A Net Over Your Head: A Neural Network Approach to Home Price Predictions

HONGTAO SUN (S3SUNHT) and JI HOON "ANDY" KIM (JKIM4223)

1 MOTIVATION AND PROBLEM STATEMENT

For most families around the world, real estate is the single, largest source of income and wealth, contributing to its significance in global markets. Consequently, real estate economics have been widely discussed in literature garnering extensive interest from both buyers and sellers. Baseline models used in literature include using multiple linear regression with model selection, Principal Component Regression and regression forests. However, work on using neural networks to gain an understanding of the housing market has been relatively minimal. Therefore we aim to answer, can we develop a novel neural network to outperform the baseline models? Or can we develop another novel model that can outperform all the aforementioned models?

2 METRICS

Before we delve into the data, we have to understand the metric that will be used to optimize and evaluate the models with. Since the dataset was derived from a Kaggle competition, we will be using the same metric that is used for the competition and give an intuition as to why this measure is used[Kaggle 2017]. After training our models, we obtain both the true output and our predicted output of our model. We note that house price prediction is a regression problem and thus, we wish to use some metric that takes in these two values and outputs a value that "scores" our models. We use the Root Mean Squared Logarithmic Error (RMSLE):

$$RMSLE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\log y^{(i)} - \log \hat{y}^{(i)})^2}$$

where N is the number of data points in testing set, $y^{(i)}$ is the i-th test true value and $\hat{y}^{(i)}$ is the predicted value.

This is similar to RMSE that is used to evaluate many models except that we take the difference between the logarithms instead of their actual values. This is because our datasets are composed of homes that are orders of magnitude difference in price and a bad prediction of a high-valued home will far outweigh the weights of a bad prediction of a low-valued home when using the RMSE metric. RMSLE will allow us to capture the orders of magnitude we are off in our predictions as opposed to the actual difference in housing price.

3 DATA OVERVIEW

We have two datasets that we will use in our study: HOUSE and MACRO.

3.1 Dataset HOUSE

HOUSE contains 30470 records of house information as well as their final sold price "price_doc". We consider the final sold price "price_doc" as our target variable. Features in HOUSE can be subsequently divided into building feature columns (11 features) and neighborhood feature columns (279 features). Building features are features particular to the unit building (e.g. total square-footage: "full_sq", the floor number of the unit: "floor"). Neighborhood features are features that pertain to the locality of the unit(e.g. number of shopping malls in the district of the unit: "shopping_mall_raion", distance to the metro: "metro_km_walk"). Many of the features are categorical as well such as names of the district, material of the buildings, etc...

Immediately, we note that the data is very noisy and sparse. There are 51 features out of the 291 features that have missing values which range from 0.08204% to 47.3926% of the data missing for that feature. In order to handle the missing data, we ideally impute these values and fill in the values with statistically sampled values; however, this is computationally expensive and cannot be run in the first forms our model. Therefore, we currently fill the missing values with 0's which hopefully will be changed with more computational power.

Furthermore, much of the data demonstrates high multicollinearity as some of the correlation values between features are very close to 1 (e.g. the correlation between the number of children in preschool and the number of children between 0 and 13 of age is 0.996) which makes it difficult to perform regression. The variable most correlated with the price of the housing is square footage of the home with a correlation of 0.306.

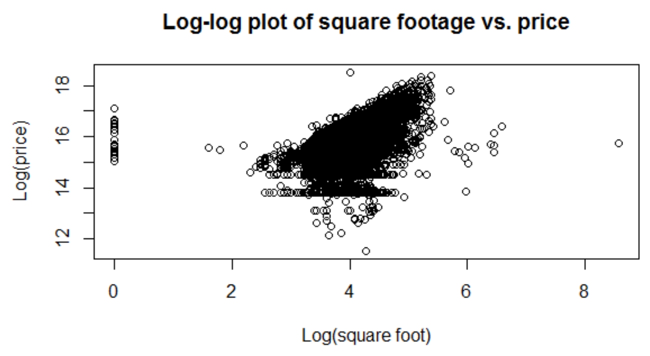

Fig. 1. Log-log plot between price of the home and its square footage

We also check the distribution of the prices of our data points on a log-scale in Figure 2. This distribution at a glance is normal and hence, our choice of using the RMSLE discussed earlier.

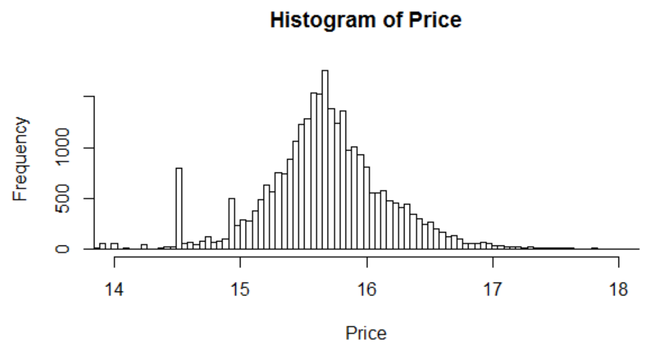

Fig. 2. Histogram and distribution of the price of properties. Price is on a log-scale

3.2 Dataset MACRO

The dataset MACRO includes features from macroeconomic variables. Each datapoint is the factors for the day from January 1st, 2010 to October 16th, 2016 comprising of 2484 data points and 100 features. As is the case with the HOUSE dataset, this dataset is just as sparse with many missing values, including a variable with nearly 71% of the data missing. Again, we use the same process as that used in HOUSE to fill in these missing values.

3.3 Modeling Dataset Preparation

To prepare modeling dataset, we need to combine the two datasets. We calculate the transaction data based on timestamps from the HOUSE dataset as well as the economic data record data from the MACRO dataset. With these two dates, we can append the two datasets together to form a comprehensive dataset with 391 features.

4 METHODS

4.1 Baseline Models

4.1.1 Principal Component Regression. Since the data is highly multicollinear as well as a large number of features, we need to perform a regression that can handle a reduced amount of features. For this portion, we try PCR since PCR usually performs well with high-dimensional data while also being able to potentially eliminate the singularities resulting from the $X^T X$ term in OLS. Principal Component Regression performs poorly with categorical variables so for the initial analysis, we omit these categorical variables for now.

4.1.2 Lasso and Ridge Regression. Since our data is highly correlated, the $X^T X$ term in OLS when solving for the coefficients is close to singular. In order to combat this, we introduce bias into our model in order to achieve a lower RMSLE through regularization with both the Lasso and Ridge regression estimates.

4.1.3 Regression Forests. As a contrast to some of the more aforementioned theoretical models, we will try using a non-parametric regression model such as regression trees in order to solve our problem. This will attempt to capture some of the interaction terms as well as some non-linearities in the data which cannot be capture as well by the other two models.

4.1.4 XGBoost Gradient Boosting. Now to gain a different perspective, we will try a gradient boosting method to see if we can better predict on this data set with respect to RMSLE. The added benefit of this is that the model is iterative similar to neural networks and the boosted trees only improve our predictions on the training set for each round. Naturally, this makes it a solid model to handle the sheer number of NAs we have in our model as well. This is our likely top-contender for the baseline models.[XGBoost 2015]

4.2 Neural Network

In our project, we want to compare results from neural networks against the baseline models. Since neural network includes non-linearity and complex structure, we hope that neural network can uncover more information compared to baseline models and outperform the other models with respect to the metric enumerated in Section 2.

5 EXPERIMENTAL RESULTS

5.1 Baseline Models

5.1.1 Principal Component Regression. As the most basic model, we perform rectified linear regression (since the output of our response cannot be negative) with all of the features in our dataset and the response log-transformed. Understandably, this results in a poor fit and achieves a RMSLE value of around 0.54. The poor fit likely developed from the large amount of heteroscedasticity as seen with the changing variance seen in Figure 3. We perform variable selection with backward selection and reduce the number of variables to 122 as opposed to the 292 we start out with. Refitting the model selected by model selection, we achieve a RMSLE error of 0.519326.

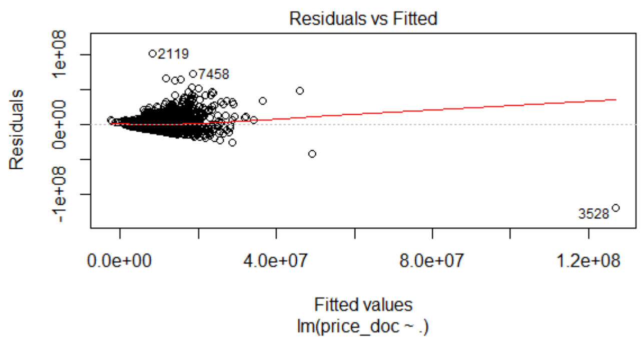

Fig. 3. Residuals vs. Fitted plot with linear regression

Now for this section, we perform PCR. We split our training data set into 90% for training and 10% for cross-validation and instead perform PCR on all the features to tackle some of these singularity issues. Running PCR on the log-transformed response, we achieve an RMSLE of 0.5196193 which is actually a poorer performance seen with model-selected linear regression. Therefore, we move onto other regressions that might yield better results.

5.1.2 Lasso and Ridge Regression. We perform both ridge regression and lasso regression with all of our parameters fitted in our model. We again use the log-transformed response and using the optimal empirically achieved λ, we achieve a RMSLE of 0.5031492 with ridge regression and 0.5068441 with Lasso regression with the best λ. Therefore, we have so far achieved the best model in terms of RMSLE score.

5.1.3 Regression Forests. Now we evaluate the random forest models. We evaluate with 100 trees generated from our data and use the log-transformed response once more. Running this against all of our features, we achieve a RMSLE score of 0.4606898. This improved score may be due to the quick variable selection and the ability of random forest to capture interactions and non-linearity in the data. [G. De'Ath 2000]

5.1.4 XGBoost Gradient Boosting. Part of the difficulties with working with the gradient boosting framework is that there are plenty of parameters to work around and finding the optimal set is rather difficult. Creating a grid of parameters, we explore the parameter space and choose the set of parameters where XGBoost performs the best (obtains the lowest RMSLE score on the training set). To train the model, the response is log-transformed as before. The optimal parameters found were to be $\lambda = 1.7$ (regularization term), $\eta = 0.23$ (learning rate), $h_{max} = 9$ (max depth of the trees created) with a linear regression objective (since we are not dealing with categorical outputs, rare events, nor some ranking task). Feeding these parameters, we achieve an RMSLE score of 0.4301.

5.2 Neural Network First Attempt

Now, we try to build a simple feed forward neural network using Tensorflow framework[TensorFlow 2017]. As a preliminary experiment, we impose several constraints for our data.

(1) We only use building features for our simple neural network
(2) The choices of hyper parameters for this test model is choose rather arbitrarily
(3) We only train our network for a small number of epochs because of time constraint

5.2.1 Features and Hyper-parameters Used. In this experiment, we only use the 11 building features. The feature columns are either numeric or categorical. We tried different structures for the neural network. For the following result and discussion, we're presenting a network with 2 hidden layers, with 25 and 15 neurons respectively. For each layer, we use a RELU activation function. We used a 70-30 split of training and test data.

We have 21,329 training examples. We trained our network for 40 epochs with mini-batch size of 40 examples.

5.2.2 Result and Discussion. After training using MSE loss function for neural network, we compute RMSLE for our prediction. The RMSLE is 1.952689 for the network we described above. We can see that this result is vastly worse than the baseline model. However, this is understandable as we only included building features.

5.3 Neural Network Final Attempt

To build on our previous neural network result and to improve it based on the points listed in last section, we build a more complex neural network model. In this model, we incorporate all building, neighborhood, and macroeconomic features (389 total). With that, our input dataset is high dimensional with numeric and categorical feature columns.

In addition, in this attempt, instead of 0, we fill missing numeric values with the column's median. For categorical feature, we create a separate category for missing values.

Similar to the previous attempt, we still use a 70-30 train-test split and create a feed-forward neural network with 2 hidden layers with 25 and 15 hidden neurons respectively. After adjusting learning rate from 0.01 to 0.1, the model was able to converge within 40 epochs.

5.3.1 Result and Discussion. We compute RMSLE for our prediction from the enriched neural network. The RMSLE is 0.2544. Here, we see a vast increase in model performance compare to the previous neural network. This result also beat all baseline model. However, this is understandable as we only included building features. This result confirms our initial suspicion that neural network based models can perform well in the context of house price prediction.

However, there are still many areas we want to work on that could improve the performance of this model. Most noticeably, we could cross validate more combinations of hyper-parameters in the model. We will comment more on possible improvements in the next section.

6 DISCUSSION AND FURTHER WORK

Table 1 summarizes the results of the various models used in this paper. We note that out of all the baseline models, the XGBoost gradient boosting framework performs the best against our test dataset. With that said, we note that the performance of the neural network far outperforms all of the baseline models with respect to RMSLE.

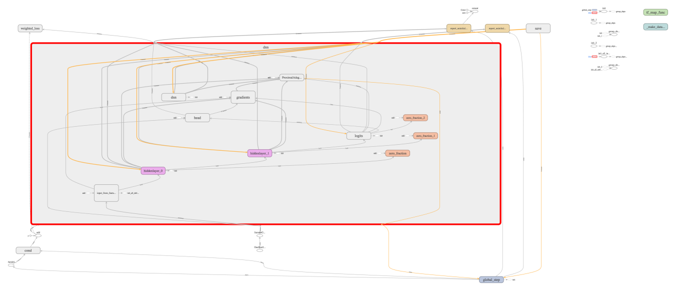

Fig. 4. Simple Neural Network Structure

Table 1. RMSLE metrics achieved by various models

Model	RMSLE
PCR	0.5196
Ridge	0.5068
LASSO	0.5031
Regression Tree	0.4607
XGBoost	**0.4301**
Neural Network	**0.2544**

The result we have is promising in the sense that it confirms that even though neural network is not widely used in house price prediction, it can produce superior results compared to standing models.

Although we successfully lowered the RMSLE between the baseline models and the neural network, we note that there is much room for improvement in both areas.

6.1 Further work on Baseline Models

(1) For Ridge/LASSO regression, we can try to perform better model selection to reduce the number of coefficients in our model since the high amount of coefficients might still produce overfitting even with regularization.
(2) Perform a more robust method of selecting parameters for λ values for Ridge, LASSO and XGBoost models as well as the other parameters for XGBoost. The set of values we explored may have simply converged to a local minima with respect to RMSLE.
(3) Again, impute the dataset to treat the missing values with a more statistically sound method rather than replacing NAs with the median of the column.

(4) Perform better feature engineering to extract more meaningful columns. Simply using one-hot encoding on categorical variables might not be the best method (For example, change a variable such as *photo_urls* to *number_of_photos*).
(5) Explore creating an ensemble of the baseline models (especially with XGBoost) such as bootstrapping or bagging to see if we can improve our prediction.
(6) Perform k-fold cross validation with more computational power rather than splitting the training set into training, dev and test sets once. This will allow us to better train the model since there will be more data to run across.

6.2 Further Work on Neural Network

(1) We can perform more experiments to determine the best hyper-parameter combination. To achieve this, we will need to utilize parallel computing because otherwise training would not be feasible.
(2) This project only considered shallow network. More work need to be done to determine whether deeper networks can improve model performance. If so, by how much.
(3) Since we have relatively small dataset to train neural network, we can include a higher percentage of our examples as training examples instead of using a 70-30 train-test split.
(4) We can investigate on how different activation functions affect model results.
(5) We can investigate on how imputing missing values instead of filling in with median values affects model results.

7 CONTRIBUTIONS

Both team members worked on cleaning up the data, structuring the dataset and providing the overview of the data.

Hongtao worked on using TensorFlow to implement the neural networks and doing research on neural networks to see how they can be used for real estate economics. Andy worked on creating the baseline models and their metrics to provide a baseline for the neural network to evaluate against.

REFERENCES

K. Fabricius G. De'Ath. 2000. Classification and Regression Trees: A Powerful yet Simple Technique for Ecological Data Analysis. *Ecology* 81, 11 (2000), 3178–3192.

Kaggle. 2017. Sberbank Russian Housing Market. (2017). https://www.kaggle.com/c/sberbank-russian-housing-market

TensorFlow. 2017. TensorFlow Wide&Deep Learning Tutorial. (2017). https://www.tensorflow.org/tutorials/wide_and_deep

XGBoost. 2015. Introduction to Boosted Trees. (2015). http://xgboost.readthedocs.io/en/latest/model.html