King County House Prices Prediction with Artificial Neural Networks

Dong Liu
dliu9@nd.edu
University of Notre Dame
Notre Dame, Indiana, USA

Zhongrui Zhang
zzhang39@nd.edu
University of Notre Dame
Notre Dame, Indiana, USA

Yanjie Chen
ychen48@nd.edu
University of Notre Dame
Notre Dame, Indiana, USA

Chenyu Guo
cguo2@nd.edu
University of Notre Dame
Notre Dame, Indiana, USA

ABSTRACT

This project aims to develop Artificial Neural Network (ANN) models for predicting house prices in King County, by leveraging the features provided in the dataset. Besides, It was expected that the ANN model would provide accurate predictions and insights into the significant factors affecting house prices.

ACM Reference Format:
Dong Liu, Zhongrui Zhang, Yanjie Chen, and Chenyu Guo. 2023. King County House Prices Prediction with Artificial Neural Networks. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

1.1 Problem Definition

In fact, the house market is profitable and interesting to many parties. Hence, house prices forecasting has drawn much attention to different individuals and organizations. Recently, different machine learning techniques were applied to house prices problems. ANN is chosen as it has good result in various forecasting problems. For instance, Jaen in 2002 proved that neural networks have high prediction accuracy (96%) in simulations [1]. Therefore, ANN models are considered to evaluate the performance on the King County house prices dataset.

1.2 Data Sources

The data that will be used for this project was collected in a study done by Shiva Chandel. The data is accessed from Kaggle: https://www.kaggle.com/datasets/shivachandel/kc-house-data. This dataset contains houses sold between May 2014 to May 2015 in King County, an area in the US state Washington. The dataset has 21 attributes and the outcome variable "price", representing the sold price of the given property. The dataset attributes are as follows:

(1) **id**: Unique identified for a house.
(2) **date**: House was sold.
(3) **price**: The sold price of the given property, which is the prediction target.
(4) **bedrooms**: Number of bedrooms.
(5) **bathrooms**: Number of bedroom.
(6) **sqft_living**: Footage of the living.
(7) **sqft_lot**: Footage of the lot.
(8) **floors**: Floors(levels) in house.
(9) **waterfront**: House which has a view of a waterfront.
(10) **view**: Has been viewed.
(11) **condition**: How good the condition is (overall).
(12) **grade**: King County's grading scale is used to determine the dwelling unit's total grade.
(13) **sqft_above**: Square footage of house apart from basement.
(14) **sqft basement**: Square footage of basement.
(15) **yr_built**: Built Year.
(16) **yr_renovated**: Year of renovation of the house.
(17) **zipcode**: Zip code.
(18) **lat**: Latitude coordinate.
(19) **long**: Longitude coordinate.
(20) **sqft_living15**: The area of the interior where the 15 closest neighbors' dwelling quarters are located.
(21) **sqft_lot15**: The size of the 15 closest neighbors' nearest property parcels.

In total, there were **21,613 data points**.

2 DATA VISUALIZATION AND PRE-PROCESSING

Upon downloading the data set, we performed several exploratory data analysis (EDA) tasks, and visualized different dimensions of the data for better intuition. Then, we scrutinized through all the entries and features, hunting for missing values, noisy values, or inconsistent values. After preliminary data cleaning, we moved on to extract features of interest, normalizing the features, as well as encoding the data frame to be used in machine learning models.

2.1 EDA Visualization

In order to develop a better understanding of the data set we are going to deal with, we performed several data visualization tasks with matplotlib.pyplot between different features of interest. The visualizations we made are the follows:

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

(1) A summary statistics of all variables is given in Figure 1. Besides, it was checked that there is no missing value contained in this dataset.

	id	price	bedrooms	bathrooms	sqft_living
count	2.161300e+04	2.161300e+04	21613.000000	21613.000000	21613.000000
mean	4.580302e+09	5.400881e+05	3.370842	2.114757	2079.899736
std	2.876566e+09	3.671272e+05	0.930062	0.770163	918.440897
min	1.000102e+06	7.500000e+04	0.000000	0.000000	290.000000
25%	2.123049e+09	3.219500e+05	3.000000	1.750000	1427.000000
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000

	sqft_lot	floors	waterfront	view	condition
count	2.161300e+04	21613.000000	21613.000000	21613.000000	21613.000000
mean	1.510697e+04	1.494309	0.007542	0.234303	3.409430
std	4.142051e+04	0.539989	0.086517	0.766318	0.650743
min	5.200000e+02	1.000000	0.000000	0.000000	1.000000
25%	5.040000e+03	1.000000	0.000000	0.000000	3.000000
50%	7.618000e+03	1.500000	0.000000	0.000000	3.000000
75%	1.068800e+04	2.000000	0.000000	0.000000	4.000000
max	1.651359e+06	3.500000	1.000000	4.000000	5.000000

	grade	sqft_above	sqft_basement	yr_built	yr_renovated
count	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000
mean	7.656873	1788.390691	291.509045	1971.005136	84.402258
std	1.175459	828.090978	442.575043	29.373411	401.679240
min	1.000000	290.000000	0.000000	1900.000000	0.000000
25%	7.000000	1190.000000	0.000000	1951.000000	0.000000
50%	7.000000	1560.000000	0.000000	1975.000000	0.000000
75%	8.000000	2210.000000	560.000000	1997.000000	0.000000
max	13.000000	9410.000000	4820.000000	2015.000000	2015.000000

	zipcode	lat	long	sqft_living15	sqft_lot15
count	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000
mean	98077.939805	47.560051	-122.213898	1986.552492	12768.455652
std	53.505026	0.138564	0.140828	685.391304	27304.179631
min	98001.000000	47.155899	-122.518997	399.000000	651.000000
25%	98033.000000	47.471001	-122.328003	1490.000000	5100.000000
50%	98065.000000	47.571800	-122.230003	1840.000000	7620.000000
75%	98118.000000	47.678001	-122.125000	2360.000000	10083.000000
max	98199.000000	47.777599	-121.315002	6210.000000	871200.000000

Figure 1: Summary Statistics of All Variables

(2) A histogram of target variable "price" is given in Figure 2. It was shown that it is positively skewed, which means that most houses have lower prices with a few high-priced outliers. Hence, normalization of the target variable is needed.

2.2 Data Cleaning

After scrutinizing the data set, we reached the conclusion that the dataset is already pretty clean for further processing. No missing data, noisy data, or inconsistent data issue was observed, so we moved on to data pre-processing rather quickly. At the end of this stage, our data set contains 21,613 rows (objects) and 21 columns (features).

2.3 Feature Correlation

A heatmap of correlations with all variables is given in Figure 3. It was found that variables "bathrooms", "sqft living", "grade", "sqft above", "sqrt living15" have correlations larger than 0.5 with target variable "price". Among such variables, "sqft living" has the strongest correlation. Scatter plots of house prices versus these 5 variables are given in Figure 4.

2.4 Data Pre-processing

We performed several tasks in the data pre-processing stage so the resulting data frame can be directed fed into different machine learning models. Our tasks involve the following:

(1) Selecting features of interest from the raw data frame, we dropped the columns "id" and "date", which are the two categorical features that we are not interested in.
(2) Normalizing feature columns. Feature scaling through normalizing all the features is applied on the training set and testing set separately, using

$$x = \frac{x - mean}{standard\ deviation}$$

. This way, we ensured consistent feature scales, essential for model performance.
(3) Implementing the min-max normalization method by using the function MinMaxScaler() to rescale our target variable house prices for both training and testing dataset and to normalize all of our feature columns in order to guarantee that the range of each variable is between 0 and 1. Since all of the variables are numerical features, the min-max normalization equation is defined as follows:

$$f(x) = \frac{x - min(X)}{max(X) - min(X)}$$

, where X is a feature column and $x \in X$. The min-max normalization method leads to improved model performance and convergence during training.
(4) Transforming the target variable "price" into float32 and reshaping to work with the MSE loss. Therefore, the transformed dataset is suitable for modeling. Considering the volume of our dataset, the dataset with 19 variables is separated into a training set and testing set. In result, 80% samples (17,290 observations) are used to train further models. Trained models are then compared performance on the left 20% samples (4,323 observations).

3 SOLUTIONS AND METHODS

As we have preprocessed and cleaned data, we are now ready to apply different machine learning models to the dataset and gauge performances. We trained five neural network models: dense layer, RNN, BiRNN, LSTM, and Seq2Seq. The five models are implemented with 100 numbers of epochs in total. The architecture of each model will be presented in order.

3.1 Baseline Approach

Our baseline model is the linear regression of a full model predicting house prices based on all 19 variables. It has an R-squared value 0.70. Our goal is to, via implementing other models, reduce the loss generated by the baseline model.

3.2 Machine Learning Approaches

3.2.1 Dense Layer. A dense layer model initialize neural networks with two single layer feed forward networks. One "fc1" is with 18 inputs and 64 output and another "fc2" with 64 inputs and 18 output. The model then defines a forward method such that first pass data through a rectified-linear activation function over fc1(x) and next pass it into fc(2). The model sets loss function as the MSE loss and sets gradient descent optimizer with learning rate 0.001. In each epoch in the model, weights are updated with forward pass and

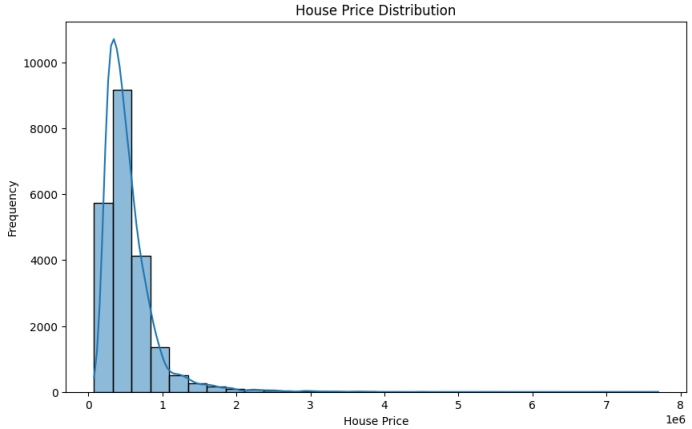

Figure 2: House Prices Distribution

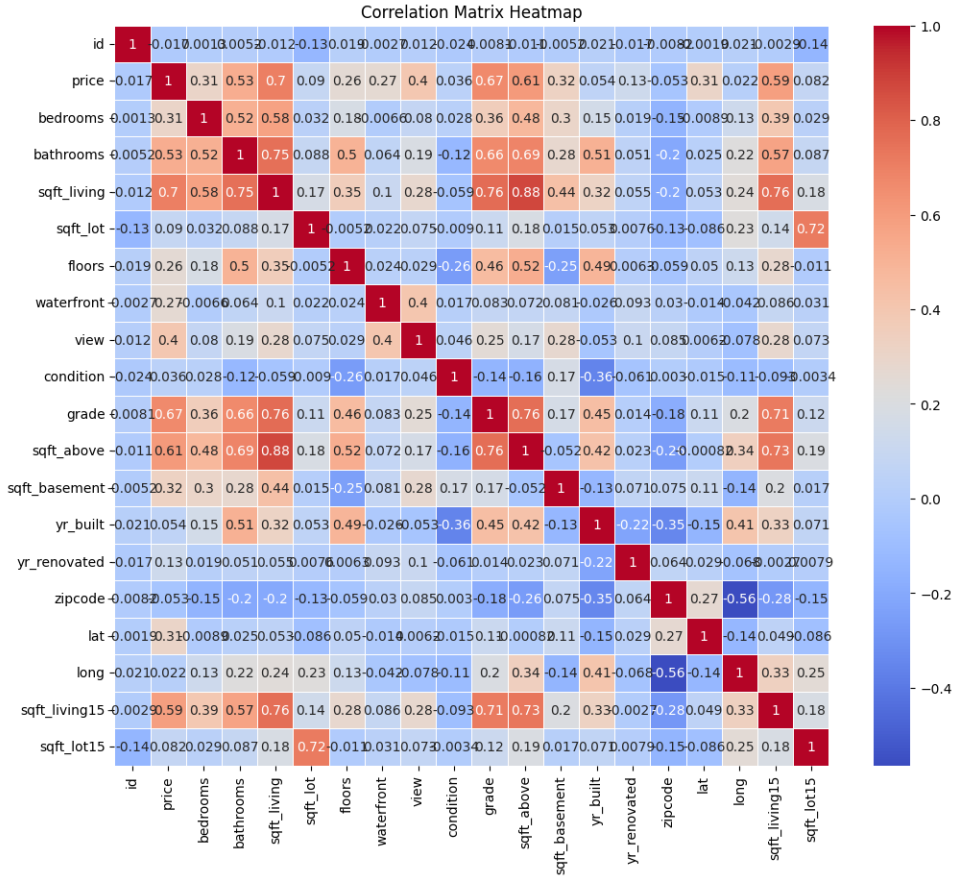

Figure 3: Correlations between Variables

backpropagation. MSE values are calculated on the training set and testing set.

3.2.2 RNN. An RNN model initializes neural networks with a multiple-layer RNN "rnn" with input size 18, hidden size 32 and 2 recurrent layers and a single layer feedforward "fc" with 32 inputs and 1 output. The model defines a forward method such that first pass data through "rnn" and next pass it into "fc". The model then sets loss function as the MSE loss and sets gradient descent optimizer with learning rate 0.001. In each epoch in the model, weights

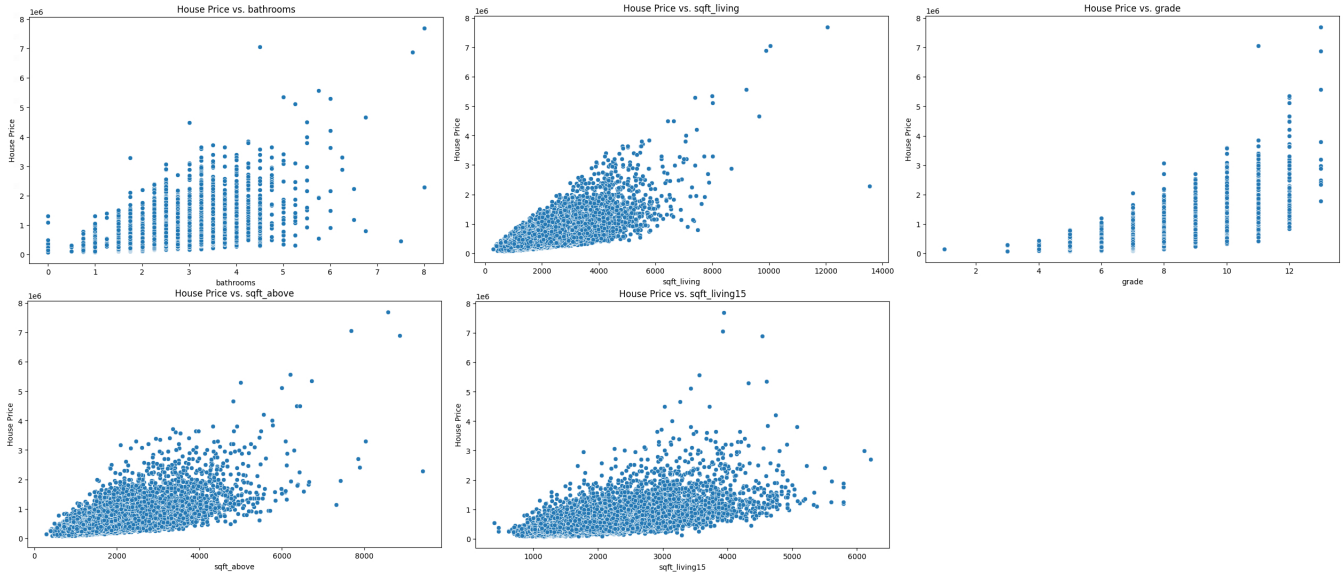

Figure 4: Scatter Plots of House Price with 5 Relatively Strong Correlations

are updated with forward pass and backpropagation. MSE values are calculated on the training set and testing set.

3.2.3 BiRNN. A BiRNN model initializes neural networks with a multiple-layer bidirectional RNN "birnn" with input size 18, hidden size 32 and 2 recurrent layers and a single layer feedforward "fc" with 64 inputs and 1 output. The model defines a forward method such that first pass data through "birnn" and next pass it into "fc". The model then sets loss function as the MSE loss and sets gradient descent optimizer with learning rate 0.001. In each epoch in the model, weights are updated with forward pass and backpropagation. MSE values are calculated on the training set and testing set.

3.2.4 LSTM. An LSTM model initializes neural networks with a long-short term memory "lstm" with input size 18, hidden size 32 and a single layer feedforward "fc" with 32 inputs and 1 output. The model defines a forward method such that first pass data through "lstm" and next pass it into "fc". The model then sets loss function as the MSE loss and sets ADAM optimizer with learning rate 0.001. In each epoch in the model, weights are updated with forward pass and backpropagation. MSE values are calculated on the training set and testing set.

3.2.5 Seq2Seq. A Seq2Seq model initializes neural networks with two long-short term memory "lstm" as "encoder" and "decoder" and a single layer feed forward with 128 inputs and 1 output. The model defines a forward method such that it first encodes the input sequence and decodes the target sequence, and then a linear layer is applied to get the output sequence. The model sets loss function as the MSE loss and sets ADAM optimizer with learning rate 0.001. In each epoch in the model, weights are updated with forward pass and backpropagation. MSE values are calculated on the training set and testing set.

4 EVALUATION AND RESULTS

Code Repository: KC house Prediction

4.1 MSE Loss

In order to ensure the accuracy of each model and make proper comparisons between models, we use MSE loss to calculate the average loss for each model.

We can define the MSE loss metric as follows:

$$\text{Mean Squared Error} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

, where n is the number of samples, y_i is the true value of the i-th sample, and $\hat{y}_i$ is the predicted value of the i-th sample. The MSE is calculated as the average of the squared differences between the true and predicted values.

In general, MSE loss is more sensitive to outliers than MAE loss because it involves squaring the error term. So, MSE is better suited when to minimize the average square error, and the dataset does not contain significant outliers. MAE loss is more robust to outliers because it takes the absolute value of the error term, so MAE is used when the dataset has outliers and the goal is to minimize the average absolute error. R-Squared loss measures the difference between the predicted values and the true values of the dependent variable. It is used to evaluate the performance of the regression models during training. The goal is to optimize the model's parameters to minimize the R2 loss. However, R2 loss is less commonly used as a loss function in deep learning because it can be difficult to optimize directly. In our implementation, we previously normalized all the data to the range of (-1, 1), so outliers will not significantly impact our model loss prediction function. Thus, MSE loss is the most suitable evaluation metric to use among the three.

4.2 Evaluation Results

The train and test loss of the five models are given in table 1 to 5.

Epoch	Train Loss	Test Loss
[10/1000]	0.0687	0.0731
[20/1000]	0.0643	0.0686
[30/1000]	0.0603	0.0644
[40/1000]	0.0567	0.0606
[50/1000]	0.0534	0.0572
[60/1000]	0.0504	0.0540
[70/1000]	0.0476	0.0512
[80/1000]	0.0451	0.0485
[90/1000]	0.0428	0.0461
[100/1000]	0.0407	0.0439

Table 1: Train and Test Loss of Dense Layer Model

Epoch	Train Loss	Test Loss
[10/1000]	0.0471	0.0083
[20/1000]	0.0318	0.0064
[30/1000]	0.0217	0.0051
[40/1000]	0.0151	0.0043
[50/1000]	0.0107	0.0037
[60/1000]	0.0079	0.0034
[70/1000]	0.0059	0.0031
[80/1000]	0.0047	0.0030
[90/1000]	0.0039	0.0028
[100/1000]	0.0033	0.0028

Table 2: Train and Test Loss of RNN Model

Epoch	Train Loss	Test Loss
[10/1000]	0.2666	0.0921
[20/1000]	0.1325	0.0481
[30/1000]	0.0669	0.0262
[40/1000]	0.0345	0.0151
[50/1000]	0.0184	0.0093
[60/1000]	0.0103	0.0063
[70/1000]	0.0063	0.0048
[80/1000]	0.0043	0.0039
[90/1000]	0.0033	0.0034
[100/1000]	0.0028	0.0032

Table 3: Train and Test Loss of BiRNN Model

The plots of loss curves of each neural network model are shown in Figure 5, as ordered from left to right, top to bottom: Dense Layer, RNN, BiRNN, LSTM, Seq2Seq.

From the above plots of loss curves, we can see that the dense layer model has the highest training loss and test loss among all the models. The training loss decreases significantly as the epochs progress, but the test loss remains relatively high and does not converge to the training loss, indicating that the model may be overfitting the data. The RNN model has a lower training loss and test loss compared to the Dense Layer model. The model shows a decreasing trend in both training and test losses as the epochs progress and converges in the end, indicating that the model is learning the patterns in the data effectively. The BiRNN model has a higher initial training loss, but it decreases significantly with each epoch. The test loss decreases sharply as well and converges to the training loss in the end, which shows that the model is learning effectively from the data. This model has the relatively low test loss. For both RNN and BiRNN models, the test loss curves are below the training loss curve. This is likely because when the parameters are randomly initialized, they tend to be biased towards the portion of test set with better performance, so the test losses are initially lower.

Epoch	Train Loss	Test Loss
[10/1000]	0.0030	0.0033
[20/1000]	0.0026	0.0030
[30/1000]	0.0024	0.0028
[40/1000]	0.0023	0.0026
[50/1000]	0.0023	0.0026
[60/1000]	0.0023	0.0026
[70/1000]	0.0023	0.0026
[80/1000]	0.0023	0.0026
[90/1000]	0.0022	0.0026
[100/1000]	0.0022	0.0026

Table 4: Train and Test Loss of LSTM Model

Epoch	Train Loss	Test Loss
[10/1000]	0.0039	0.0036
[20/1000]	0.0027	0.0031
[30/1000]	0.0023	0.0026
[40/1000]	0.0023	0.0026
[50/1000]	0.0023	0.0026
[60/1000]	0.0023	0.0026
[70/1000]	0.0022	0.0026
[80/1000]	0.0022	0.0026
[90/1000]	0.0022	0.0026
[100/1000]	0.0022	0.0026

Table 5: Train and Test Loss of Seq2Seq Model

The LSTM model has the lowest initial training and test losses compared to other models. The model shows a decreasing trend in both training and test losses as the epochs progress and converges to the training loss very quickly, which results in the lowest test loss among all the models, indicating that the model is learning effectively from the data. The Seq2Seq model has a training loss and test loss comparable to the LSTM model. The model shows a decreasing trend in both training and test losses as the epochs progress and converges to the training loss very quickly, which also results in the lowest test loss among all the models, indicating that the model is learning effectively from the data.

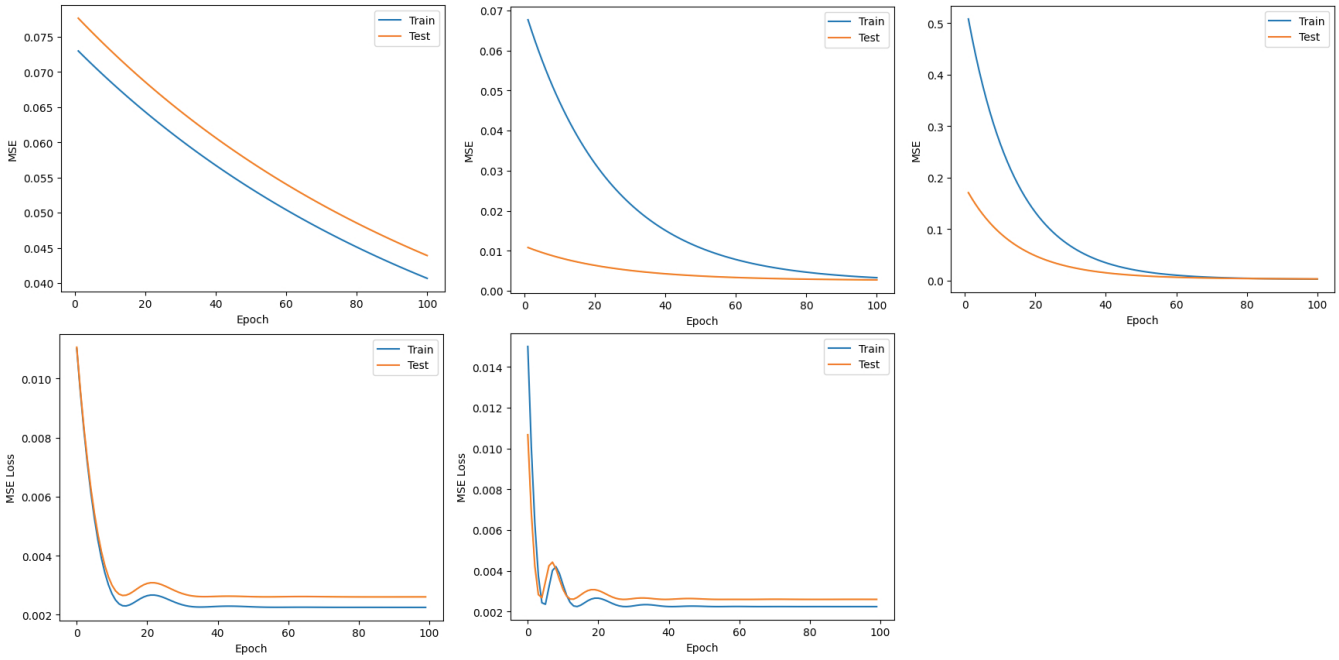

Figure 5: MSE Loss Curves of Five Neural Network Models (Ordered from left to right, top to bottom: Dense Layer, RNN, BiRNN, LSTM, Seq2Seq)

Overall, for forecasting King County house prices, the LSTM and Seq2Seq models with the aforementioned architectures have the lowest test losses among all models, indicating that these models are better suited to learning from the given dataset. In addition, since the training loss and test loss are very close to each other for LSTM and Seq2Seq models, it implies that these two models are more capable of accurately predicting King County house prices for previously unseen data if put into practice. What's more, the RNN and BiRNN models perform relatively well with lower test losses; however, the dense layer model shows relatively higher test loss, indicating overfitting.

5 CONCLUSION AND FUTURE WORK

In summary, this paper develops five Artificial Neural Network (ANN) models for predicting house prices in King County, compared to the baseline model, a full linear regression model. Before modeling, data was preprocessed including dropping unnecessary columns 'id' and 'date' and normalization and min-max scaling to the split data. Then five neural networks, which are dense layer, RNN, biRNN, LSTM and seq2seq models, are constructed with corresponding architecture. In result, the LSTM model with initialization functions of a LSTM layer and a linear layer, forward method such that passing data through LSTM layer and next put it into linear layer, and with MSE loss and adam optimizer with learning rate 0.001. Hence for predicting house prices in King County, the LSTM model with given architecture is most accurate with loss 0.0026 on the test set. However, it is interesting to notice that the seq2seq with given architecture has better generalization ability.

All of the aforementioned techniques can be applied to time series prediction, where the goal is to forecast future values of a sequence based on historical data. For example, linear regression can be used to model trends and patterns in the data, while RNNs, BiRNNs, LSTMs, and Seq2Seq models can capture more complex dependencies and relationships between the inputs and outputs.

Moreover, we believe that such a domain of problem will be of greater significance in the future, with the growth of the house market. Future work can focus on developing more sophisticated models to probe the data and derive knowledge on more feasible attributes of house prices. Particularly, our findings could be utilized independently or in conjunction with fundamental predictions to generate perspectives on house price trends. Our empirical methodology should be simple to implement, which is crucial for many decision makers, and it has the potential to be generalized for housing price forecasting in other cities and countries.

REFERENCES

[1] Ruben D. Jaen. 2002. Data mining: An empirical application in real estate valuation. In *FLAIRS Conference*. (2002), 314–317.