Lab4 for STAT 415

Xiaolong Liu

May 2024

1 Introduction

In this lab, we will develop various predictive models using housing price data and evaluate their performance through diverse statistical analyses and visualization techniques.

2 Description of Variables and Data Preparation

Here are descriptions of each variable in the dataset. We apply one-hot encoding to transform categorical variables and scale numerical variables. For the response variable **sold_price**, we take the natural logarithm (log). This transformation is necessary because using the original scale results in a very large training loss, which prevents our predictive models from achieving meaningful results. The models perform significantly better with log(sold_price) as the response variable. (Using standardized instead of log has similar result). A more detailed discussion of this approach will be presented later. There're 110 variables in total.

Table 1: Description of Variables

Column Name	Description
property_url	URL of the property listing on realtor.com
style	Style of the house (e.g., condo, single house, mobile home)
beds	Number of bedrooms
full_baths	Number of full bathrooms
half_baths	Number of half bathrooms
sqft	Total square footage of the house
year_built	Year when the house was built
sold_price	Sale price of the house in USD (target variable)
lot_sqft	Square footage of the lot
latitude	Latitude coordinate of the property
longitude	Longitude coordinate of the property
county	County where the property is located
stories	Number of stories in the house
parking_garage	Number of parking garages

3 Predictive Models

We will split the dataset into different parts to fit our models. For the simple linear model, since there is no model selection problem (i.e., no hyperparameters), we will simply split the data into a training dataset (80%) and a test dataset (20%). For all other models, we will divide our data into three parts: a training dataset (60%), a validation dataset (20%) (used to choose the model), and a test dataset (20%).

3.1 Linear Models

3.1.1 A simple linear model

First, we fitted a simple linear model to the data, which performed poorly. The mean squared error (MSE) on the training dataset was about 0.375, and the MSE on the test dataset was approximately 1.24×10^{19}, despite trying multiple random seeds! This indicates severe overfitting issues.

Below is the histogram of the true response in the test dataset.

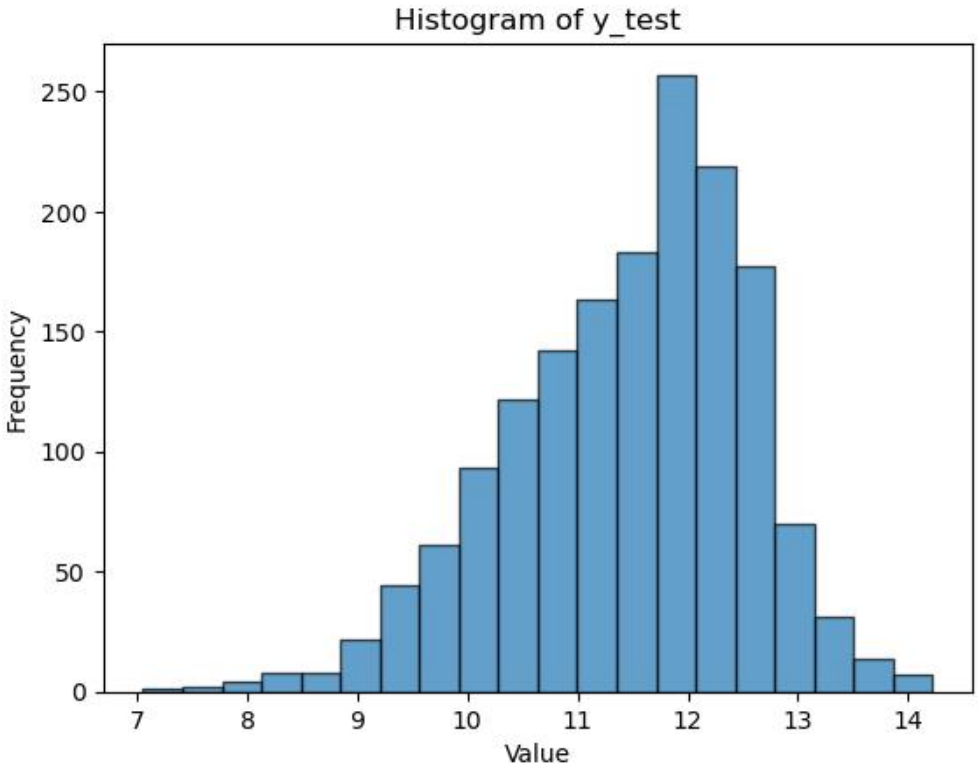


Figure 1: Histogram of the true response in the test dataset.

Here is the histogram of the predicted response. We observe several extremely large values, making it difficult to appreciate the density of the predicted values.

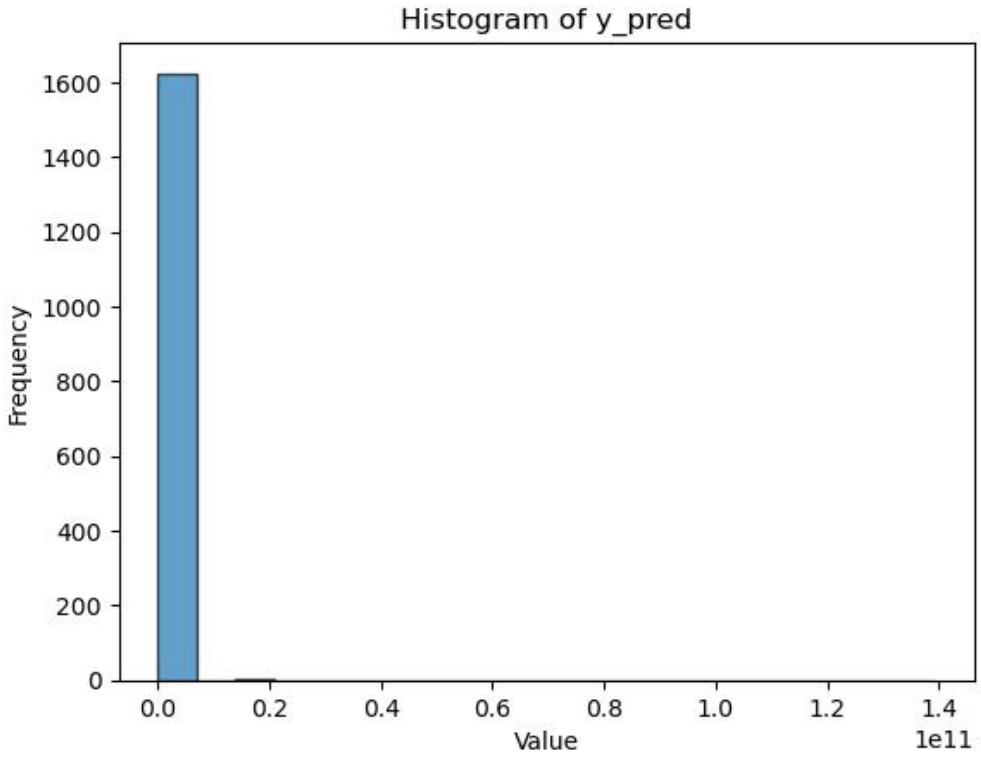


Figure 2: Histogram of the predicted response with extreme values.

To improve visualization, we excluded values greater than the 99% percentile and redrew the histogram. The resulting shape is somewhat similar to the true response but not perfect. Given the presence of extreme values, which we temporarily ignored, this simple linear model does not perform well.

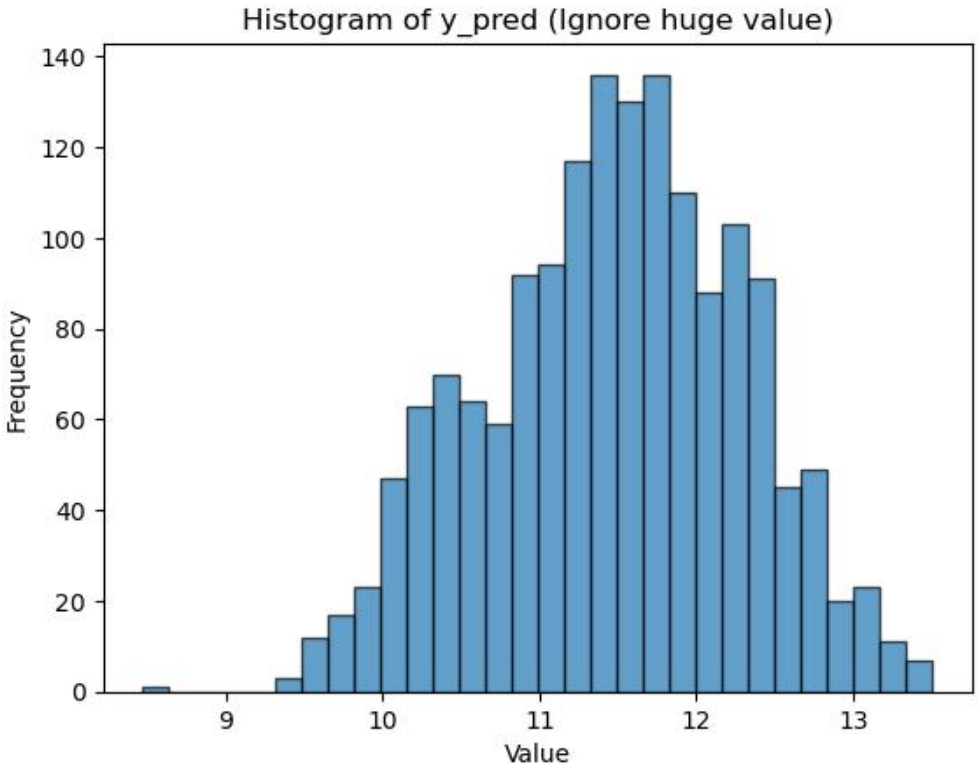


Figure 3: Adjusted histogram of the predicted response after removing extreme values.

3.1.2 The linear model errors

Then, we plotted histograms for the errors. Below is the histogram for errors in the training dataset.

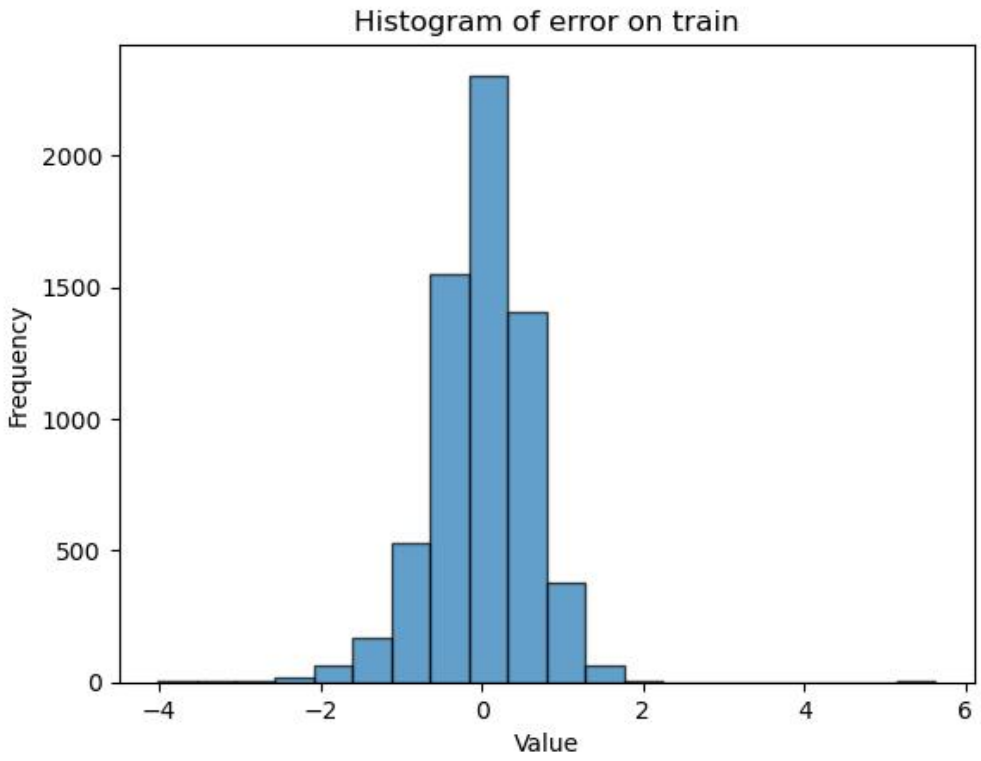


Figure 4: Histogram of errors in the training dataset.

Here is the histogram for errors in the test dataset, excluding the extreme values.

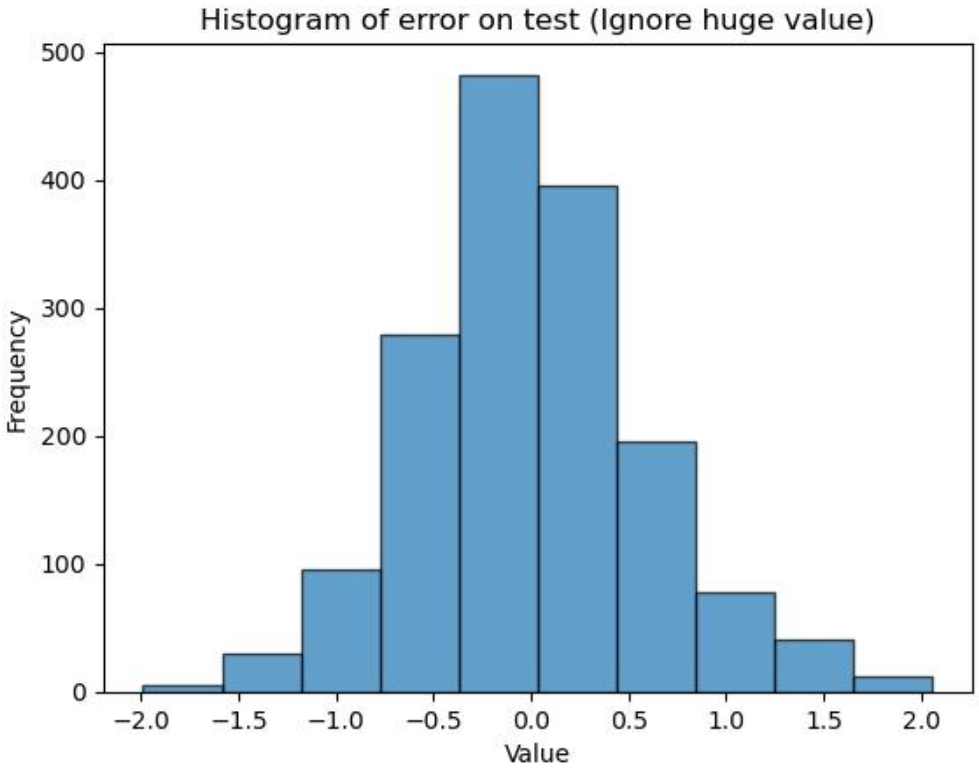


Figure 5: Histogram of errors in the test dataset with an extreme value ignored.

We observe that the distribution of errors is similar to a bell shape, which aligns with the assumptions of the normal linear model. The 'bell' in the training dataset is higher than the one in the test dataset. However, note that there are several extreme values in the data that are not shown in these histograms, which is unusual.

3.1.3 Linear model with L1 and L2 regularization

To combat overfitting, we implemented L1 and L2 regularization techniques. We utilized grid search along with a validation dataset to determine the optimal hyperparameter α, which is the coefficient of regularization.

For L1 regularization (Lasso), the best α was found to be 0.001. The mean squared error (MSE) on the training dataset is 0.41, on the validation dataset is 0.43, and on the test dataset is 0.40. This represents a significant improvement over the simple linear model. Below is a figure showing the features selected by this method:

Selected Features by Lasso:

	Feature	Coefficient
0	style_CONDOS	0.076713
1	style_CONDO_TOWNHOME_ROWHOME_COOP	-0.128211
3	style_MOBILE	-0.431689
6	style_TOWNHOMES	0.105822
11	county_Bureau	-0.165633
13	county_Carroll	-0.298073
14	county_Champaign	0.284610
18	county_Clinton	0.055175
20	county_Cook	0.479917
24	county_DeKalb	0.026370
26	county_DuPage	0.796858
36	county_Grundy	0.594435
44	county_Jersey	0.226171
47	county_Kane	0.527022
48	county_Kankakee	0.052792
49	county_Kendall	0.449678
50	county_Knox	-0.496772
51	county_LaSalle	0.061904
52	county_Lake	0.341176
54	county_Lee	-0.456921
59	county_Madison	0.260975
65	county_McHenry	0.276673
66	county_McLean	0.113180
67	county_Monroe	0.501011
71	county_Ogle	-0.244126
85	county_St. Clair	0.190079
87	county_Stephenson	-0.519360
88	county_Tazewell	-0.057837
90	county_Vermilion	-0.183182
95	county_Whiteside	-0.574738
96	county_Will	0.504718
98	county_Winnebago	-0.374940
100	beds	0.045138
101	full_baths	0.290040
102	half_baths	0.144487
104	year_built	0.164144
105	lot_sqft	0.055874
106	latitude	0.372331
107	longitude	-0.081040
108	stories	0.061636
109	parking_garage	0.136419

Figure 6: Features selected by L1 regularization.

For L2 regularization (Ridge), the best α was 1. The MSE on the training dataset is 0.38, on the

validation dataset is 0.42, and notably higher on the test dataset at 9.2. This is still an improvement over the simple linear model but not as effective as L1 regularization.

3.2 Decision Tree

3.2.1 Decision Tree Model

We fitted a decision tree model, employing grid search and a validation dataset to identify the optimal parameters. The parameters tested included:

- `max_depth`: [3, 5, 10, 20, None]
- `min_samples_split`: [2, 10, 20]
- `min_samples_leaf`: [1, 5, 10]
- `max_features`: [1.0, 'sqrt', 'log2', None]

The best parameters determined were `max_depth` of 10, `max_features` as 1.0, `min_samples_leaf` of 10, and `min_samples_split` of 2.

The mean squared error (MSE) results were as follows:

- Training dataset: 0.22
- Validation dataset: 0.37
- Test dataset: 0.36

These results show an improvement over those obtained with the linear model, demonstrating the efficacy of the decision tree in this context.

3.2.2 Visualization

We have attempted to visualize the decision tree constructed from our analysis. Due to the extensive size of the tree, it is impractical to display it clearly within the confines of this document. Instead, we have uploaded the visualization to Google Drive. You can access the image via the following link: Google Drive Link (You may need wait a second when you zoom in the image, https://drive.google.com/file/d/1n1bvOrmI-kbk2yIqq9EoTJ9yJckuGisB/view?usp=share_link).

3.3 Random Forest

3.3.1 Random Forest Model

We next fitted a random forest model to our dataset, utilizing grid search and a validation dataset to determine the most effective parameters. The parameters explored were as follows:

- `n_estimators`: randint(50, 200)
- `max_depth`: [3, 5, 10, 20, None]
- `min_samples_split`: randint(2, 20)
- `min_samples_leaf`: randint(1, 10)
- `max_features`: ['sqrt', 'log2', None]

The optimal parameters identified were `n_estimators` of 63, `max_depth` of None, `min_samples_leaf` of 2, `min_samples_split` of 13, and `max_features` as None.

The mean squared error (MSE) observed was:

- Training dataset: 0.12
- Validation dataset: 0.29
- Test dataset: 0.27

These results indicate that the random forest model outperformed all previous models tested.

3.3.2 Comparison of the forest outputs versus the true data distribution

We have constructed a histogram to compare the outputs of the random forest model with the true data distribution in the test dataset. The comparison reveals that the distributions are quite similar, though the distribution for the random forest outputs exhibits a more distinct bimodal pattern than the true data.

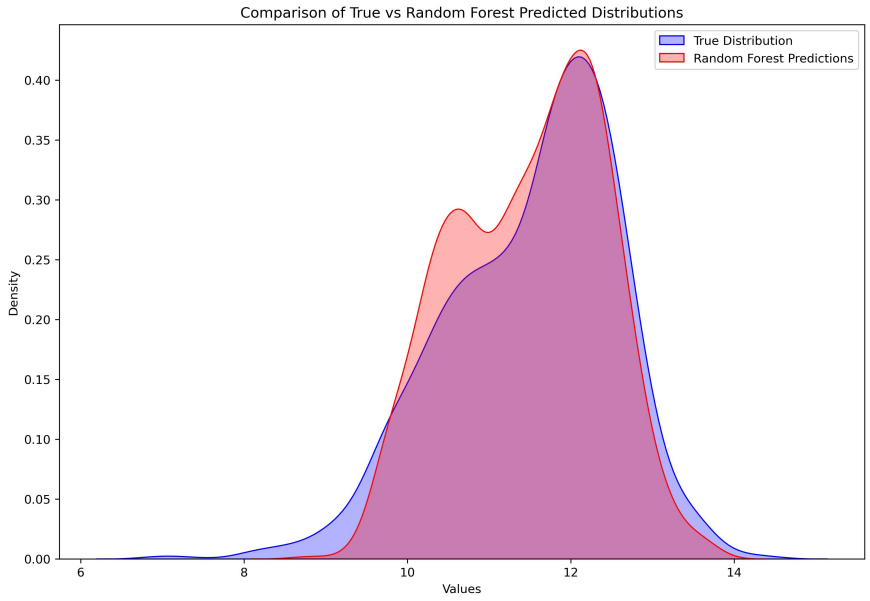


Figure 7: Comparison of random forest outputs and true data distribution on the test dataset.

3.3.3 Tune random forest

In fact, we have already tuned our random forest model using grid search to find the best parameters. These parameters significantly influence the model's performance. For instance, using the following settings:

- `n_estimators` = 10
- `max_depth` = 2
- `min_samples_split` = 10
- `min_samples_leaf` = 5
- `max_features` = 10

resulted in the following mean squared error (MSE) values:

- Training dataset: 0.72
- Validation dataset: 0.78

- Test dataset: 0.73

These results are worse than those obtained with the previous models.

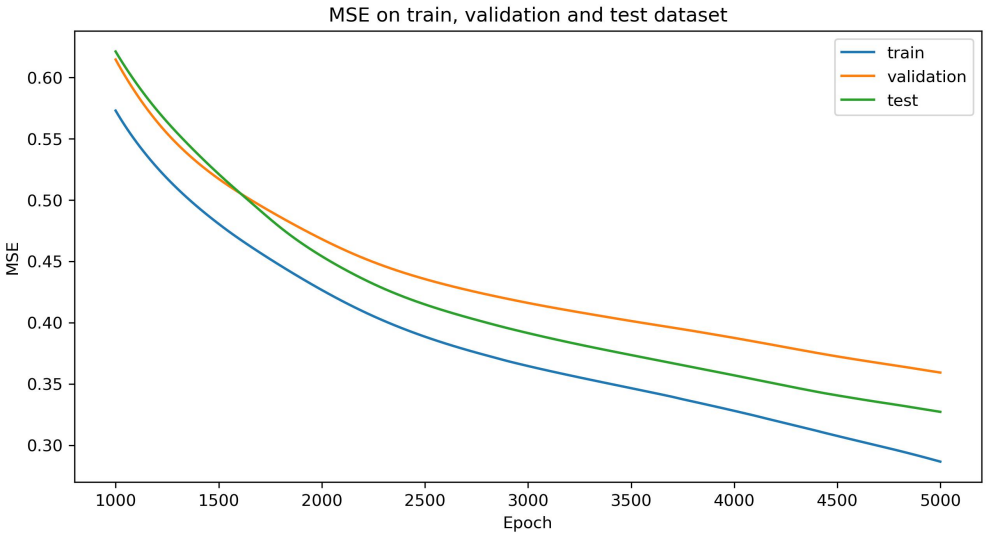
Indeed, `n_estimators` affects the robustness of the model, while other parameters like `max_depth`, `min_samples_split`, `min_samples_leaf`, and `max_features` can influence the accuracy of the model—potentially leading to overfitting if set too high. All of these parameters are crucial for optimizing performance.

3.4 Neural Networks

3.4.1 3-layer neural network with sigmoid activations

We employed a 3-layer neural network with sigmoid activation functions (MSE loss and Adam optimizer, first layer 64 nodes, second layer 32 nodes, third layer 1 node). There are $64n + 2177$ parameters in the model. The model was trained for up to 5000 epochs, incorporating an early stopping mechanism that halted training if the loss on the validation dataset did not improve after 10 consecutive checks. This approach yielded the following mean squared error (MSE) values:

- Training dataset: 0.28
- Validation dataset: 0.36
- Test dataset: 0.33



In fact, the MSE could potentially be reduced further by extending the number of training epochs. However, due to computational constraints, we decided to conclude our training at this point.

3.4.2 Comparison of the forest outputs versus the true data distribution

We conducted a comparison between the outputs of the neural network and the true data distribution from the test dataset. It is well-documented that neural networks tend to converge to the mean output. Our observations confirm that the distribution of the neural network outputs clearly tends to converge towards the mean of the true distribution.

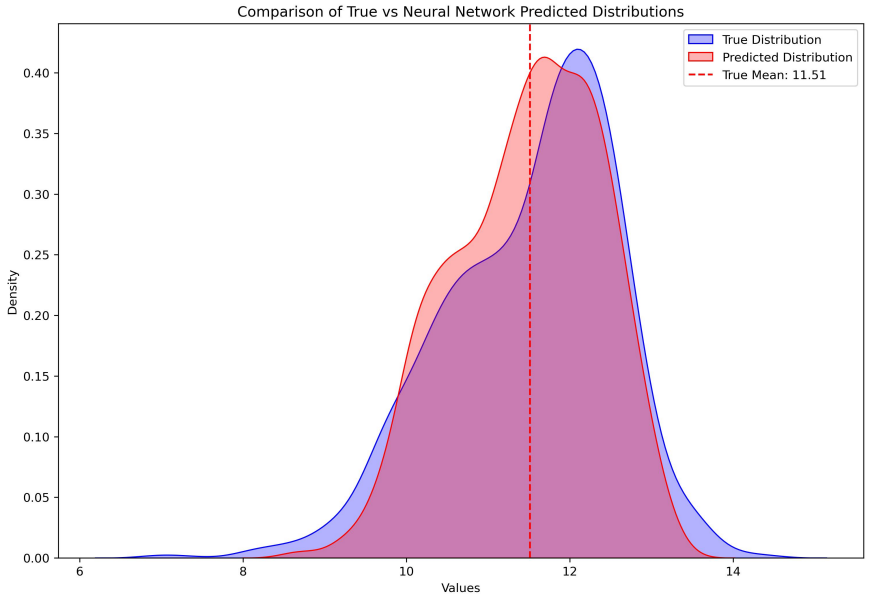


Figure 8: Comparison of neural network outputs with the true data distribution.

This phenomenon was consistent across various trials; it does not depend on the random seed used during initialization. We have retrained our model using several different random seeds to verify this behavior. The results of these additional experiments are included in the Appendix.

3.4.3 Tune the net

We implemented a four-layer neural network featuring tanh activation functions, dropout with a probability of 0.5, and batch normalization (using `BatchNorm1d`). The network architecture includes the following node configuration: first layer with 128 nodes, second layer with 64 nodes, third layer with 16 nodes, and the final layer with a single node. There are $128n + 9441$ parameters in the model. We used the mean squared error (MSE) loss function and the Adam optimizer for training. The network was trained for 15,000 epochs without employing early stopping.

This training regimen resulted in the following mean squared error (MSE) values:

- Training dataset: 0.42
- Validation dataset: 0.34
- Test dataset: 0.30

Interestingly, the loss on the training dataset was greater than that on the test and validation datasets (this may because the regularization like dropout and batchnorm), and it continued to decrease over time.

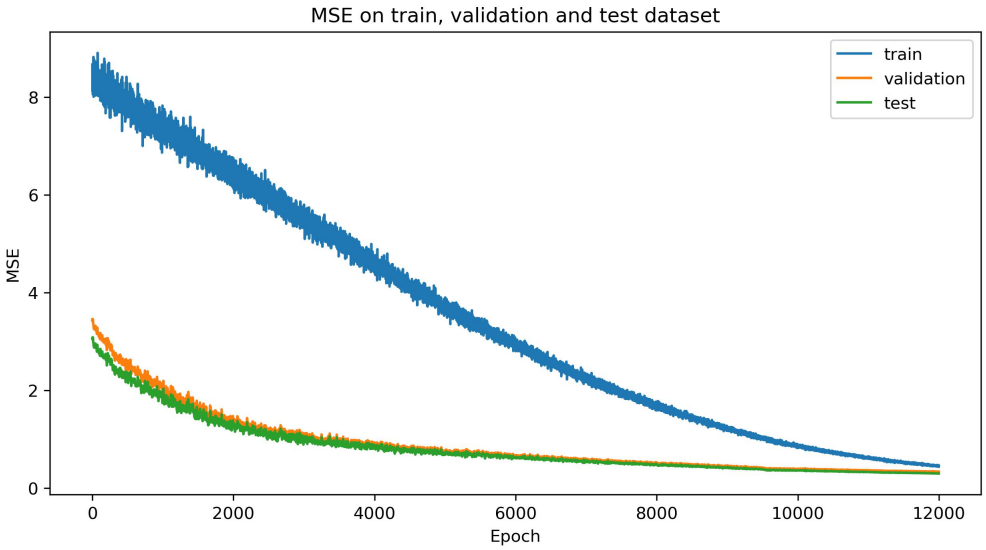
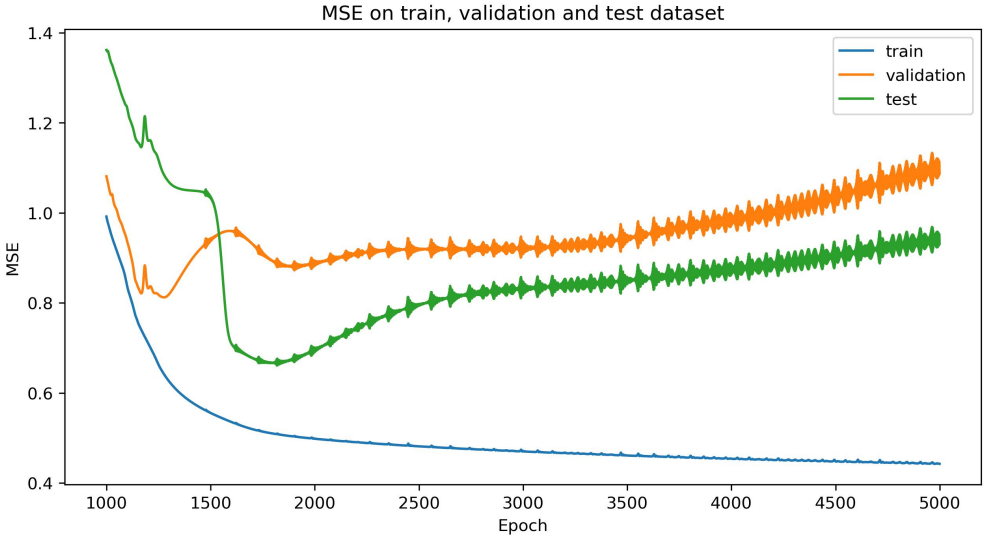


Figure 9: MSE on train, validation and test dataset (remove first 2000 epoches).

If we remove dropout and batch norm:



3.5 Transfer Learning

3.5.1 Compare to performance on WI housing and on IL housing

We trained various models using the `WI_housing_price.csv` dataset and compared their performance with those trained on the Illinois dataset.

Model Performance Overview:

- **Simple Linear Model:**
  - MSE on train: 0.044
  - MSE on test: 1.577

- **L1 Linear Model:**
  - MSE on train: 0.05
  - MSE on validation: 0.56
  - MSE on test: 0.37
- **L2 Linear Model:**
  - MSE on train: 0.13
  - MSE on validation: 0.48
  - MSE on test: 0.38
- **Decision Tree:**
  - MSE on train: 0.19
  - MSE on validation: 0.55
  - MSE on test: 0.30
- **Random Forest:**
  - MSE on train: 0.07
  - MSE on validation: 0.65
  - MSE on test: 0.47
- **Neural Network:**
  - MSE on train: 0.186
  - MSE on validation: 0.535
  - MSE on test: 0.54

Comparative Analysis:

- For the simple linear model, L1, and L2 linear models, as well as the decision tree, the models trained on the `WI_housing` dataset performed better than those trained on the `IL_housing` dataset.
- For the random forest and neural network, the results were reversed. This may be due to the fact that the `WI_housing` dataset is smaller, which suggests that fewer parameters (which may cause overfitting) are necessary.

3.5.2 Transfer learning with Linear Model

Now we try our best to achieve some transfer learning. Take a linear model and train it on the data from `IL_housing_price.csv`. Then, take that trained linear model and try to make predictions on the data from `WI_housing_price.csv`. Since the counties in Illinois (IL) are not in Wisconsin (WI), we cannot use this information anymore. Additionally, we should add more types in **style** in `WI_housing_price.csv` to ensure consistency with IL.

We use an L1 regularization linear model on `IL_housing_price.csv`. Grid search selects $\alpha = 0.001$.

The mean squared error (MSE) observed was:

- Training dataset: 0.48
- Validation dataset: 0.43
- Test dataset: 0.49

Then we use this model to predict on `WI_housing_price.csv`, resulting in an MSE of 0.827.

3.5.3 Transfer learning with Neural Network Model

Now we attempt to achieve some transfer via training. First, train a neural network on `IL_housing_price.csv`. After this initial training, fine-tune the network by training on the data from `WI_housing_price.csv`.

We train the same 3-layer network model as before, implementing an early stopping rule. We obtain an MSE of 0.331 on the training set and 0.320 on the validation set.

Then we train this model on `WI_housing_price.csv` for 1000 epochs (as the dataset is small, and prolonged training risks overfitting). The mean squared error (MSE) observed was:

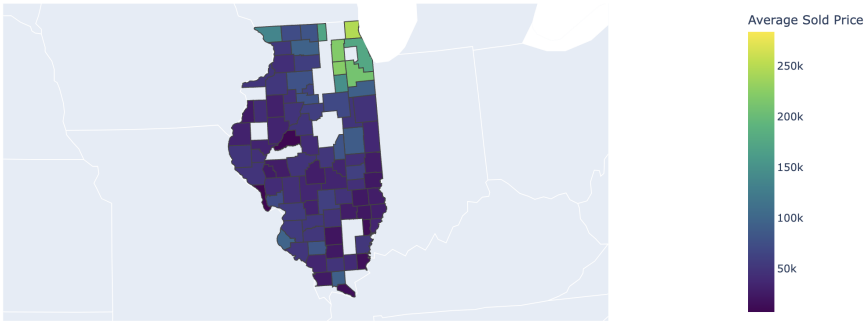
- Training dataset: 0.210
- Validation dataset: 0.686
- Test dataset: 0.520

This result is not bad, but it only surpasses the result of a simple linear model when all models are directly fitted on `WI_housing_price.csv`.

3.6 Visualization

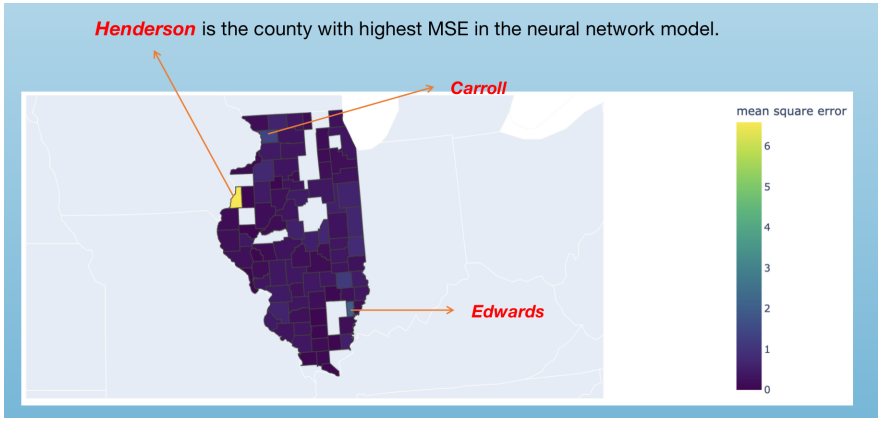
3.6.1 Average sale price Map

Now we create a Choropleth map of Illinois counties, colored by the average sale price of foreclosure properties.



3.6.2 MSE Map

Now we develop a second Choropleth map showing predictive errors by county using Neural Network. We could find that **Henderson** (MSE: 6.59) is more difficult to predict accurately.



However, there is only one observation for **Henderson** (MSE: 6.59) in the dataset, and we use the MSE in the test dataset to evaluate each county. As a result, there is no observation for **Henderson** in the training dataset. The second largest MSE is for **Edwards** (MSE: 1.99), which also has only one observation in the dataset. Following that is **Carroll** (MSE: 1.33) with 12 observations, which we believe is not caused by randomness. (The average of all the MSE is 0.37)

Removing these three counties does not help to lower the MSE. This may because they do give some information of the price by other variables like style, beds and so on.

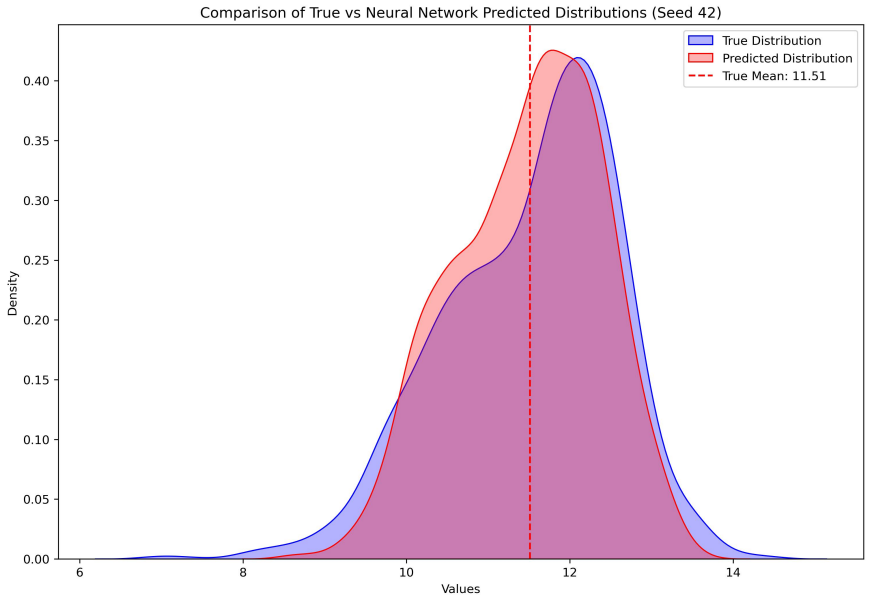
4 Extra Discussion

As stated initially, all predictions are made for the logarithm of the price. Although we achieve a very low Mean Squared Error (MSE) in the logarithmic space, when we revert to the original scale by exponentiating, the predictions can exhibit a significantly higher MSE. This issue arises because a small error in logarithmic space can translate into a substantial error in the original price space upon exponentiation. Note that using a standardized scale instead of the logarithm does not mitigate this problem.

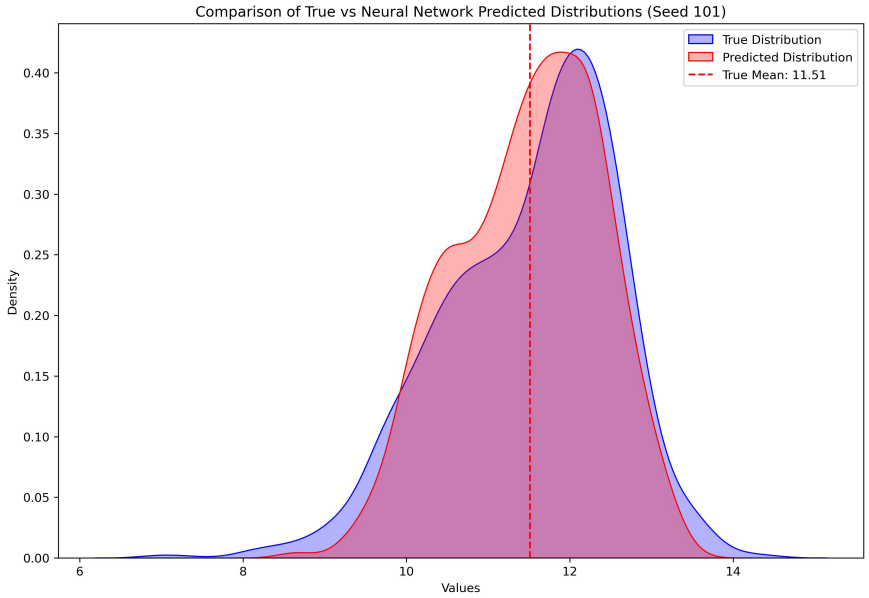
Given the substantial variance in the original data (sold prices), it is reasonable for our model to exhibit a large Mean Squared Error (MSE) when predicting on the original scale.

5 Appendix

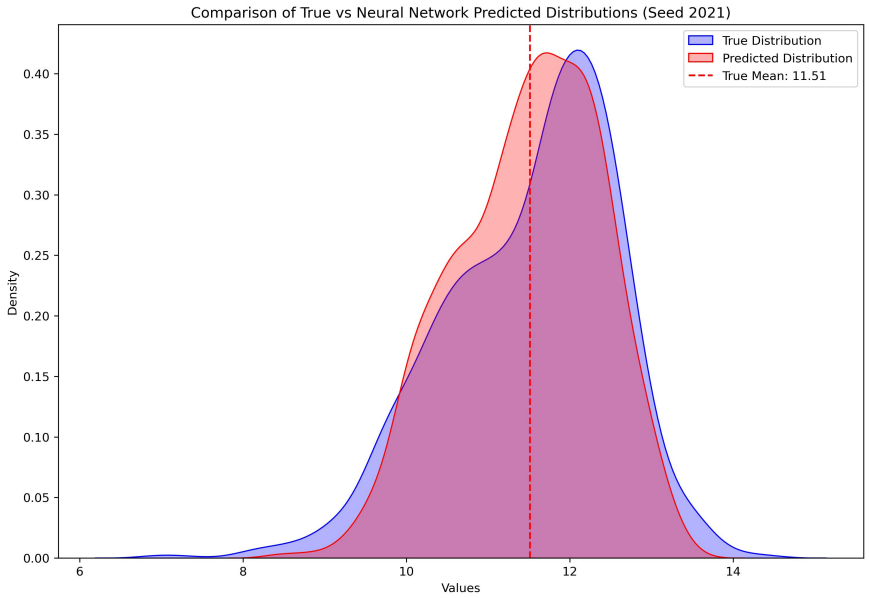
seed = 42



seed = 101


Comparison of True vs Neural Network Predicted Distributions (Seed 101)
True Distribution
Predicted Distribution
True Mean: 11.51
Density
Values
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
6
8
10
12
14

seed = 2021


Comparison of True vs Neural Network Predicted Distributions (Seed 2021)
True Distribution
Predicted Distribution
True Mean: 11.51
Density
Values
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
6
8
10
12
14