Neural Networks Report

Jamie Todd, Savraj Sian, Ifte Chowdhury, Matt Neave

Description of the model

In pre-processing of the input and output, various choices were made regarding normalisation and missing value replacement. Various scalers were tested for normalisation of the training data using a multi-layered neural network; the results are presented below.

Scaler applied to x (input)	**Scaler applied to y (output labels)**	**Root Mean Squared Error (RMSE)**
Standard	Standard	46972.4
Min-Max	Standard	57699.7
Standard	Min-Max	50015.4
Min-Max	Min-Max	57598.5
Standard	Max-Abs	50326.4
Max-Abs	Standard	61430.2

The *Standard* scalar scales the data to have 0 mean and unit variance, *Min-Max* scales the data to be between 0 and 1, and *Max-Abs* scales such that the maximum absolute value of each feature is set to 1. Applying the *Standard* scaler to both x and y yielded the lowest RMSE, and so pre-processing uses the *Standard* scaler for both the input and output labels.

To fill in missing values of the data, the mean and median of each column was tried, as well as *IterativeImputer* from the *SciPy* library. The 'ocean_proximity' attribute uses the mode (i.e. the most common text) since this is a string. The RMSE for the metrics are listed below.

Metric	**RMSE**
Mean	52458.3
Median	52689.5
IterativeImputer	52776.7

The mean gave the lowest RMSE, so the pre-processing uses this metric.

Our model consists of a linear input layer, two hidden layers, and a linear output layer, with a ReLU activation function within the hidden layers (see "Final evaluation of the best model" for more information on this). The ReLU activation function tends to converge better than sigmoid-like functions and is less computationally expensive. This follows Occam's Razor, where the model shouldn't be overcomplicated when a relatively simple model is sufficient and achieves high accuracy.

The optimiser chosen was *SGD* (stochastic gradient descent) since whilst it is more sensitive to hyperparameter tuning compared to other options like *Adam* (adaptive moment estimation), which utilises adaptive learning rates for parameters and tends to converge faster, *SGD* generalises better which is advantageous when presented with unseen data.

Batching is utilised to decrease training time and reduce noise, as can be seen in mini-batched gradient descent where it also prevents overfitting on individual data points. Shuffled indices are used in an effort to prevent overfitting; it also allows for more data to be used for training and hyperparameter tuning, which will result in a better trained model. Early stopping is used in the training of the model once the optimal hyperparameters have been found, as it will result in better generalisation whilst also preventing overfitting.

Description of the evaluation setup

To evaluate the model, a separate held-out dataset is used for testing with an 80/20 split for training/testing data. Since indices are shuffled, the hyperparameter tuning and evaluation of the model use different parts of the dataset.

The evaluation metrics chosen are RMSE and R2 score since they provide a good representation of how close the model's predictions are to the actual house price, however as detailed later, RMSE is the primary metric used. RMSE focuses on the distance between the predicted and correct house price in a geometric sense whereas R2 score gives an insight into how correlated the data is; the higher the score, the more correlated the predictions and labels are. By using these metrics, a more complete evaluation of the model can take place.

A minimum baseline which we established for the RMSE is the case when the mean of the median house value is 'guessed', regardless of the input, since a model doing this is the laziest form of learning; the mean is 207251.9 and the RMSE that results from using this value for all predictions for the training dataset is 115683.5.

Hyperparameter search and findings

For hyperparameter search, we used a neural network with one hidden layer for our model. We decided to tune the number of layers after the other three hyperparameters since this greatly reduced training time and returned satisfactory results.

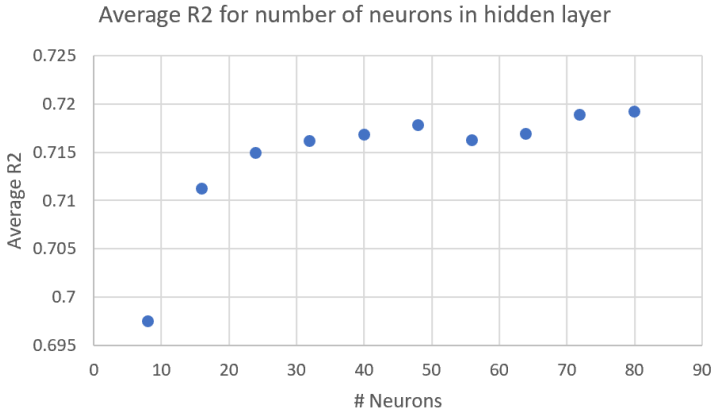
We tuned three different hyperparameters for the model:

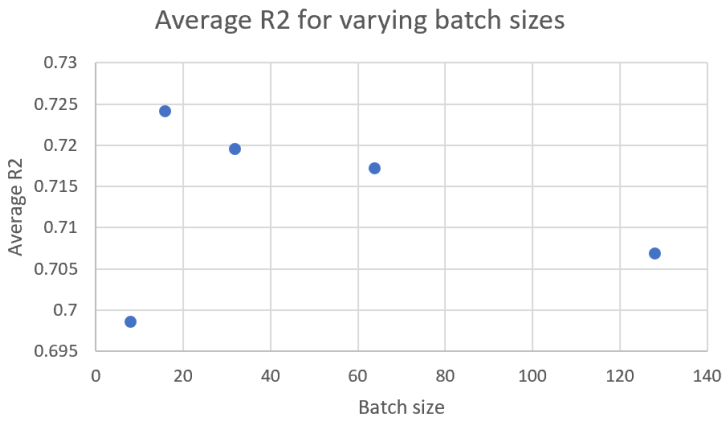
- Learning rate, ranging from 0.002 to 0.1, in steps of 0.002
- Batch size, ranging from 8 to 128, in steps of powers of 2 (i.e. 8, 16, 32, 64, 128)
- Number of neurons in the hidden layer, ranging from 8 to 80, in steps of 8

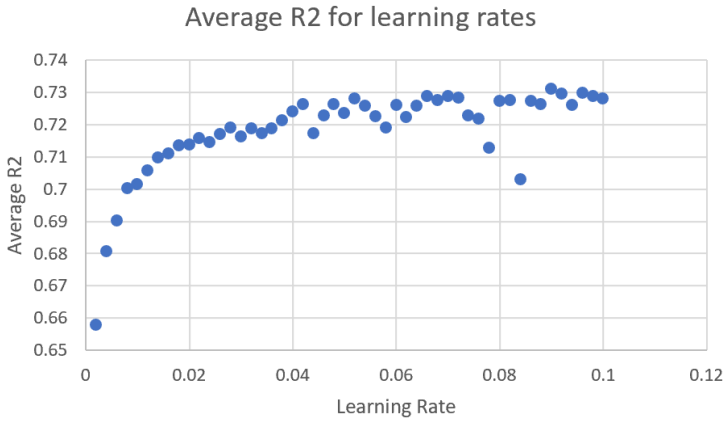
Each model was trained over 10 epochs for an efficient search, using RMSE as the metric for determining the optimal hyperparameters.

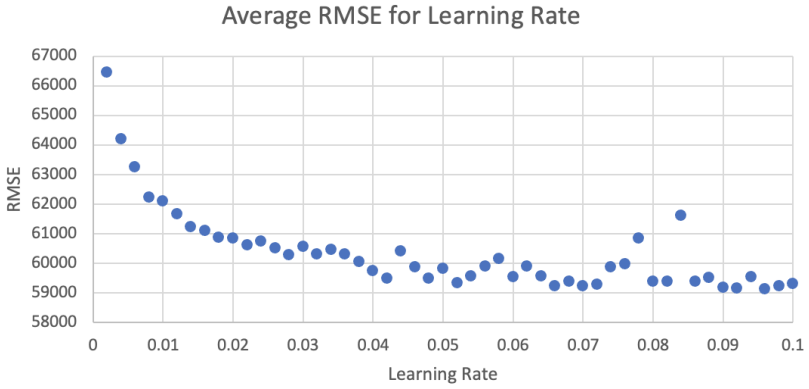
We chose RMSE as the sole metric for hyperparameter tuning as it is easy to understand the error, whilst giving a clear indication of how the model will perform on unseen data. On the other hand, R2 score is complex to interpret and does not give an obvious indication on how the model would perform on unseen data. Overall, 2500 different models were tested.

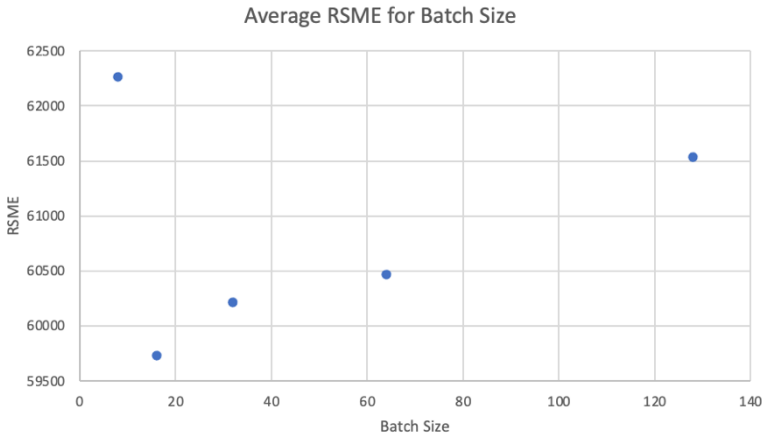
Some graphs of our results are shown below:

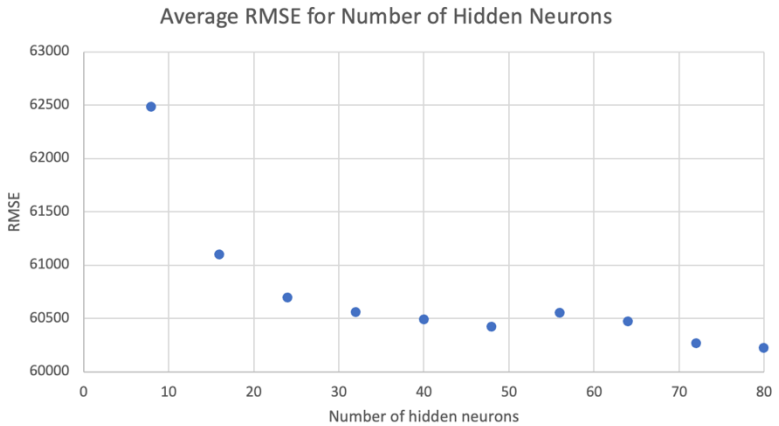












These graphs were generated by taking the average R2 and RMSE for each hyperparameter being optimised as every combination of the hyperparameters was tested. They give a general trend for the correlation between the metric and hyperparameter being changed, however as explained in the next section, they do not give a complete picture of what the optimal values are. From these graphs a higher learning rate, batch size of around 16 and a higher number of neurons in the hidden layer yield better results with a lower RMSE and higher R2 score.

The full spreadsheet of our results can be found in the code repository.

Evaluation of the tuned model

From our hyperparameter tuning results, the model which achieved the lowest RMSE on the test data had the following tuned hyperparameters:

- Learning Rate: 0.014
- Batch Size: 16
- Number of neurons in hidden layer: 64

During hyperparameter tuning, when trained over 10 epochs, the RMSE achieved from this model was 56657.2, and the R2 score was 0.752.

This model was not significantly better than the rest; 9 models in total achieved an RMSE < 57000, and one other model achieved a R2 score > 0.75, though this was still less than our optimal model's score.

From our hyperparameter tuning results and graphs, it may seem like the optimal model would have a much higher learning rate, closer to 0.1 as opposed to 0.01, however these results may be misleading as a lot of high learning rate models affect the weights by too much, leading to predictions being extremely high numbers, which were excluded from the data.

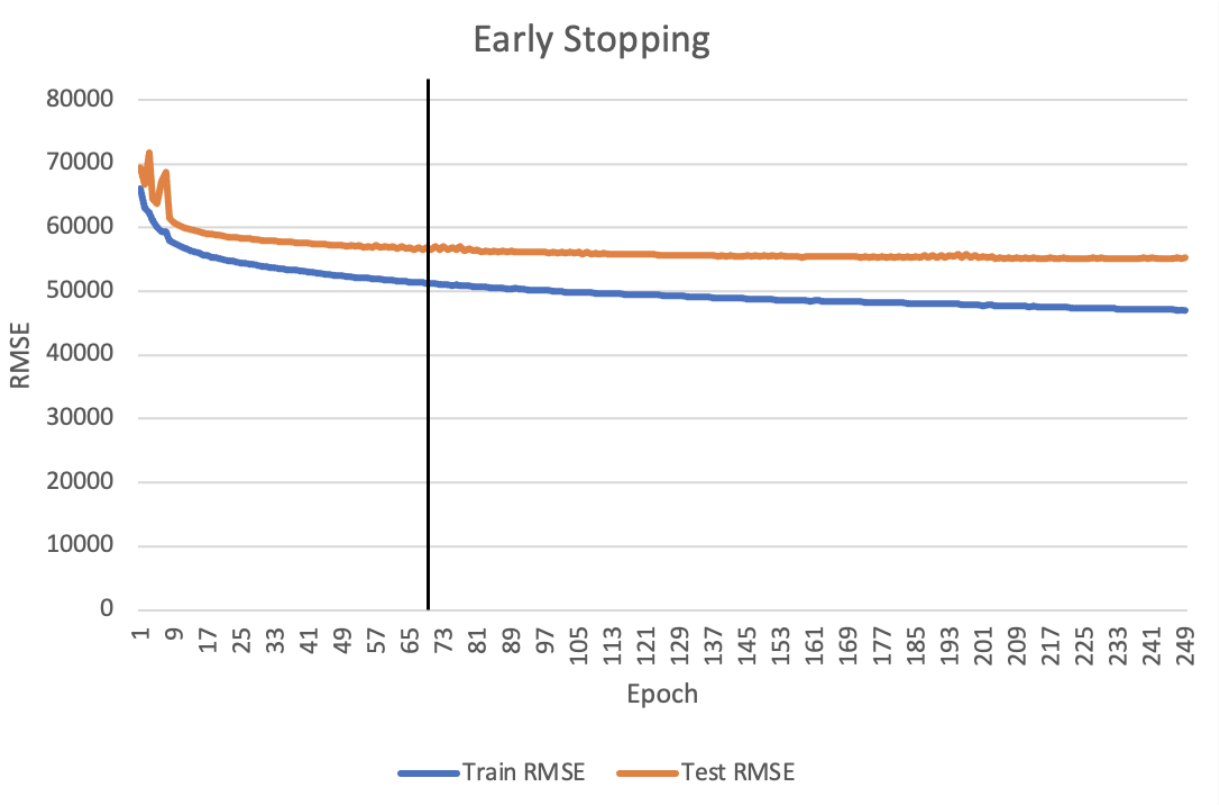
Further improvements & final evaluation of the best model

Our first step to further improving this model was to experiment an additional hidden layer to see if this achieves an even lower RMSE. We did a similar approach to hyperparameter tuning where we ranged the number of neurons per hidden layer between 8 and 80 in steps of 8 for both hidden layers, leading to 10 * 10 = 100 different models being tested.

From this experiment, we found that 56 neurons in the first hidden layer, and 64 neurons in the second hidden layer achieved the lowest RMSE, with a score of 56481.7. This is an improvement from our original model found from hyperparameter search.

Despite hyperparameter tuning returning the best set of parameters with the lowest error, these parameters were not always reliable and sometimes caused the error to diverge to infinity. As a result, we adjusted the learning rate to the largest value that did not cause the model to encounter invalid values based on the CSV output from the hyperparameter search. This value is 0.006.

Our final step to further improving this model was to implement a technique to prevent overfitting known as *early stopping*. Our implementation of early stopping works as follows: After every epoch, the RMSE is calculated for the updated model, and if the RMSE is greater (worse) than the previous 10 epochs, then the model stops training, and the current state of the model is what's used for predictions.



The above graph shows the behaviour of the RMSE for the training and test data used for the early stopping, with the black line indicating the epoch in which training terminated. This is useful as the RMSE for the training data decreased beyond this point, however the RMSE for the test data remained relatively constant, indicating that the model would have continued to overfit to the training data and therefore perform worse on unseen data.

With early stopping applied, after training on the whole dataset, our final model achieved an RMSE of 49037.9.