AIRBNB Price Prediction

By:
Saad Saleem
&
Sakif Khan

Section 1:

The problem we have identified is a regression problem. The problem we are solving is given an Airbnb listing's attributes (name, location, room type, house rules, host info etc.), the models will predict its price per night. The target variable is log price, which helps deal with skewed values for price dominating dataset.

This problem is interesting as predicting Airbnb price is very difficult, as there are so many factors to consider, so it proves to be a fun challenge. A good model could also prove to be very useful, to both consumers and hosts, as it can help them identify fair/competitive prices for listings.

Our dataset was found on Kaggle (link: Airbnb Open Data). The original dataset is around 100,000, but after selecting relevant features and dropping incomplete rows, the working set we used is ~39,000 entries. A limitation of this, is that dropping rows in this fashion may introduce selection bias, as certain types of listings might have certain fields invalid.

Section 2:

For the task we identified, we did not need to use an encoder to get numerical vector embeddings as our data is already numerical and not raw text after feature selection/engineering. The input to the pytorch network consists of 9 numerical/binary features: 'host_verified', 'instant_bookable', 'Construction year','service fee','minimum nights','number of reviews','reviews per month', 'review rate number', and 'availability 365'. Prior to training the features were standardized with StandardScaler, which uses scale from training split only, avoiding potential data leaks.

The network itself consists of a 2-layer MLP regression head that was implemented with torch. The first layer projects the 9 input features into a 64 dimension hidden representation, followed by ReLU activation function. The second layer maps this representation into a single scalar output (log price) which is the prediction.

The model was trained using mean squared error as the loss function, computed on log price, and Adam as our optimizer with a learn rate of 0.001. The batch size was also set to 64 over 50 epochs, with train loss computed as the average MSE every epoch. The number of epochs was determined by looking at the loss and R^2 curves. As seen in figures 1 & 2 below, the loss drops off quite quickly and stabilizes while the R^2 value rises very high after a short number of epochs, with no sign of divergence. Given that the model had converged fairly early for both curves, 50 epochs were selected to be conservative and sufficient for our model.

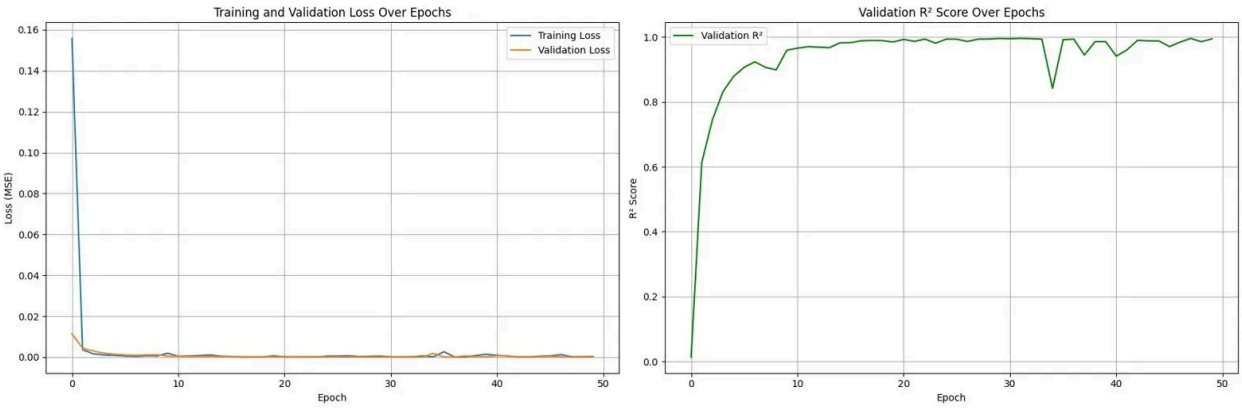


Figure 1: Loss and validation graph/ Validation over Epochs

```
Epoch   1/50 | Train Loss: 0.1558 | Val Loss: 0.0114 | Val R²: 0.0138
Epoch  10/50 | Train Loss: 0.0019 | Val Loss: 0.0005 | Val R²: 0.9593
Epoch  20/50 | Train Loss: 0.0007 | Val Loss: 0.0002 | Val R²: 0.9846
Epoch  30/50 | Train Loss: 0.0005 | Val Loss: 0.0001 | Val R²: 0.9954
Epoch  40/50 | Train Loss: 0.0014 | Val Loss: 0.0002 | Val R²: 0.9861
Epoch  50/50 | Train Loss: 0.0003 | Val Loss: 0.0001 | Val R²: 0.9944
```

Figure 2: MLP Model output while training

Section 3:

For the LLM test we used OpenAI's GPT-4o, accessed via the OpenAI Python API. GPT-4o was chosen over smaller models like GPT-3.5-turbo due to its stronger numerical reasoning and instruction following capabilities, which are important for a regression task where the model must output a precise number.

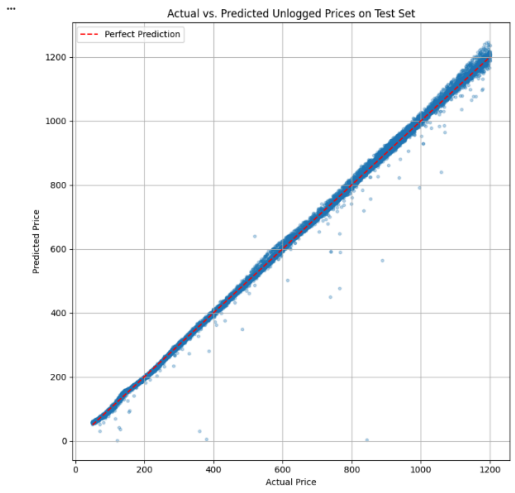
We first tried zero shot promoting, where we simply described the task and gave the model the values to work through, this resulted in values that were set around the median of the data set showing that the model really didn't know what it was doing. We then switched to a few shot prompting technique where we used 25 stratified examples that were selected from the dataset to help the model better understand what it was working with. The data was stratified to ensure the model saw the entire range of the dataset. The following prompt was used on the model:

"You predict nightly Airbnb prices in New York City.","Below are 25 real listings with their actual prices to calibrate your predictions.", "Use these as reference. When asked, respond with ONLY a single integer, no symbols, no explanation."

Temperature was set to 0 to ensure clear output and reproducible predictions. The model's response was parsed using a regular expression to extract the first integer from the output, handling any cases where the model included symbols or text despite instructions. The model was evaluated on 100 randomly sampled test examples to manage API rate limits of 30,000 tokens per minute.

Section 4:

Both models were evaluated using R^2, Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and a custom accuracy metric, which measured how accurate a model was based on if the difference between the predicted and actual price differed less than $50. These metrics were chosen because they cover: how much variance a model accounts for, the average error of the model, and a real world example of an evaluation metric. All of this tells us how often a model would give us a useful prediction. After running the models the figures 4&5 were formed, shown below:



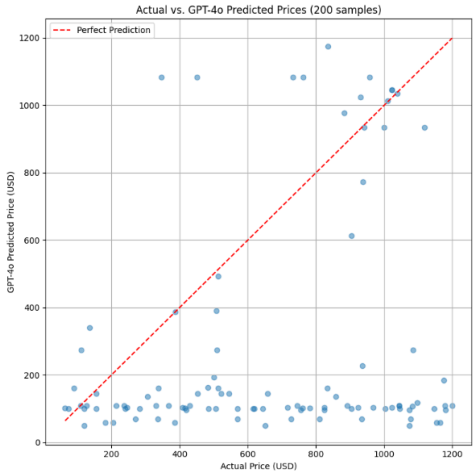


Figure 4: MLP Graph **Figure 5: LLM Graph**

As we can see from the graphs, the MLP model was almost perfect, whereas the LLM was somewhat accurate, however it predicted significantly lower prices almost all the time compared to the actual higher values. We can see the models statistics compared side by side in figure 6 below:

--- Model Comparison ---

	Metric	MLP Model	GPT-4o LLM
0	R^2	0.9982	-1.7902
1	MAE ($)	6.25	450.95
2	RMSE ($)	13.90	569.56
3	Accuracy (within $50)	99.74%	13.00%

Figure 6: Side by side comparison

The MLP achieved an R^2 of 0.9982, where it accounted for about 99% of the variance in the data. It also had an MAE of $6.25 and RMSE of $13.90, at the end it achieved a 99.74% accuracy, it is important to note that this high accuracy may be due to the custom metrics range, its possible if we lowered the threshold the accuracy might change.

The LLM achieved an R^2 of -1.79, MAE of $450.95 and RMSE of $569.56 with a 13% accuracy across the 100 examples it predicted. The negative R^2 shows that the model performed worse than randomly guessing or even just using the mean to predict price. This also tells us that the features we choose do not have significant predictions for the LLM, where it may need more context.

Fail Cases:

Fail cases for both models can be seen in Figure 7&8 Below:

	NAME	neighbourhood group	room type	minimum nights	number of reviews	Actual Price ($)	Predicted Price ($)	Error ($)	Abs Error ($)
0	Historic Designer 2 Bed. Apartment	Manhattan	Entire home/apt	999.0	2	805.0	40410.121094	39605.121094	39605.121094
1	Clean 1br w/Parking Incl. also 10min near JFK...	Queens	Private room	458.0	34.0	483.0	3023.510010	2540.510010	2540.510010
2	Stunning 1 Br, West Village Luxury w/ great views	Manhattan	Entire home/apt	365.0	15.0	740.0	2330.679932	1590.680054	1590.680054
3	Cozy Home in Harlem ✨	Manhattan	Entire home/apt	825	23.0	303.0	1650.699951	1347.699951	1347.699951
4	Luxury dmnn Bldg + Empire State Views & Roof Top!	Manhattan	Entire home/apt	365.0	7.0	765.0	2073.760010	1308.760010	1308.760010
5	Brooklyn Charm, Close to Manhattan (30+ Days O...	Brooklyn	Entire home/apt	200.0	314.0	284.0	516.000000	232.000000	232.000000

Figure 7: MLP Fails

Listing	Actual	Predicted	Error
The Swissôtel Chicago	$ 350.00	$ 1187.75	+837.75
Belmont Pleasures	$ 1597.00	$ 890.94	-706.06
Victorian Classic	$ 350.00	$ 95.72	-254.28
The Dorm	$ 249.00	$ 98.91	-150.09

Figure 8 MLP Fails on custom data

	NAME	neighbourhood group	room type	minimum nights	number of reviews	Actual Price ($)	Predicted Price ($)	Error ($)	Abs Error ($)
0	Space & Comfort in Victorian Brooklyn home	Brooklyn	Entire home/apt	3.0	5.0	1183.0	52.0	-1131.0	1131.0
1	Sunny Spacious room near central park	Manhattan	Private room	1.0	163.0	1108.0	104.0	-1004.0	1004.0
2	Entire 2 Bedroom in a smart new house	Queens	Entire home/apt	4.0	93.0	1077.0	102.0	-975.0	975.0
3	Large furnished room with 2 beds	Queens	Private room	30.0	14.0	1004.0	58.0	-946.0	946.0
4	Brooklyn Style Hostel RM4 #4	Brooklyn	Shared room	1.0	9.0	972.0	58.0	-914.0	914.0

Figure 9: LLM Fails

For comparison custom data was gathered manually from Airbnb websites and the MLP model was tasked with predicting price with that data. The manually selected data was intentionally selected in areas outside of NYC, as all of the training data includes listings exclusively in NYC. As seen in figure 8, the model struggles to accurately predict the price. This could be due to multiple factors, like a few features were missing and had to be filled with default values, the manually selected listings being from different cities, and the fact that we chose these examples because they were such unique listings and were almost outliers by nature.

When looking at the models we see that the MLP severely over predicts and the LLM severely underpredicts. The LLM may act this way due to the very few examples it was trained on, given more resources (money/api calls) the model could have done better.

Conclusion:

This project was both challenging and enjoyable because it provided an opportunity to explore how different AI approaches perform on real world pricing data. Comparing traditional machine learning methods with modern large language models gave valuable insight into the strengths and limitations of each approach, especially when working with structured numerical datasets.

Contributions:

Saad- LLM model in colab notebook and Section 3&4 of report

Sakif- MLP model in colab notebook and section 1&2 of report