Carter Prince

Project 2: Neural Networks (ScalaTion)

Predicting Home Prices in King County, WA

Introduction

This project implements a neural network regression model to predict house prices in King County, Washington using the ScalaTion framework. The goal was to develop a multi-layer perceptron that could accurately estimate home values based on various property characteristics and geographic location. As an exploration of neural network architectures using functional programming principles, the project involved comprehensive hyperparameter tuning and feature engineering to identify optimal model configurations.

The dataset contained over 21,000 house sales with 18 numerical features including bedrooms, bathrooms, square footage, location coordinates, and property condition ratings. A key enhancement was the one-hot encoding of zipcodes, which added 70 binary features representing distinct geographic areas within King County. This categorical encoding allowed the model to learn location-specific pricing patterns beyond what latitude and longitude coordinates alone could capture. An additional preprocessing step involved applying a logarithmic transformation to the target variable (house prices) to address the wide range of values and improve model convergence. A randomized hyperparameter search was conducted across 15 sampled configurations to evaluate the impact of network depth, width, activation functions, learning rates, and batch sizes on prediction accuracy.

Dataset

The King County house sales dataset comprises 21,613 residential property transactions with prices ranging from $75,000 to $7,700,000. The dataset includes 18 numerical features describing each property: the number of bedrooms and bathrooms, living space and lot size in square feet, number of floors, waterfront status, view quality, overall condition rating, grade classification, above-ground and basement square footage, year built, year renovated (if applicable), geographic coordinates (latitude and longitude), and average living space and lot size of the 15 nearest neighbors. The date field was included in the original data but was used primarily for record-keeping rather than as a predictive feature.

A critical enhancement to the feature set was the one-hot encoding of the zipcode variable. Rather than treating zipcode as a numerical value, each of the 70 unique zipcodes in the dataset was converted into a separate binary indicator feature. This categorical encoding

enables the model to learn distinct price characteristics for each neighborhood while avoiding the false assumption that zipcodes have an ordinal relationship. Combined with the 18 numerical features, the final feature space consisted of 88 total features (18 numerical plus 70 zipcode indicators). The data was split into training (80%, 17,290 samples) and test (20%, 4,323 samples) sets to ensure reproducibility.

Prior to model training, the target variable underwent logarithmic transformation to compress the wide range of house prices and improve numerical stability during optimization. The original price range of $75,000 to $7,700,000 was transformed to a log-scale range of approximately 11.23 to 15.86. The numerical features were then normalized using standardization, which transforms each variable to have zero mean and unit variance. This preprocessing step is critical for neural network training as it ensures all features contribute proportionally to the learning process and helps gradient descent converge more efficiently. The zipcode one-hot encoded features were left as binary indicators (0 or 1) and concatenated with the scaled numerical features.

Methods

The neural network architecture consisted of a fully-connected feedforward network implemented in ScalaTion with a configurable number of hidden layers and neurons per layer. Each hidden layer employed an activation function to introduce nonlinearity into the model. The network architecture was parameterized to explore different model capacities during hyperparameter tuning. Three activation functions were evaluated: LeakyReLU (lreLU), hyperbolic tangent (tanh), and sigmoid, to determine which best suited this regression task. The final layer produced a single continuous output representing the predicted log-transformed house price, which was then exponentiated back to the original dollar scale for evaluation.

Hyperparameter optimization was performed through randomized search over a parameter space with 15 sampled trials. The search explored hidden layer widths of 16, 32, and 64 neurons; network depths of 1, 2, and 3 hidden layers; activation functions including LeakyReLU, tanh, and sigmoid; learning rates of 0.00001, 0.00005, 0.0001, and 0.0005; and batch sizes of 64 and 128. Each configuration was trained for 5 epochs during the screening phase, with performance measured by root mean squared error (RMSE) on the test set in log-scale. Results were logged and compared to identify the optimal configuration.

Analysis of the 15 trials revealed clear performance patterns. Networks using LeakyReLU activation substantially outperformed those using tanh or sigmoid functions. The best performing configuration achieved a test RMSE of 0.3458 on the log-scale and featured 16 neurons per hidden layer across 2 hidden layers, LeakyReLU activation, a learning rate of 0.0005, and a batch size of 64. This moderate-sized architecture struck an effective balance

between model capacity and generalization. Following hyperparameter selection, a final model was trained for 50 epochs using the optimal configuration to allow sufficient time for convergence.

Results

The hyperparameter search identified important architectural patterns for this regression task. LeakyReLU activation proved superior to both tanh and sigmoid functions, likely due to its ability to avoid vanishing gradients while maintaining computational efficiency. Network size demonstrated diminishing returns beyond moderate architectures, with the 16-neuron, 2-layer configuration outperforming both simpler single-layer networks and more complex 3-layer alternatives. Learning rate selection was critical, with the moderate value of 0.0005 providing stable convergence without the slow progress of smaller rates or the instability of larger ones. The smaller batch size of 64 was selected over 128, potentially offering better gradient estimates through more frequent updates.

The final model, trained for 50 epochs with the best hyperparameters, achieved strong predictive performance on the housing dataset. On the training set, the model attained an RMSE of $183,345.37, a mean absolute error (MAE) of $92,355.66, and an R^2 coefficient of determination of 0.755. On the held-out test set, performance was actually slightly better with an RMSE of $163,484.16, MAE of $95,481.31, and R^2 of 0.786. The fact that test performance exceeded training performance suggests the model generalized well to unseen data without overfitting, though the modest R^2 values indicate substantial room for improvement in capturing price variance.

Feature importance analysis revealed which variables the model weighted most heavily in making predictions. The top features included floors (importance: 4.73), longitude (4.59), square footage of living space (4.58), number of bathrooms (4.31), waterfront status (4.30), view quality (4.28), grade classification (4.26), basement square footage (4.21), number of bedrooms (4.18), lot size (4.16), above-ground square footage (4.15), year renovated (4.07), year built (4.03), neighbor living space average (3.95), and latitude (3.93). This ranking indicates that structural characteristics like floors and living space, geographic location (longitude and latitude), and quality indicators (grade, view, waterfront) were the most influential predictors.

Conclusion

This project successfully implemented a neural network regression model for house price prediction using the ScalaTion framework, demonstrating the complete machine learning workflow including feature engineering, data preprocessing, model architecture design, hyperparameter optimization, and rigorous performance evaluation. The addition of one-hot encoded zipcode features expanded the feature space to 88 dimensions and allowed the

model to capture neighborhood-specific pricing patterns. The application of logarithmic transformation to the target variable proved essential for handling the wide range of house prices spanning two orders of magnitude.

The final model achieved moderate predictive accuracy with a test RMSE of $163,484 and R^2 of 0.786, indicating the model explains approximately 79% of the variance in house prices. While this demonstrates meaningful predictive capability, the substantial absolute errors on some predictions suggest limitations in the model's ability to handle outlier properties or capture complex nonlinear interactions between features. The hyperparameter search across 15 configurations revealed that architectural choices matter significantly: LeakyReLU activation outperformed alternative functions, moderate network sizes (16 neurons, 2 layers) proved optimal, and careful learning rate selection was critical for achieving good convergence.

The ScalaTion framework provided a functional programming approach to neural network implementation, offering strong type safety and mathematical abstractions well-suited to numerical computing tasks. Feature importance analysis highlighted that location (longitude/latitude), structural characteristics (floors, living space), and quality indicators (grade, view, waterfront) were the most influential predictors, validating intuitions about real estate valuation. The close alignment between training (R^2 = 0.755) and test (R^2 = 0.786) performance demonstrates that the model generalized adequately without significant overfitting.

As a first neural network project using ScalaTion, this work provided valuable hands-on experience with functional implementations of machine learning algorithms, the importance of target variable transformation for regression tasks, feature engineering with categorical variables, and the practical challenges of hyperparameter tuning. The results demonstrate that thoughtful preprocessing (log transformation and zipcode encoding) combined with systematic architecture selection can yield functional predictive models, though further improvements through ensemble methods, additional feature engineering, or alternative architectures could potentially reduce prediction errors, particularly for unusual properties.

Supplementary Materials

All training code is available on GitHub at
https://github.com/carterprince/king-county-homes.

Note: The relevant Scala source file `kingCountyHousePriceModel.scala` must be placed in the `modeling/neuralnet` directory in ScalaTion's source tree.

The original dataset is available on Kaggle at
https://www.kaggle.com/datasets/harlfoxem/housesalesprediction.