Sidarth Raman
Machine Learning CSCI 1420
04/27/2023

Real-Estate Price Prediction Model

GitHub Repository

Problem:

Main Objective: The problem I attempted to solve was to create an accurate real estate pricing prediction model using linear regression. To solve this problem I used data from the State Department of Revenue (DOR) publication of data to the Somerville Board of Assessor database. The data provided was real-estate data on Somerville, Massachusetts, a suburban city with a population of just over 80,000 people. The dataset contained 107,900 unique data points each with 71 columns of information pertaining to the house. I will use this data to create a model, and then train the model on my data, and eventually test my model. My hope is that with the given data, I can accurately predict the price of the houses in Somerville, within a given threshold percentage. I also will use this model to draw more conclusions about the real estate market.

Secondary Objectives

1. What influences house prices the most? I also plan on doing some further investigation into the greatest influences on house prices. As the area, to an extent, is held constant, it will be interesting to see the resulting coefficients on the variables I choose, as they will indicate the correlation between them and house prices.
2. Can real-estate pricing be classified in a linear way? I plan on doing an analysis of the model itself as well. If it can be classified linearly the difference between a house exactly half the size, bedrooms, square footage, etc. of another will be exactly half.
3. What changed over time? Since I have data from before 2000 to 2020, I can see if there what differences there are in the model if trained on different time periods.

Approach:

To solve my problem, and create the model, I will utilize Machine Learning, specifically TensorFlow and Keras' Linear Regression.

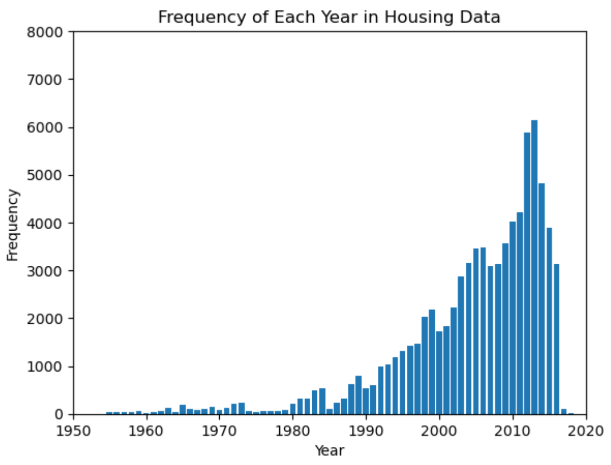
Preprocessing: During my initial approach, I began with heavy pre-processing which was necessary to only select information that was useful and pertinent to the problem. Of the 71 columns only a few proved to be helpful. I selected the following to be used in my model.

Column	**Description**
LAND VAL	Value of the land that the house is built on.

IMPROVE VAL	The value an assessor places on the property. (the structures, the streets, the sewer connection, etc.)
SQFT	Square footage of the house.
ROOMS	The total number of rooms.
BEDROOM	The total number of bedrooms.
BATH	The total number of bathrooms.
LIVING AREA	Total finished living area. Not to include open porches, balconies, terraces carports, or garages.
GROSS AREA	The total area includes common circulation areas, such as corridors, balconies, etc.
YEAR	The year the home was built.
PERCENT GOOD	A grade given by the property owner to describe the state of the house.
STORIES	The total number of stories in the house.

Note: I decided not to use "Parcel Val" as Parcel Val is the sum of the land value and the house price, and therefore would be too closely correlated to the actual label.

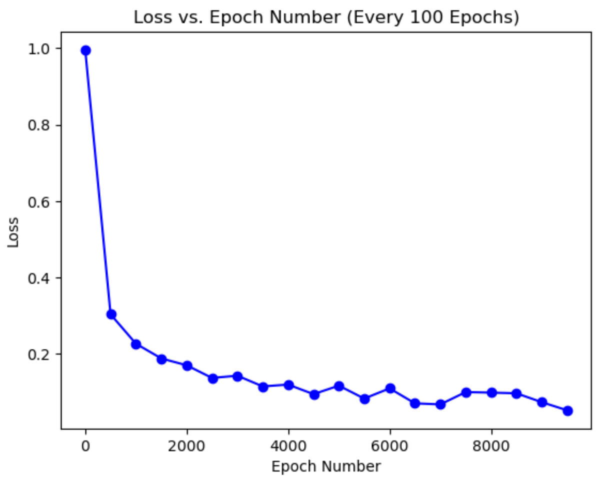
Data Removed: I went ahead and removed any house with "Sale Price" that was under $100,000. This is because many of the databases were insignificant areas of land that should not be under the category of real estate. I also used "SALE DATE" between 2010 and 2015, as this was the most common data, and if I created a time interval too large then the linear regression wouldn't be fitting for the entire period. (*See graph for details*). I also, obviously, removed 'null' and '0' values in my dataset as this would throw off the linear regression.



Model:

Initially, I created a very basic model with not many features, upon running the program I was able to make fixes to improve the model.

1) I began by adding more dense layers, as the data had many fields it was learning for, adding three more dense layers helped fit the complexity of my data.
2) I then added ReLu activation instead of sigmoid activation as there are lots of studies showing the convergence of ReLu is far better than that of Sigmoid. It is important to note that almost any activation function does take away from the linearity of the model.
3) I then increased my model to 10000 epochs as the loss was not converging on just 1000 epochs.
4) To counter overfitting I added a kernel and bias regularizer with L2 regularization.
5) I played around with hyper-parameters and increased the learning rate as the learning was still converging even after increasing the epochs.



Initial Model (Keras Linear Regression):

- ❖ Single Dense Layer
  - ➢ Sigmoid activation
- ❖ MSE loss
- ❖ Stochastic Gradient Descent (Learning Rate of 0.01)
- ❖ Accuracy (Within 20% of the actual price)
- ❖ 100 epochs
- ❖ 80/20 split on training and testing data

Final Model (Keras Linear Regression):

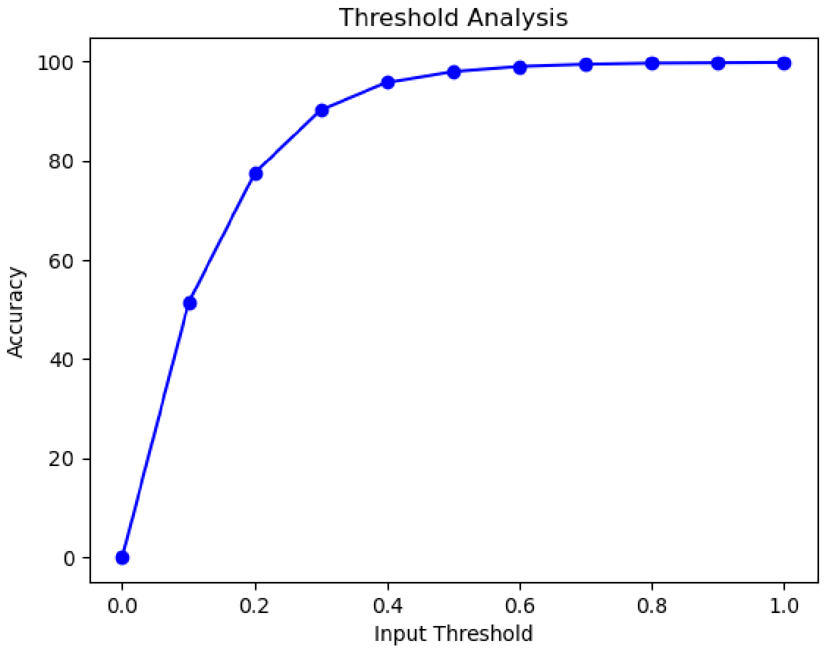
- ❖ Four Dense Layer
  - ➢ ReLu activation
  - ➢ (128, 64, 32, 1 units)
  - ➢ Kernel Regulizer (L2 Regularization with a coefficient of 0.01 on the Final Dense Layer)
  - ➢ Bias Regulizer (L2 Regularization with a coefficient of 0.01 on the Final Dense Layer)
- ❖ MSE loss
- ❖ Stochastic Gradient Descent (Learning Rate of 0.02)
- ❖ Accuracy (Within 20% of the actual price)
- ❖ 10000 epochs
- ❖ 70/30 split on training and testing data

Results:

Training Accuracy: 86.19%
Testing Accuracy: 77.53%

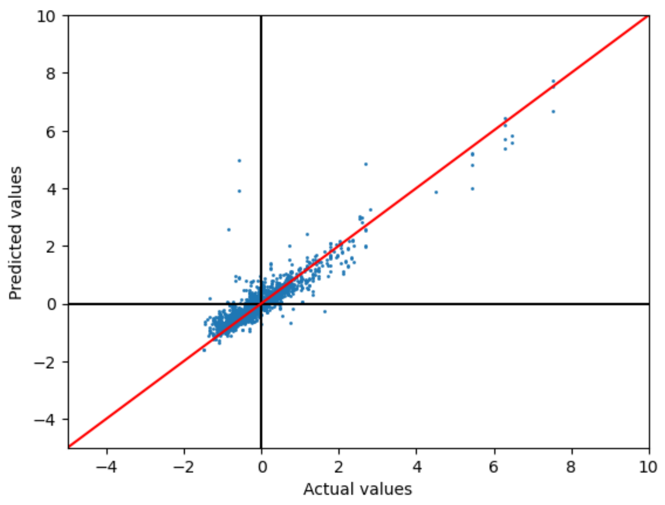
Results Summary

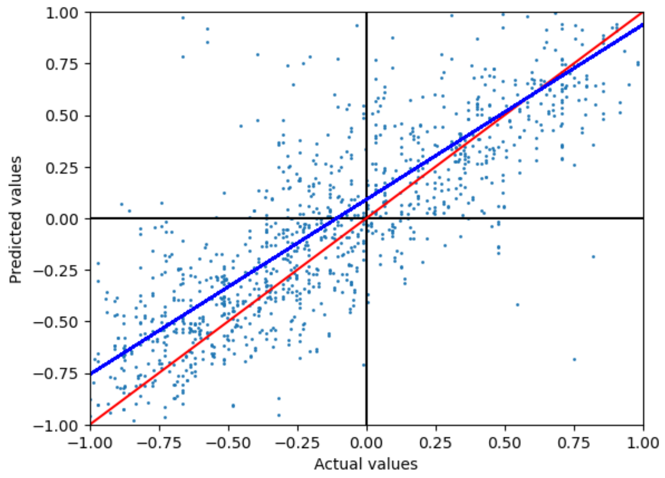
The results of the model were based on a 5-year range between 2010 and 2015. This is because the largest 5-year subset for the dataset was between these years. I also kept the threshold at 0.2. This is because, between 2010 and 2015, there was approximately a 20% difference between the median of the 5-years and the highest and lowest quarters of the 5-year span.[1] The graph on the right is the accuracy of the testing dataset with different thresholds.



Actual vs. Predicted Values

Here is the scatter plot of the predicted versus actual value. On the left, you can see the entire dataset, and on the right is the zoomed-in graph at the origin. You can see that the data is fit fairly well and there are a few outliers, but not many. We can also see that as homes become more expensive, it becomes less linear and the model seems to underestimate the value of the homes, which can show that the model isn't very linear. The blue line here shows a linear line of best fit for the data and we can see that it is shifted a little off the origin, and therefore the slope is also adjusted slightly.

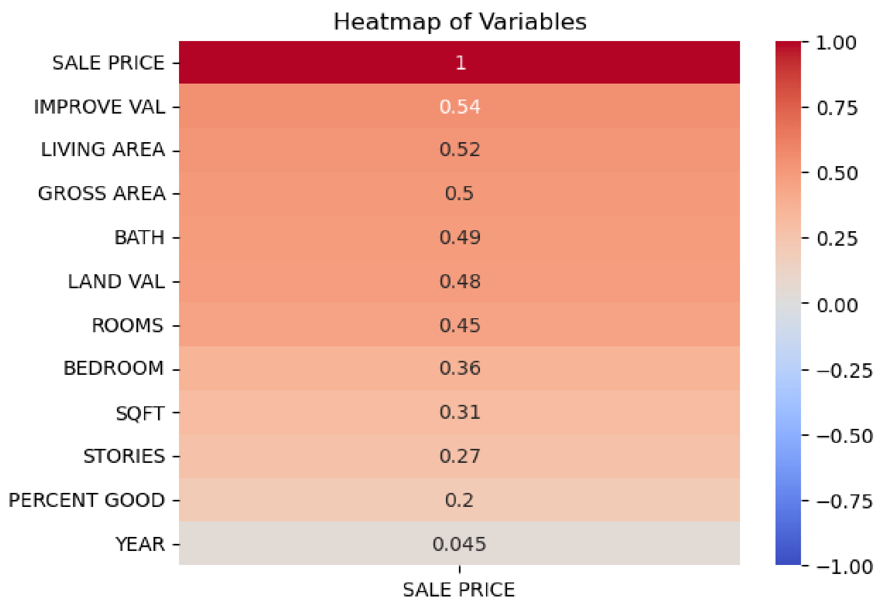




[1] https://fred.stlouisfed.org/series/ASPUS

Correlation Coefficients

One other thing I wanted to test was the correlation between the features and the Sale Price. It was interesting to see that none of the features, on its own, was very highly correlated with the Sale Price, but they still combined to give effective data. This is probably due to the fact that by adding feature sets, there could be linear and non-linear combinations of these features. We initially expand the units to 128 with a ReLu activation which means we are, in a way, adding dimensions to the feature set.



Note on Non-Linearity

The reason I was able to achieve this accuracy was because of the additional non-linear features I added. When I tried removing any non-linear activation function (*removed ReLu*), I was surprised to see a 28.59% and 21.22% drop in training and testing accuracy respectively. This is because then the model would just be a linear combination of the inputs, none of which have a very high correlation with the Sale Price.

Future Implementation:

If I was able to improve this project further, one thing I would change is the scope of my model. I would scrape Realtor.com for data on all houses in the United States and have a more general model for all houses in America, and while accuracy would be sacrificed, I would have a better idea of the housing market in the United States. I would also be able to create smaller sub-models to see patterns in certain large cities. Comparing the coefficients of the variables in New York, NY, and Henderson, NV might show general trends on what drives up prices in different areas of the country. This analysis would be interesting on a larger scale.

I would also attempt to solve the same problem using different models. I think a simple Neural Network Model would fare very well with this problem as it has been very accurate in other prediction models. Although the data isn't very complex, it will still be useful as it can learn non-linear relationships very well.