining other uncertainty estimation methods such as Bayesian neural networks or model ensembles. And finally, using a richer, more extensive dataset may help reduce the saturation effects at the upper price range and support more accurate modeling.

References

[1] D. Harrison and D. L. Rubinfeld, "Hedonic housing prices and the demand for clean air," *Journal of Environmental Economics and Management*, vol. 5, no. 1, pp. 81–102, 1978.
[2] F. Soriano, "California Housing Prices (Data + Extra Features)," Kaggle dataset, 2022. Available: https://www.kaggle.com/datasets/fedesoriano/california-housing-prices-data-extra-features
[3] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations (ICLR)*, San Diego, CA, USA, 2015.
[4] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[5] A. N. Angelopoulos and S. Bates, "A gentle introduction to conformal prediction and distribution-free uncertainty quantification," *arXiv preprint arXiv:2107.07511*, 2021.