CSC 180-01 Intelligent Systems (Fall 2023)

Project 3: Prediction House Prices using Tensorflow

William Moosakhanian, ID# 302537495

Xiong Moua, ID# 220278332

Due: October 27, 2023

Problem Statement

The problem statement for this project is to train a model that can accurately predict the price of a house by using both textual and visual data.

Methodology

For this project, the preprocessing and model creation was much less complicated than the previous projects. For starters, Professor Chen gave us the code for preprocessing the images and the numeric data was processed in the same manner as the previous projects. We essentially created a number list that kept track of the houses that were within the constraints of our problem statement. We wanted to remove the outliers to give the model a more consistent data set to work with. For the first model, we only kept houses between 100,000 and 900,000 and used the number list to find the corresponding pictures. We one-hot encoded the categorical data, zip codes, bedrooms, bathrooms, and normalized the numeric data, the square footage of the houses, or the area. Afterwards we dropped the num column because the model didn't need it and it was only there for data preprocessing. We also dropped the price column because that's what the model is supposed to predict, and giving it the price made the model overfit almost perfectly. For the additional features we copied all the data preprocessing and tweaked the models to make them more in line with the results we wanted to achieve.

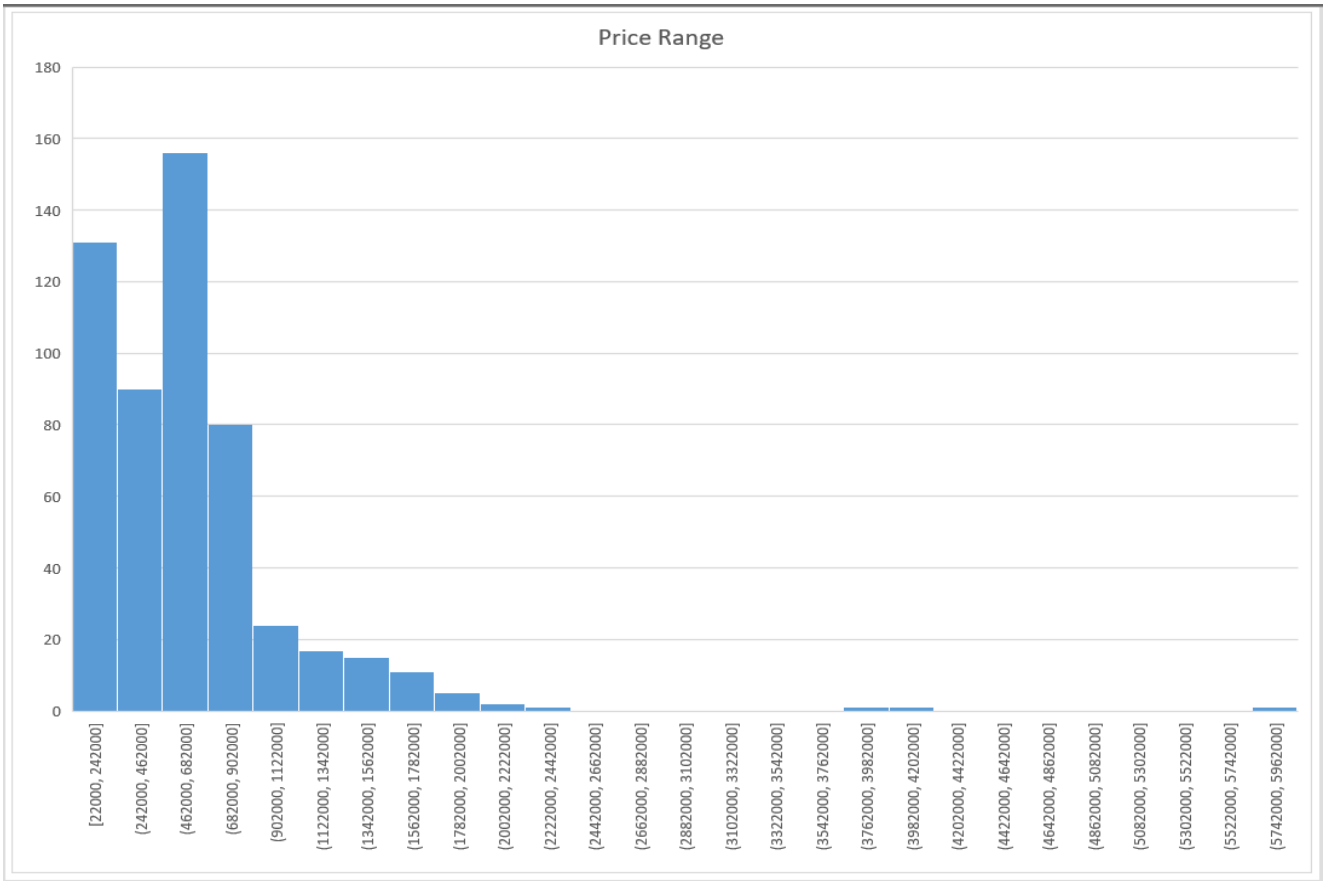
Experimental Results and Analysis

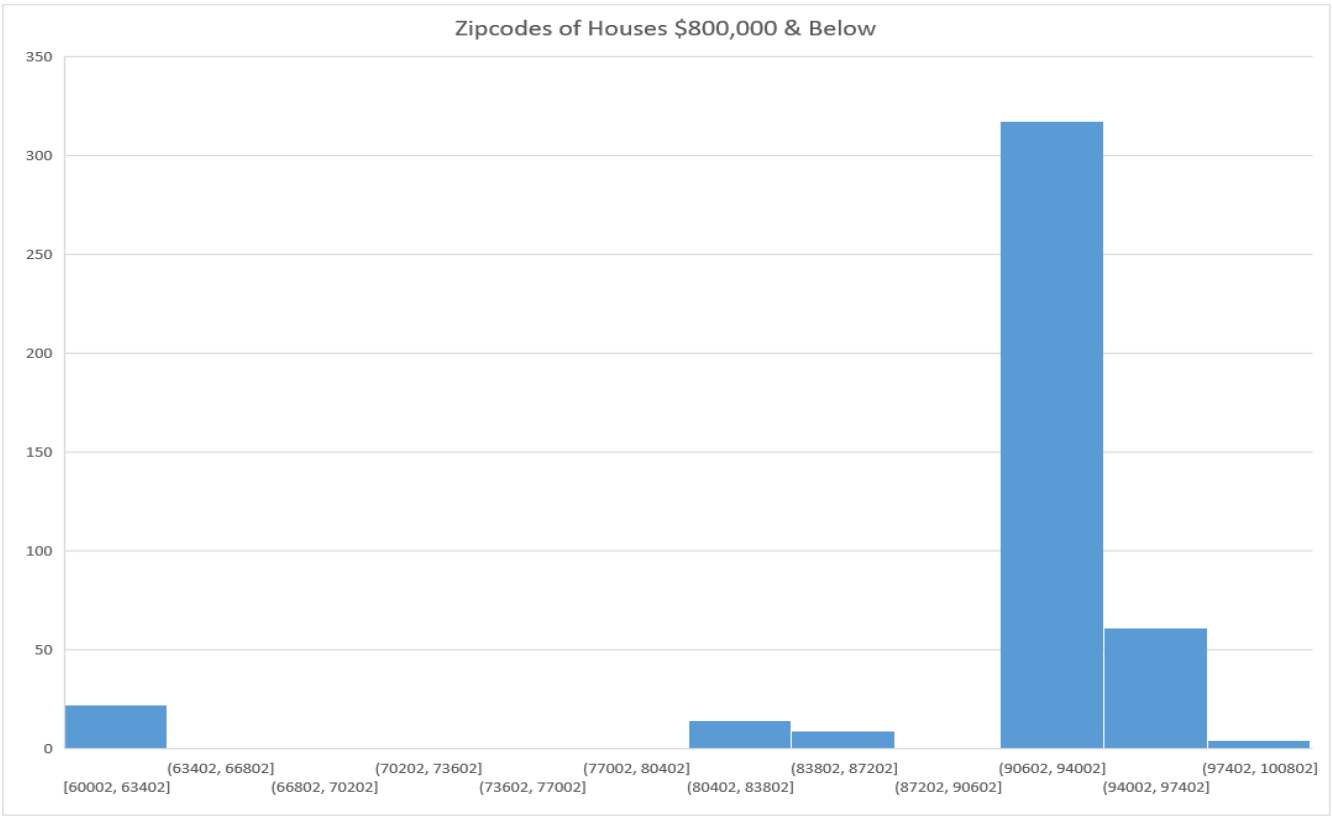
Model	CNN	NN	Activation	RMSE
Basic Model	2x Convolution 2x MaxPooling2D	4x Hidden Layers:128,64,32,16 2x Hidden Layers(Merged):10, 10	relu, linear (output layer)	135239.143
Model w/Dropout	2x Convolution 2x MaxPooling2D 2x Dropout	5x Hidden Layers: 512,256,256,256,128 4x Hidden Layers(Merged):128,64,32,16	relu, linear (output layer)	122235.429
Model w/Dropout and Outliers Removed	2x Convolution 2x Maxpooling2D 1x Dropout	5x Hidden Layers: 512,256,256,256,128 4x Hidden Layers(Merged):128,64,32,16	relu, linear (output layer)	91075.618

Our initial model performed quite terribly, and used convolution2D and MaxPooling2D twice for the CNN, with the convolution layers using a 4x4 kernel and MaxPooling2D using a 2x2 window. Afterwards we flattened the output and made our neural network model. This contained four layers, with 128 in the first, 64 in the second, 32 in the third, and 16 in the fourth. Every layer for the CNN and neural network used relu as the activation function prior to the concatenation. Then we merged the two models and made three more layers using the merged model. The first two used 10 neurons each with relu while the output layer used 1 neuron and the linear activation function.

After the model finished the training process, we had it predict the values and discovered that it was predicting in a completely random pattern. We came to the realization that our model was doing this because it was too simple. With more fine tuning, we realized that if we increased the patience from 2 to 10, added a few more layers, and adjusted our learning rate, our model would produce more accurate results. With this we decided to move onto the additional features to see if there was anything we can do to increase the accuracy of our model.

For our first additional feature, we implemented dropout layers to our CNN model. The dropout layers were added after each pooling layer in the CNN input model. The first dropout layer was 0.25 and the second was 0.125. This resulted in our model having a slight decrease in the RMSE by about 4000. This means that our model was more accurate compared to the basic model. With this we decided to implement our second additional feature. For this additional feature we created a chart of the zipcodes and the price ranges of all the houses.





First we looked at the price ranges of the entire dataset. This chart showed us that the majority of the houses were within the $22,000 to $902,000 range. With this information we had new min and max values for the price range. After this we created a graph for the zipcodes of the house prices within our new price range. With this filter we were happy with our new dataset to train the model with.

Task Division and Project Reflection

The task was divided evenly among the two members of the team. Xiong was tasked with the data preprocessing with the visual data and the textual data, along with creating the model that takes in two data inputs and outputs only one result. Will also helped with the creation of this model. Once the basic model was functioning, Will fine tuned the model for better results. After we were satisfied with our results we began working on the additional features. Overall Xiong and Will worked closely together throughout this entire project.

Some important things we learned for this project were the limitations of Google Colab and Kaggle Notebook and understanding how models with multiple inputs were created. Starting off with Google Colab, this was our first attempt at the project because we wanted to take advantage of their GPU for this project. However we quickly ran into some issues because the cell responsible for merging the images for each house together would take too long to finish compared to our own machines. We decided to move over to Kaggle Notebook because we thought it would be better than Google Colab. Shortly after deciding to move over to Kaggle Notebook, we ran into an issue with the size of data that we can input into Kaggle Notebook. On

a free account we were limited to a data size of 1000. This meant that our data would be cut in half resulting in a model that would be highly inaccurate. Between Google Colab's rather slow processing power and Kaggle Notebook's low data size, we ultimately decided to run this project on our own machines. For future projects, and our final project, we will be taking the time to make sure that Jupyter Lab will be taking full advantage of our 2080 GPU. Moving onto the knowledge that we gained for this project, we learned that creating a model that allows multiple input layers was rather helpful. This means that for future projects, we don't have to merge the data into one massive numpy array. We can create a model that allows multiple different inputs instead. Furthermore we learned that the more complex we made our model, we got better results for this particular use case. When the model was too simple, it tended to perform more randomly and produce bad results.