

HDB Resale Price Prediction

Marcus Chao, Athena Tan, Wen Xin Foo

March 2025

1 Introduction

The HDB market in Singapore is incredibly volatile, characterized by dynamic price fluctuations influenced by a multitude of interconnected factors, such as economic shifts, government policies, demographic trends, and market sentiment. Given this complexity, buyers and sellers face considerable challenges in navigating the housing landscape, necessitating robust, data-driven insights to make informed and strategic decisions.

Reliable predictions of HDB resale prices are crucial not only for prospective homeowners but also for real estate analysts and policymakers. Accurate forecasting models enable stakeholders to better understand market dynamics, optimize investment strategies, and formulate effective housing policies. Therefore, this project seeks to develop predictive models that provide clear, actionable insights into HDB resale prices, ultimately contributing to greater transparency and stability in Singapore's housing market.

Our approach to this project was to explore multiple different types of models whilst assessing their performance with respect to this problem statement, so that we could understand the limitations and advantages of each model in practice.

The codebase for this project can be found [here](#).

2 Primary Dataset

Our primary dataset is taken from a government source (Housing Development Board, 2025). The dataset includes historical HDB resale data from January 2017 till the current date, and is regularly updated. this dataset consisted of 381,165 entries, and contained the following columns:

Column	Data Type
month	object
town	object
flat_type	object
block	object
street_name	object
storey_range	object
floor_area_sqm	float64
flat_model	object
lease_commence_date	int64
remaining_lease	object
resale_price	float64

Table 1: DataFrame Columns and Data Types from Government Data

3 Exploratory Data Analysis

Before performing an exploratory data analysis (EDA), we found and removed 180,625 duplicates. Thankfully, there was no null data. The EDA conducted was fairly in-depth, analysing each input feature against the target feature. However, for conciseness, this report will only detail the more interesting plots.

3.1 Target Feature Analysis

Our Target feature was resale price. Thus, we plotted some graphs to understand its distribution and how it changed over time, since this is a Time-Scaled dataset. From the plots, we saw a roughly normal distribution of resale prices. Across time, the resale price has overall increased, but we see some cyclical trends as well (e.g. resale volume will decrease in the months of February – April)

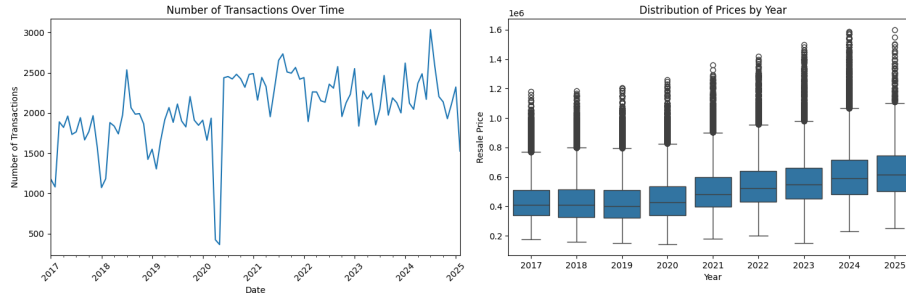


Figure 1: Target feature plots

3.2 Storey Range Analysis

By plotting the average resale price distribution across the storey of the HDB, we saw a clear trend; the average resale price increased as the storey range increased.

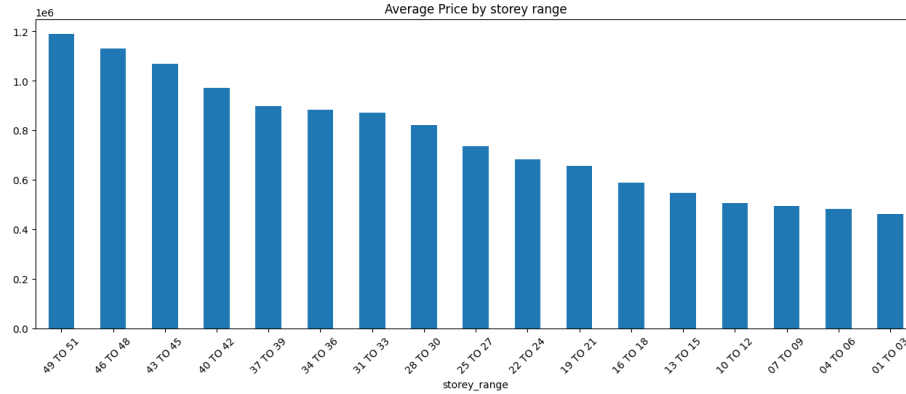


Figure 2: Storey Range plot

3.3 Floor Area and Remaining Lease Duration

We performed a scatterplot against the 2 numerical values within our dataset, the duration of the remaining lease in months and the floor area in squared meters. The floor area is represented by the x axis, and the remaining lease duration is represented by the hue of the points on the scatterplot. We can roughly see some expected trends; as the floor area increases, so does the resale price. Likewise with the remaining lease duration. In the scatterplot, we do see an outlier with significantly more square footage than the other points. We took note of this for our data cleaning process.

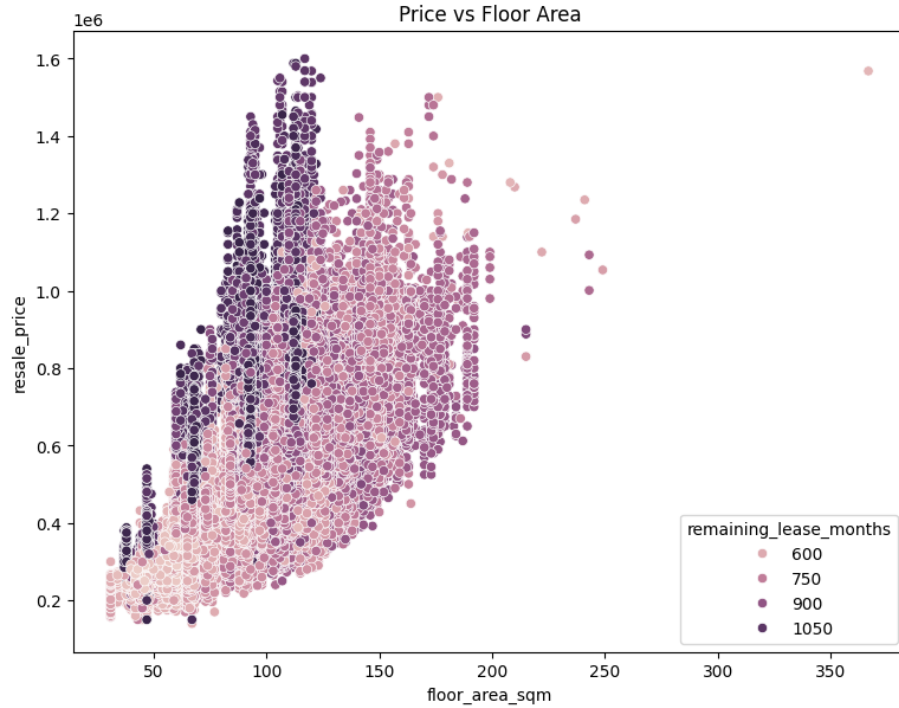


Figure 3: Floor Area and Remaining Lease plot

4 Data Preprocessing

Although the data was fairly clean, we still needed to adapt it to work properly with the models we decided to use.

4.1 Removing Outliers

Before we began our data cleaning, we focused on removing some outliers. As we are working with Multivariate data, we didn't want to remove outliers based on a single variable. Thus, we decided to utilise Mahalanobis distance to identify outliers based on multiple numerical variables (Sharma, 2023) .

	floor_area_sqm	remaining_lease_months	resale_price	mahalanobis_dist
182474	366.7	564	1568000.0	11.688276
200280	117.0	1038	1600000.0	7.101499
174508	112.0	1088	1588000.0	7.090145
197905	106.0	1020	1550000.0	7.060178
174615	113.0	1050	1580000.0	7.047303
...
148453	113.0	1135	1450000.0	6.051356
148368	113.0	1137	1450000.0	6.050331
183103	117.0	1023	1450000.0	6.048822
171900	172.0	808	1500000.0	6.019152
175710	97.0	1030	1370000.0	6.017186

Figure 4: Highest Mahalanobis Distance of Datapoints

Given the significant gap between the datapoint with the greatest mahalanobis distance and the rest of the dataset, we decided to remove this single datapoint.

4.2 Column Encoding

To prepare the data to train the models, we had to transform and encode some of the columns. From the original 11 columns, we only used 8 of the most relevant columns as features.

The **month** and **remaining_lease** columns, data was originally presented in years and months format. For these, we converted them to the columns **m_from_jan_2017** and **lease_months_left** columns, which are integer columns representing the number of months.

The **town** and **flat_model** columns represent categorical data, which we decided to encode into their different categories through one-hot-encoding.

The **storey_range** and **flat_type** columns represent ordinal data, which we decided to encode with integer label encoding. A higher storey range and higher-tier flat type was encoded as a higher integer.

Finally, we normalized all numerical data with the exception of the **month** and **remaining_lease** columns, as new input data might reside outside of that within the original data set.

month	→	m_from_jan_2017
2018-02		14
remaining_lease	→	lease_months_left
61 years 04 months		736
town	→	one-hot-encoding
ANG MO KIO		
flat_model	→	
Improved		
storey_range	→	label encoding
Ø1 to Ø3		Ø normalised
flat_type	→	label encoding
2 ROOM		1 normalised
floor_area_sqm		
44	normalised	
resale_price		
1568000		

= prev. col.
= new col.
= example.

Figure 5: Summary of encoding

4.3 Train-Test-Validation Split

With our outliers removed and columns properly encoded, we performed a train-test-validation split across the data with an 80-10-10 ratio. This same split was used across all of our baseline models.

5 Feature Engineering

Models are only as good as the data you provide them with. We sought to improve upon our baseline model performance not just through model architecture, but also through feature engineering and the introduction of more data.

5.1 Macroeconomic Data

HDB resale prices can be incredibly volatile due to factors outside of HDB features. Thus, we decided to enhance our dataset by including auxiliary datasets which represent factors such as macroeconomic changes.

We looked into macroeconomic indicators such as Gross Domestic Product (GDP), Unemployment Rate, Consumer Price Index (CPI) which is measured using Inflation, as well as housing related indicators such as Assets and Liabilities. Initially, the data collated included 56 indicators, with factors such as GDP and CPI having individual subsections into different industries and sectors of spending. (Eg. for GDP, it includes Manufacturing, Food & Beverage and more). Given how inter-related different sectors are to housing prices, we looked at the overall macroeconomic indicators like aggregate GDP to provide a more holistic and high-level view of the current economy. This allowed us to narrow down to 5 crucial macroeconomic indicators which include:

Column	Data Type
GDP In Chained (2015) Dollars	float64
Assets	float64
Liabilities	object
Total Unemployment Rate, (SA)	int64
MAS Core Inflation Measure	float64

Table 2: DataFrame Columns and Data Types for Macroeconomic Indicators

5.2 Data Reduction

To enable faster training and iteration, we decided to downsize our dataset. Training our models on the full dataset was computationally expensive and time-consuming given our large dataset of 381,000+ samples, especially since we planned to experiment with multiple models and training configurations. By narrowing the scope of our data, we were able to prototype, debug, and evaluate models more efficiently.

We also used heatmaps to exclude features depending on their correlation. Ideally, the features used would be correlated with the target variable, resale price, and uncorrelated with each other. After encoding the existing features, we obtained the following graph.

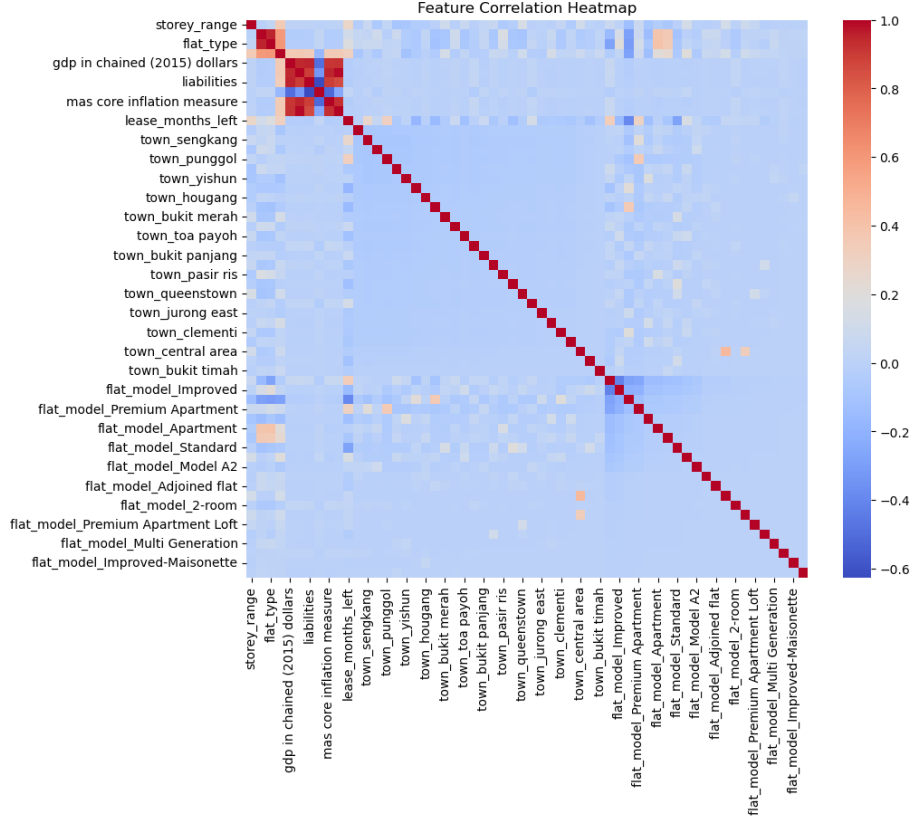


Figure 6: Heatmap for features

Given that there is a high correlation between **floor_area** and **flat_type**, we decided to look at the floor area. The unemployment rate was also highly uncorrelated with other features, including the target feature, so it was excluded from model training. As for the flat model, there are 3 types that show a correlation with resale price. So those encoded columns were kept, while the rest were combined into a new feature called **flat_model_other**. The heatmap also shows a few highly correlated columns in the top left corner. We decided to leave them as tree models and neural networks are not sensitive to multicollinearity. Since some time series models are linear, in the next step we will choose models that would be able to learn the inter-dependencies without being affected by the correlation.

To further reduce the dataset meaningfully without compromising on relevance, we chose to focus on a specific target audience—young home buyers under 40 years old. According to *The Straits Times*, several towns rank in the top 10 both for Gen Z and millennial voter share and for the highest proportion of young families (Mungcal, 2025). Assuming that people of similar

ages would live in the same areas, and since none of the towns is significantly more correlated with resale price than the rest, we decided to use the one hot encoded features for these towns. The towns used are Tampines, Choa Chu Kang, Sengkang, Sembawang and Punggol.

5.3 Datasets

Although we performed this data engineering, we found that the data reduction was harmful to our updated models. Thus, we also explored utilizing a dataset which included macro features but excluded the data reduction.

Table 2 below represents a summary of the different datasets used. The naming conventions of these datasets will be used for the models explained.

Name	Data Reduction	Macroeconomic Features
Dataset A	No	No
Dataset B	Yes	Yes
Dataset C	No	Yes

Table 3: Dataset iterations

6 Baseline Models

With our EDA and data preprocessing complete, we trained 3 baseline models to assess performance of different model types. This was to establish a performance floor on which to assess future models. All 3 baseline models were trained on the Core Dataset (Dataset A), without any feature engineering done just yet.

6.1 Neural Networks

We first aimed to train a basic neural net on the available data. We built a simple multilayer perceptron (MLP) as pictured below.

Layer (type)	Output Shape	Param #
Linear-1	[-1, 128]	6,784
Linear-2	[-1, 64]	8,256
ReLU-3	[-1, 64]	0
Linear-4	[-1, 32]	2,080
ReLU-5	[-1, 32]	0
Linear-6	[-1, 1]	33
Total params: 17,153		
Trainable params: 17,153		
Non-trainable params: 0		

Figure 7: Baseline MLP Architecture

Despite its simplicity, the baseline model captures non-linear interactions in the dataset and outperforms linear regression models. However, its predictive accuracy is limited by its simple architecture, which we will seek to improve with future models. The input layer matches the number of preprocessed features in our dataset, while the output layer produces a single continuous value representing the predicted HDB resale price.

Training was done using mean squared error as a loss function, and with the Adam optimizer. Further training details are depicted in the table below.

Parameter	Setting
Epochs	10
Batch Size	512
Learning Rate	0.001
Loss Function	Mean Squared Error (MSE)
Accuracy Metric	Mean Absolute Error (MAE)

Table 4: Parameter Settings

We found that with the large dataset size, 10 epochs were sufficient to reach convergence as seen in the table below. Through empirical tuning, we identified MSE as the best loss function for this architecture. However, we still used the Mean Absolute Error (MAE) as the primary accuracy metric, given its interpretability and robustness to outliers in price data.

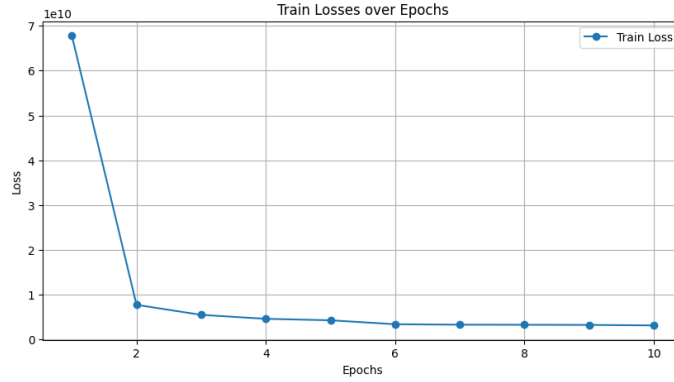


Figure 8: Train losses over epochs for baseline Neural Network

With the model trained, we achieved an MAE of 39,673.28, representing a model which is on average 40,000 SGD off the actual value of the resale price.

6.2 Tree-based Models

We explored 4 types of tree-based models. This includes: decision trees, random forest, XGBoost as well as Adaboost. We included two gradient-boosting models as it builds trees sequentially, with each tree correcting errors of the previous ones. Furthermore, random forest and decision trees have a higher possibility of overfitting given their higher variance. However, if the data are noisy, random forests might outperform as they create independent trees and include feature randomness.

Decision Tree : A hierarchical, tree-structured model consists of nodes representing feature-based splits, branches representing feature-based splits, branches representing decision rules, and leaf nodes corresponding to predicted outcomes.

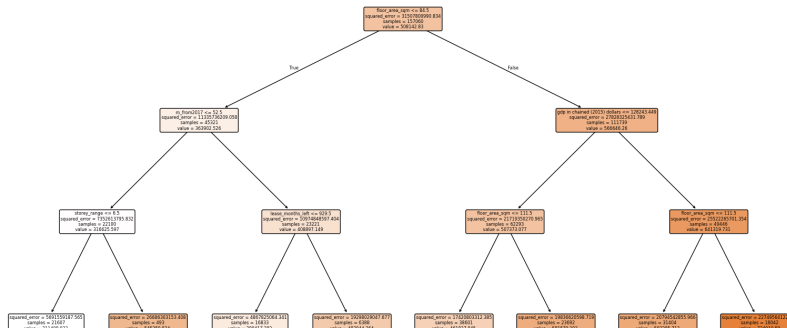


Figure 9: Decision Tree for baseline model

As shown Figure 9 the root node is the feature `floor_area_sqm`. This divides all data samples into two groups based on whether their floor area is less than or equal to 84.5 square meters. Each subsequent node splits the data further using other features, such as:

- m_from2017 (possibly months from 2017)
- GDP in chained (2015) dollars
- lease_months_left
- storey_range
- floor_area_sqm

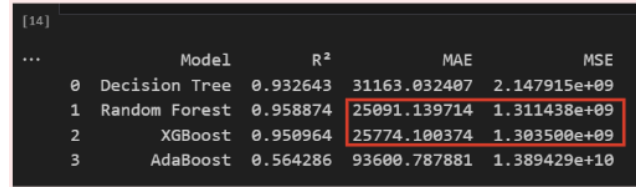
These optimal parameters are found using Random Search hyperparameter tuning. By combining the predictions of many trees, it is able to outperform Decision Tree predictions by preventing overfitting and reducing variance. Thus, this leads to a more stable and reliable performance across datasets A, B and C, providing a model that is more robust to noise and outliers.

Boosting Models : They build trees sequentially, with each new tree trained to correct the errors made by previous trees. This approach differs from Random Forest's parallel, independent tree construction. We experimented with both XGBoost and AdaBoost, with XGBoost out-performing significantly, with an MAE score of 28454.02 as compared to 936000.79. The table below shows the optimal parameters used in XGBoost.

Parameter	Setting
subsample	0.8
n_estimators	300
min_child_weight	5
max_depth	10
learning_rate	0.2
gamma	0.3
colsample_bytree	0.8

Table 6: Parameter Settings for XGBoost

Overall Results : Dataset A was used for the four tree-based models, random forest outperformed the rest, even XGBoost, with an MAE of 25091. (As seen in Figure 10)



...	Model	R ²	MAE	MSE
0	Decision Tree	0.932643	31163.032407	2.147915e+09
1	Random Forest	0.958874	25091.139714	1.311438e+09
2	XGBoost	0.950964	25774.100374	1.303500e+09
3	AdaBoost	0.564286	93600.787881	1.389429e+10

Figure 10: Summary of results of the 4 models

6.3 Time Series Forecasting

We started with a univariate time series model to predict the change in HDB resale prices over time. ARIMA was chosen over SARIMA as the model because there was little to no evidence of seasonality in the resale price over the years after evaluating with an ACF plot.

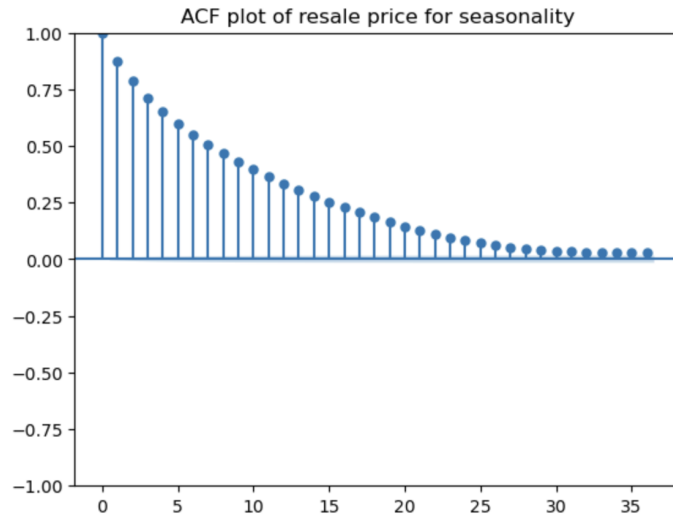


Figure 11: Seasonality Test Results

Since the data set contains multiple rows of data for each day, representing one property listing for that day, the daily average resale price was calculated to fit the input format of the model. To better fit this model, the dataset was split based on the recency of the listing. The training set contained 90% of the oldest data, then the test set contained the newest 10% of the data. The output of this forecasting model has an MAE value of 46575.64.

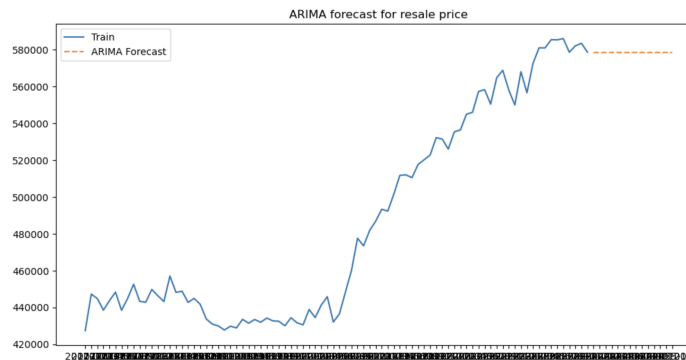


Figure 12: ARIMA forecast graph

7 Updated Models

Taking from our learnings training the baseline models, we sought to train some updated models. We altered not only the datasets to train these models, but also the architectures in an attempt to achieve better results.

7.1 Updated Neural Networks

There were 2 updated neural networks, both of which were trained on the reduced dataset with macroeconomic features (Dataset B). This was due to the original baseline models taking significant amounts of time to train on the original dataset.

Before training the model, we took a bit of time to alter the dataset in hopes of improving model results. Neural networks perform better with values within similar ranges so we used a Standard Scaler to scale the numerical values, including our target variable. We preserved the scalers in the Scalers folder to denormalise the outputs of the model.

7.1.1 Multilayer Perceptron (MLP)

The first model we trained was a natural iteration of the baseline model. The model consists of four hidden layers with decreasing layer sizes ($512 \rightarrow 256 \rightarrow 128 \rightarrow 64$), followed by a final linear output layer to predict the resale price.

We used Leaky ReLU as the activation function to provide non-linearity while mitigating the dying neuron problem. Batch normalization was applied after each dense layer to stabilize and accelerate training, while dropout was added after the middle layers to reduce overfitting. Weight initialization was done using Kaiming Normal Initialization, which is known to be well-suited for networks using ReLU-like activations.

With a smaller dataset, we could afford to run our model for more epochs and a lower training rate. We did this in the hopes that the model would be able to better fit the data.

Parameter	Setting
Epochs	130
Batch Size	512
Learning Rate	0.0005
Loss Function	Mean Absolute Error (MAE)
Accuracy Metric	Mean Absolute Error (MAE)

[H]

Table 7: Parameter Settings

With this in mind, we trained the model. We saw both the train and validation losses decrease over the epochs but they began to both plateau at about 70 epochs.

Layer (type)	Output Shape	Param #
Linear-1	[-1, 512]	8,704
BatchNorm1d-2	[-1, 512]	1,024
LeakyReLU-3	[-1, 512]	0
Linear-4	[-1, 256]	131,328
BatchNorm1d-5	[-1, 256]	512
LeakyReLU-6	[-1, 256]	0
Dropout-7	[-1, 256]	0
Linear-8	[-1, 128]	32,896
BatchNorm1d-9	[-1, 128]	256
LeakyReLU-10	[-1, 128]	0
Dropout-11	[-1, 128]	0
Linear-12	[-1, 64]	8,256
BatchNorm1d-13	[-1, 64]	128
LeakyReLU-14	[-1, 64]	0
Linear-15	[-1, 1]	65
Total params: 183,169		
Trainable params: 183,169		
Non-trainable params: 0		

Figure 13: Multilayer Perceptron Architecture

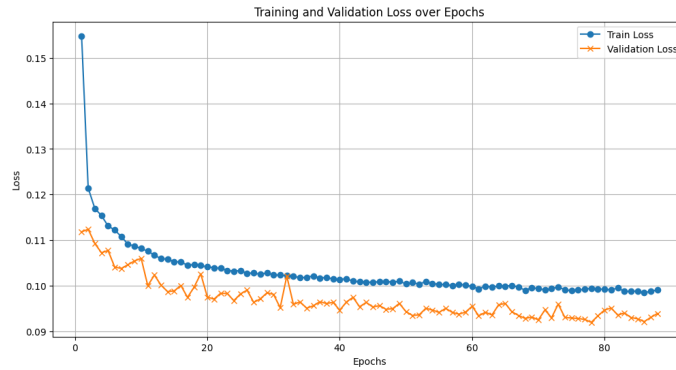


Figure 14: Train and Val Loss over epochs for MLP

Unfortunately, after denormalising, we were left with an unsatisfactory MAE of 52,580.13, which was significantly worse than our baseline model. We were not exactly sure what the issue was at the time so we continued to pursue a different neural network architecture.

7.1.2 Tabular Resnet

We implemented a residual neural network architecture inspired by ResNet, but altered for tabular data regression tasks. The model consists of three fully connected layers with intermediate batch normalization and ReLU activations,

along with dropout for regularization.

The key addition to this model is the use of a residual connection between the first and second dense layers. The skip connection helps the model retain important pricing-related signals from earlier layers, making it easier to learn meaningful patterns in resale price data without losing information as the network deepens. The final layer is once again a single-node output for continuous resale price prediction.

```
def forward(self, x):
    # Input Layer
    x = self.act(self.bn1(self.fc1(x)))

    # Residual Block
    residual = x
    x = self.act(self.bn2(self.fc2(x)))
    x = self.dropout(x)
    x += residual # Skip connection

    # Final Layers
    x = self.act(self.bn3(self.fc3(x)))
    x = self.dropout(x)

    out = self.fc_out(x)
    return out
```

Figure 15: Tabular Resnet Architecture

We once again ran the model for an increased number of epochs and a reduced learning rate. However, this time we attempted to use SmoothL1Loss (Huber Loss) as a compromise between MSE and MAE, aiming to reduce the sensitivity to outliers while still penalizing larger errors more than L1 loss.

Parameter	Setting	
Epochs	100	
Batch Size	512	
Learning Rate	0.0005	[H]
Loss Function	Mean Absolute Error (MAE)	
Accuracy Metric	Mean Absolute Error (MAE)	

Table 8: Parameter Settings

Both training and validation loss exhibit a steady downward trend, indicating successful learning without overfitting. Furthermore, the learning curve was smoother than that of the Multilayer Perceptron (presumably due to the SmoothL1Loss and skip connection).

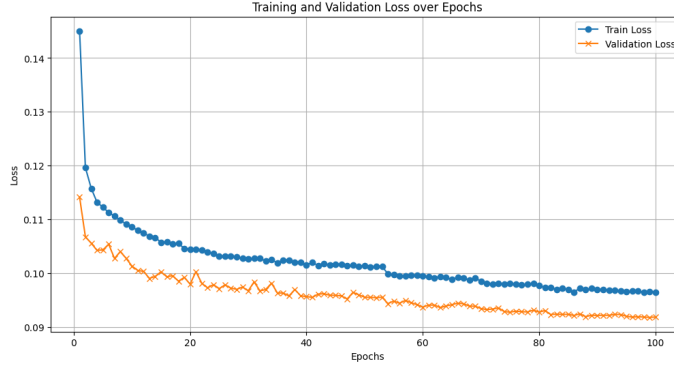


Figure 16: Train and Val Loss over epochs for Tabular Resnet

Unfortunately, once again, after denormalising, we were left with an unsatisfactory MAE of 52,122.70, only marginally better than the multilayer perceptron. We believe that this was due to a less detailed dataset, implying that our data reduction, although good at reducing our training times, appeared to have not preserved data patterns crucial to generating an accurate prediction. We did not have the time or compute to train a neural net on fuller datasets, so we moved on to other models using Dataset C.

7.2 Updated Multivariate Time Series

Despite the lower MAE value relative to the other models, we decided that ARIMA was not sufficient since it does not consider the other features of the dataset that may affect the resale price. To address this issue, we decided to explore the use of the VAR model. This is because it is able to take in multiple features to predict one target variable. Similar to the ARIMA model, the input for this model allowed one row for each datetime value. As such, the data were aggregated by date to find the monthly average for all of the features. After finding the best lag, the fitted model had an MAE value of 34732.49.

To further improve the accuracy of the VAR model, we decided to use XGBoost. This would not only reduce the MAE, but also reduce the effect of multicollinearity between the correlated features. Hyperparameter tuning was also conducted to further optimise the performance of the model. As a result, the MAE value decreased to 30548.01 and the following forecast was obtained.

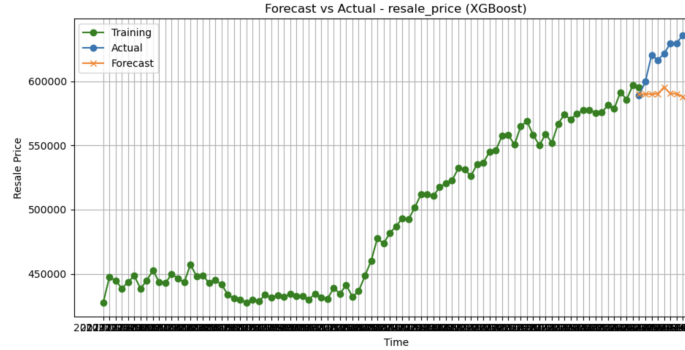


Figure 17: VAR Forecast

We also looked into optimising this model by going back to the data cleaning step and changing the input dataset. Previously, some of the features were removed but we chose to confirm if there were any underlying patterns between them and the target variable. However, even with XGBoost, the MAE was slightly higher at 31832.40. As a result, this confirmed that the low correlation between the removed features and the target variable contributed more to increase the model’s parameter estimation uncertainty rather than identifying more trends in the dataset.

7.3 Updated Tree-based models

We worked on further refining the architecture of XGBoost and Random Forest as it had significantly better MAE scores as compared to the rest. Based on the MAE scores on each dataset, the final model was trained on Dataset C, with the optimal hyperparameters found using Random Search. Feature reduction (Dataset B) resulted in significant increase in the MAE values of XGBoost. This may be due to certain complex feature interactions that XGBoost is able to capture on a larger dataset. These underlying patterns could be useful in preventing overfitting. Furthermore, XGBoost automatically ignores irrelevant or unnecessary features, hence keeping the original dataset serves the model better in preventing over-generalization of the dataset.

8 Results

As shown in Table 9, the performance of XGBoost improved to exceed Random Forest when Dataset 3 is used. This might be because of XGBoost’s ability to automatically perform feature selection. This means that macroeconomic indicators significantly affect resale prices, allowing the model prevent overfitting and able to handle high-dimensional datasets.

Dataset Description	Model	Performance
Dataset A	Neural Network	39673.28
	Random Forest	25091.14
	Time Series	46575.64
	XGBoost	28454.02
Dataset B	Multilayer Perceptron	52580.13
	Tabular ResNet	52122.70
	XGBoost	46833.09
Dataset C	XGBoost	23863.45
	Random Forest	25547.63
	Time Series	31832.40
	Decision Tree	30207.62

Table 9: Model performance on various datasets

9 Conclusion

In this project, we tried a range of models to predict HDB resale prices and found that neural networks were kind of a pain. While they’re powerful in theory, especially for high-dimensional or unstructured data like images or text, they really didn’t shine here. Training was slow and further data engineering was required. Even after trying more advanced setups like residual connections and different loss functions, the improvements were minimal.

Tree-based models like XGBoost, on the other hand, were far easier to work with. They trained faster, required less preprocessing, and performed better out of the box. We also learned that reducing dataset size helped with speed, but probably hurt performance by stripping away important patterns.

In the end, this project reminded us that simpler models often work better for structured data, and that good results rely just as much on data choices as on model architecture.

References

- [1] Housing Development Board. (2025, February 15). *Resale flat prices based on registration date from Jan-2017 onwards — HDB*. data.gov.sg. Retrieved February 15, 2025, from https://data.gov.sg/datasets/d_8b84c4ee58e3cfc0ece0d773c8ca6abc/view
- [2] Sharma, V. (2023, March 7). Unlocking the power of Mahalanobis distance: Exploring multivariate data analysis with Python. *Medium*. Retrieved from https://medium.com/@the_daft_introvert/mahalanobis-distance-5c11a757b099
- [3] Mungcal, A., Ang, Q., Raguraman, A., Hamzah, A., Marin, C., Yusof, Z. M., Wong, P. T., Chelvan, V. P., Ng, T., Tay, H. Y., Shafeeq, S., Tan,

S.-A., Adeline, S., Ang, S., Devaraj, S., Rajan, S. T., Sverdau, R., Kurohi, R., Chia, O., ... Chin, S. F. (2025, April 7). GE2025: All you need to know about your constituency. *The Straits Times*. Retrieved April 7, 2025, from https://www.straitstimes.com/multimedia/graphics/2025/04/ge2025-constituencies/index.html?utm_medium=social&utm_source=telegram&utm_campaign=sttg#ang_mo_kio