

CS583A: Course Project

Sushmith Ramesh (10443199), Revathy Ramasundaram (10444389)

May 19, 2019

1 Summary

We participate in an active competition of predicting house prices based on a set of given feature data such as the year built, neighborhood location, and overall condition. The final model we choose is a deep neural network that takes a 1460 x 81 matrix (each row is a house and each column is a feature of the houses) as input and outputs the predicted house prices. We implement the deep neural network using Keras and run the code using Google Collab. Performance is evaluated based on the mean squared error or mean absolute error of the model. In the public leaderboard, our score is 0.14638; we rank 2833 among the 4581 teams.

2 Problem Description

2.1 Problem

The problem is to predict house prices based on a set of given features about the house. This is a regression problem. The competition is at <https://www.kaggle.com/c/house-prices-advanced-regression-techniques/>. This problem explores the impact of various aspects/additions (exterior quality, inclusion of a fireplace, pool area) of a house on its actual sale price (this is useful information to both buyers and sellers). While some of these features may have an expected influence on the price, others may come off as a surprise with very minimal to no effect on the final price.

2.2 Data

The data is given as a 1460 x 81 matrix (this is given as a CSV file). The number of train sample is $n = 1460$. There are 38 continuous features and 43 categorical features. Of the 38 continuous features, we would like to predict the “Sale Price” feature of the house (thus we have 37 continuous features with the sale price acting as the “label”).

2.3 Challenges

The provided data is quite widespread in terms of range and therefore is difficult to compare and utilize with mathematical models (there is the question of how to deal with the

categorical features in case we want to utilize them in a mathematical model). Additionally, even among the continuous features, the data is still widespread and not immediately comparable. For instance, the “overall condition” features seems to range from 1 to 10, whereas the “year sold” typically range in the thousands (1900’s - 2000’s approximately). Thus even in case of utilizing all the continuous variables in a mathematical model, some form of rescaling may need to be employed.

3 Solution

3.1 Model

The final model we chose was a standard deep neural network. A description of the a standard deep neural network is online:

- <https://towardsdatascience.com/training-deep-neural-networks-9fdb1964b964>

- <http://neuralnetworksanddeeplearning.com/chap1.html>

3.2 Implementation

The dataset is a mix of numerical, and categorical variables. The categorical variables were transformed into one hot encoded vectors [1]. Now the total number of categorical features become 149. The features were normalized with mean and standard deviation. We implemented a regular deep neural network using Keras, and Tensorflow as the back-end. The code is available in github as a jupyter notebook file: https://github.com/revathyrams/House_Price_Prediction/blob/master/house_price_prediction_with_deep_neural_nets.ipynb. It takes about 1 hour to train the model, along with 5-fold cross-validation.

3.3 Deep Neural Network Model

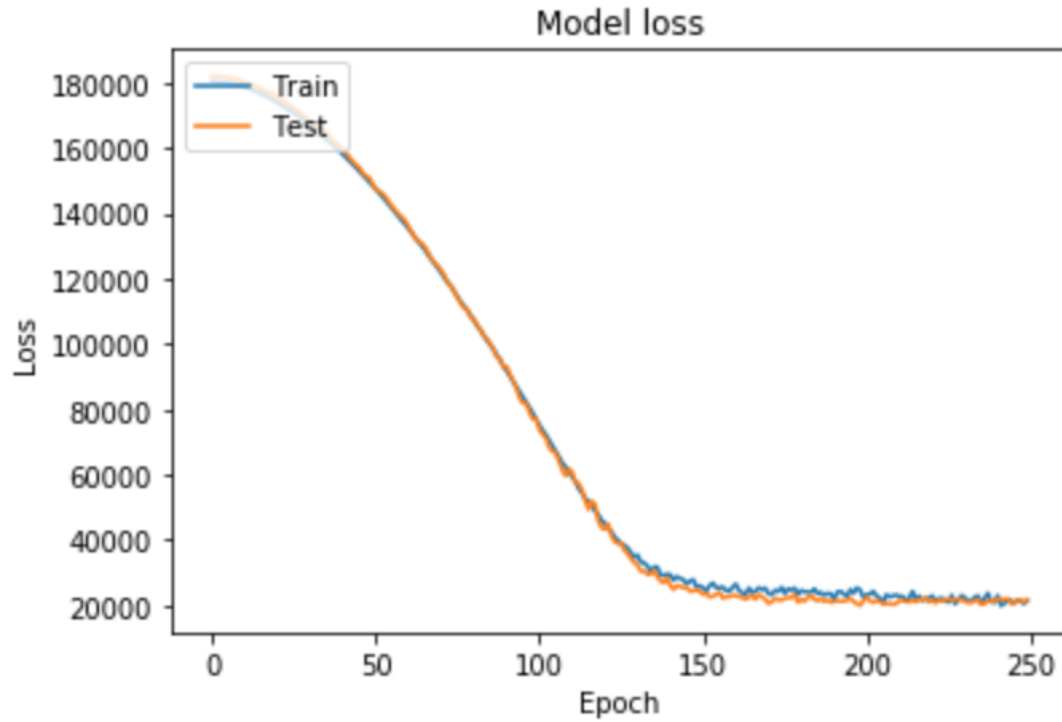
The model was trained with 1 input layer, 6 hidden dense layers and one output layer. The hidden layers were batch normalized, and regularized. The best validation loss achieved is 19,155. The actual target values in the training set range between 34900 - 755,000.

3.4 Settings

The loss function used was ‘mean absolute error’. The optimizer used was ‘nadam’.

Hyperparameter	Tuned Value
Epochs	250
Batch Size	16
Learning Rate	0.002

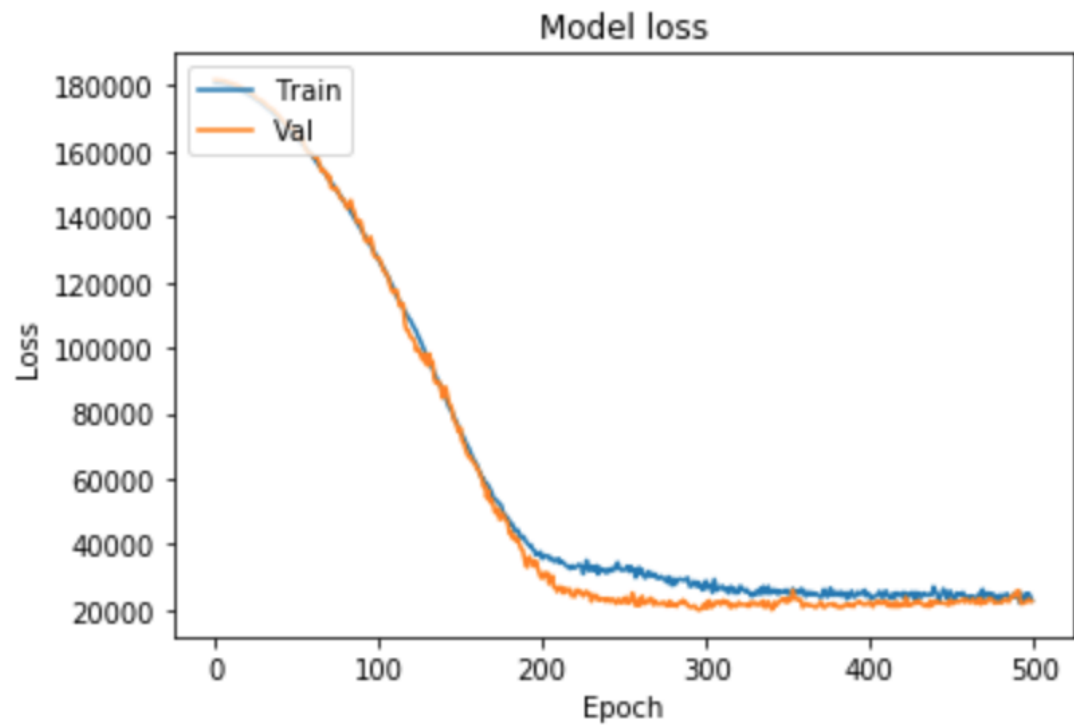
These hyper-parameters were found using cross validation.



3.5 Cross-validation

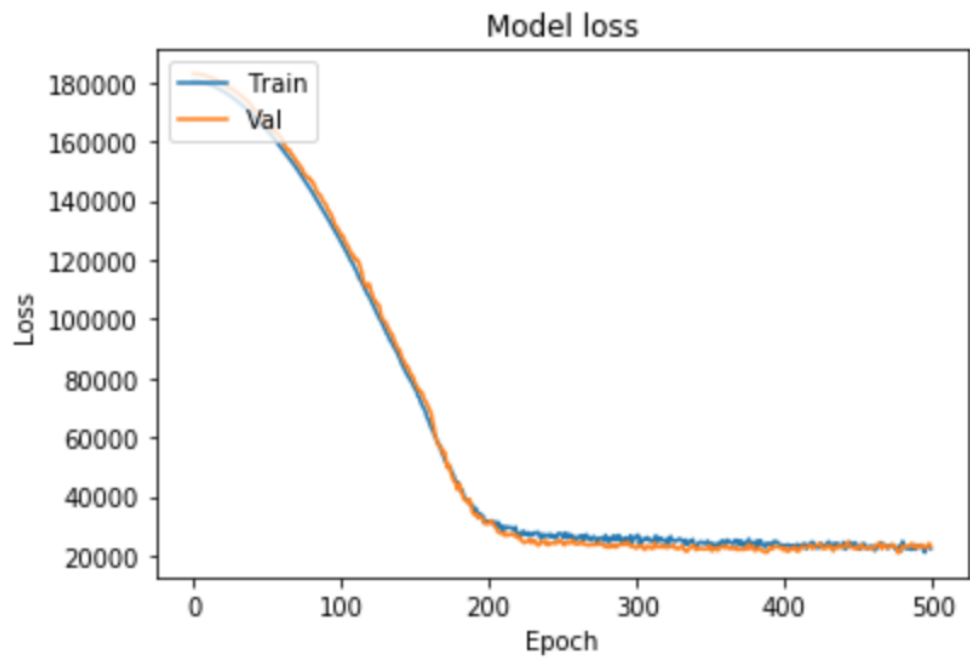
We used 5-fold cross-validation [2] method to tune the parameters. The sample size is 1460, so the number of samples in each fold were 292. The code was run on a hosted GPU instance of Google Colaboratory.

Iteration 1:



Iteration 2:

processing fold # 1



4 Compared Methods

We attempted two other methods, which we used as baselines for our final model (you can find these on GitHub: https://github.com/revathyrams/House_Price_Prediction/tree/master/baselines). One of the two was also a standard deep neural network (we'll call this the baseline standard neural network), but the preprocessing of the data was done differently. For this neural network, there were 11 dense layers, which were batch normalized and regularized and the optimizer that was used was "keras.optimizers.Adam". The other was a standard linear regression model (we compute w , the weight, using matrix computations). Both of these models did not perform as well as our chosen model and comparing between the two, the baseline standard neural network performed better than standard linear regression.

The preprocessing that was done for the two baselines involved only utilizing the continuous features of the dataset (so we end up ignoring the categorical features) in addition to removing a few of the data points that were not as useful (these outliers were determined using the IsolationForest module from sklearn). Thus, for the baselines, we end up with a 1314 x 38 matrix instead of the original 1460 x 81 matrix. Also, as mentioned in the challenges section, we apply the MinMaxScaler() from sklearn to rescale the data so that the continuous features are comparable and not too widespread. In terms of performance, the baseline standard neural network resulted in a MSE of 0.000418 as compared to a MSE of 0.003167 from the standard linear regression model (these are based on the training data which was split into training and testing data).

As discussed previously, it was difficult to compare these two baselines to our final chosen deep neural network model, as the preprocessing was different (in our chosen neural network model, the categorical features are utilized by transforming them using one-hot encoding and the MinMaxScaler() was not used, so the data and calculated MSE values were very different). We did post these baseline solutions to the Kaggle competition to see the ranking, which can be seen below.

Result of baseline standard neural network:












Overview	Data	Kernels	Discussion	Leaderboard	Rules	Team	My Submissions	Submit Predictions
3073	JonathanToro						0.15172	1 21d
3074	edwo						0.15173	6 1mo
3075	Alexey K						0.15189	5 25d
3076	sossage						0.15194	3 25d
3077	Francesco Fabrizio						0.15195	1 1mo
3078	Sushmith Ramesh						0.15195	8 2d
Your Best Entry ↑ Your submission scored 0.15408, which is not an improvement of your best score. Keep trying!								
3079	raosneha321						0.15199	1 2mo
3080	Tony Chen						0.15199	1 5d
3081	masaya kondo					</> kernelb90ce12d41	0.15201	4 1mo
3082	Saravanan srinivasan						0.15202	3 2d
3083	Josh Matlock						0.15202	1 2mo
3084	mnor169						0.15202	1 22d
3085	Thomas Gao						0.15206	1 5d

Result of standard linear regression:

Overview	Data	Kernels	Discussion	Leaderboard	Rules	Team	My Submissions	Submit Predictions
3431	vivekkr970						0.16626	1 11d
3432	Maximiliano Padulo						0.16626	5 1mo
3433	Ryoji Yamada						0.16630	3 9d
3434	karenne						0.16637	5 6d
3435	Data Science Brazil						0.16639	1 2mo
3436	Zenith Cosmos						0.16643	3 1mo
3437	Sushmith Ramesh						0.16651	1 -10s
Your Best Entry ↑ Your submission scored 0.16651, which is not an improvement of your best score. Keep trying!								
3438	Kartikey Sarode						0.16655	3 2mo
3439	Juan Acevedo						0.16656	1 2mo
3440	HVT VH						0.16661	7 1mo
3441	Agent 47						0.16665	7 2mo
3442	Abhishek001						0.16671	12 1mo

5 Outcome

We participated in an active competition. Our score in public leaderboard is 0.14638. We rank 2833 among the total participants of 4585 in the public leaderboard. Private leaderboard scores are available only when the competition ends.

2828	mann96		0.14627	1	22d
2829	Kartikey Riyal		0.14627	1	22d
2830	Bchayma		0.14629	16	1mo
2831	AI subject		0.14631	9	2mo
2832	Bentarou Kurume		0.14636	2	1d
2833	Revathy Ramasundaram		0.14638	13	now
Your Best Entry ↑ You advanced 8 places on the leaderboard! Your submission scored 0.14638, which is an improvement of your previous score of 0.14655. Great job!  Tweet this!					
2834	Phoebe Shi		0.14639	1	2mo
2835	sudharshan rajendhiran		0.14642	4	22d
2836	liuzuolin		0.14643	10	1mo
2837	AT081196		0.14649	2	2mo

6 References

- (1) deeplearningbook.org, May. 18, 2019
- (2) François Chollet, Deep Learning with Python, November 2017