AirBnB Listing Price Recommendation with Image-Based Machine Learning/Deep Learning Methods

Ivy Delos Santos (A0206611U) E0427439@U.NUS.EDU
Wang Rui (A0206477Y) WANG.RUI@U.NUS.EDU
Metha Agustina Santosa (A0206455H) E0427283@U.NUS.EDU
Yu Yue Robbie (A0206534L) E0427362@U.NUS.EDU
Yang Liu (A0206474E) E0427302@U.NUS.EDU
Code Link: https://github.com/yangliu0239/BT5153_Group_Project

Abstract

This paper employs multiple regression, neural networks and ensemble methods to train models to predict listing prices based on Singapore AirBnB data in 2019. The models involve various features extracted from AirBnB website public data coupled with engineered features including image features so as to generate a model which provides reasonable pricing for a given listing with data proof. The model can be used to advise property owners who are deciding what price to set for their listings. Through exploratory data analysis and feature ranking from models, we also provided business insights about how to strategically set up the listing features and enhance the main picture quality in order to increase possible price offer for a property.

1. Project Motivations and Problem Statement

International tourism contributes around 4% to Singapore's national GDP and is projected to rise to 4.4% in 2028. Accommodation contributes substantially to this, with 20% being reported in Singapore Tourism Board's latest available Tourism Sector Performance Report (Q2 2019).

Airbnb is one of the accommodation options for international visitors. It is a very popular tool for people sharing a spare room, apartment or house as well as a viral app for residential renting among strangers. Since it was founded in 2008, the business continues to grow rapidly and becomes a serious threat for traditional hospitality business. According to the Straits Times, there was an estimated number of 7,000 Singapore property listings on Airbnb as of November 2016, and average hosts earn around 5,000 SGD a year.

As a residential owner, one needs to know the market in order to bid for the most viable price for the house and therefore maximize profit. Airbnb does provide hosts with general guidance. However, there are no straightforward methods to determine the best price. One method may be to set your price based on the calculated average of the listed prices for listings that are similar to yours. However, one would need to update the price frequently as the market is dynamic. In addition, this method doesn't take into consideration other factors that may boost your listing's attractiveness (e.g. nearby amenities).

This project aims to generate a report that enables price prediction for Singapore AirBnB room owners based on region with feature ranking. Additionally, this project also sets to inculcate an image-based machine learning model that predicts the price of a property based on the main image in the website's listing, which might help the hosts to determine which picture should be used as the first picture for a listing to maximize the price to be charged for revenue maximization.

2. Hypothesis and Objectives

Based on intuitive thinking, we hypothesized that the property type, number of beds, property's geography location, amenities, neighbourhood (access to restaurants, supermarket, shops, etc.) are important factors which determine the Airbnb room price. Other features (such as minimum nights of stay) were deemed to not directly affect the room price, and therefore were dropped from feature engineering. In addition, we also explored an image-based machine learning model and predicted the Airbnb price based on the first image in the property's listing combined with other features.

3. Dataset Introduction

3.1 Data Source

There were 3 major data sources from which team took the data: insideairbnb.com, airbnb.com.sg and STB.gov.sg.

3.1.1 Main Data Source

insideairbnb.com, which is independent of AirBnB, scrapes from published information on AirBnB website monthly and stores it in a csv file every month, separately. Data is verified, cleansed, analyzed and aggregated.

3.1.2 Image Data

Scraped AirBnB price information from insideairbnb.com was analytically linked to pictorial data shown on AirBnB official website at the point of listing.

3.1.2 Other Imported Data

STB.gov.sg, a data source from the Singapore government provides the number of arrivals from individual nationalities by month. Data is available for each complete month and quarter.

3.2 Data Description

Insideairbnb.com stores scraped AirBnB data in a file of individual months. Given the features are the same, all the monthly data were compiled into one file in order to observe a general trend. The data is available from March 2019 to Nov 2019. In this project, our aim was to predict the price of properties, whose existing data is available in the dataset.

There are approximately 8000 listings per month (71949 in total in 9 months) and 106 features recorded for each listing with 3.4% missing data. Missing data is mainly from thumbnail_url (64.6%), medium_url (62.6%), xl_picture_url (52.8%), notes (41.8%) and neighborhood_overview (40.4%), which are not related to our major model explanation, so missing data won't be influencing the model and conclusion significantly.

Singapore Tourism Board dataset includes detailed breakdown of the number of visitors from individual countries, regions, and continents by month.

Table 1. Features highly likely related modelling.

Features	Meaning	Type*
id	Identifying the property	C
scrape_id/month	Timestamp of data scraping	N
picture_url	First picture of the listing	P
host_is_superhost	host category	C
neighbourhood	District name of the listing	C
neighbourhood_group_cleansed	Generalized district name of the listing	C
latitude	Latitude of the listing	N
longitude	Longitude of the listing	N
property_type	Type of the listing	C
bed_type	Type of beds offered	C
room_type	Type of room offered	C
bathrooms/bedrooms/beds	Number of bathrooms/bedrooms/beds provided	N
amenities	Facilities and services provided by the host	C
price	Daily price	N
security_deposit	Minimal deposit amount for booking	N
cleaning_fee	Housekeeping fees	N
guests_included	Maximal number of guests	N
minimum_nights	Least nights the guests needing to stay	N
maximum_nights	Maximal nights the guests staying for	N
availability_30	Available days the next corresponding days	N
review_scores_rating	Review scores by 100	N
number_of_reviews	Total number of reviews	N
reviews_per_month	Ave. number of reviews	N
Number of Arrivals	Number of visitors	N
cancellation_policy	Cancellation policy (flexible/moderate/strict)	C

* C: Categorical; N: Numerical; P: Pictorial

AirBnB listing dataset has been explored in terms of price and number of such properties in various areas of Singapore, shown in Figure 1.

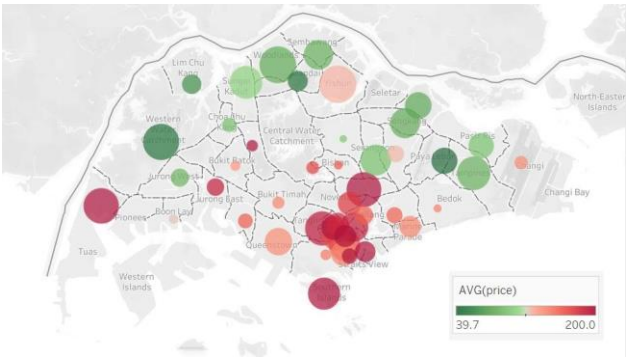


Figure 1. Heatmap of listing price. The size of the circle represents the number of properties in the neighborhood.

4. Data Preparation and Pre-Processing Flow

4.1. Feature Selection

We selected the features with respect to the daily price for individual occupancy as the dependent or target feature.

4.1.1 InsideAirBnB Data Feature Selection

There are 106 features provided in the dataset. Based on our domain knowledge, we reduced the number to 31. This was further reduced by dropping features that were later deemed irrelevant such as latitude, longitude and last_scraped. Features that seemed to be already represented by other features were also dropped, such as neighbourhood_cleansed which is better represented by neighbourhood_group_cleansed (which only had 5 categories). Review_scores_rating was also selected as it

has one of the highest correlations with price and represents the overall rating.

4.1.2 STB DATA FEATURE SELECTION

Total number of arrivals for different months from different destination has been download from Singapore Tourism Board (STB.gov.sg) and used as one of features.

4.1.3 AIRBNB DATA FEATURE SELECTION

The picture or photo of the respective property from AirBnb.com was used as a feature.

4.2. Data Pre-Processing

Some data pre-processing strategies used were as follows:

4.2.1 DATA CLEANING

Removed dollar sign ($) from price-related fields and converted to float.

4.2.2 HANDLING MISSING DATA & OURLIERS

Below were the strategies used to handle missing data:

review_scores_rating: 38.24% values were missing. However, with no intuitive way to impute ratings and with sufficient data remaining (44437 rows), we decided to drop missing ratings.

host_is_superhost: As the hosts can only be either superhost or just a normal host, missing values were taken to be non-superhost and set to "false".

bathrooms, bedrooms, beds: As these values cannot be zero in a rental house, missing values were replaced with the median. Outliers were also identified based on 3sigma (99.7%) and replaced with median as well.

security_deposit, cleaning_fee: Missing values were replaced with "0" as we suspect that null in these entries simply means that there are no additional charges for security deposit and cleaning fee. Outliers were also identified based on 3sigma (99.7%) and replaced with median as well.

reviews_per_month: Treated as no review & replaced with 0.

4.2.3 HANDLING CATEGORICAL FEATURES

As machine learning algorithms cannot work with categorical data directly, one hot encoding was performed to convert categorical data (*host_is_superhost, neighbourhood_group_cleansed, property_type, room_type, bed_type, cancellation_policy, month*) to integers before analysis.

4.2.4 LOG TRANSFORMATION FOR PRICE

Entries with *price* = 0 or *price* > 600 (>99%: outliers) were removed. Log transformation was then done to normalize the right-skewed price distribution.

4.2.5 NUMERICAL DATA NORMALIZATION

For neural network, prior to splitting the data into train & test sets, it is important to make sure that the scale of the input features are similar in order to make it easier for the initialization of the neural network. In this project, we used existing packages from scikit-learn (StandardScaler and MinMaxScaler).

4.2.6 IMAGE DATA PROCESSING

For image processing, we used the Image Module from Python Imaging Library to convert images retrieved from the website URLs and to Numpy array. Then the input data was divided by 255 (max of RBG data) to scale the data in the range of 0 to 1.

4.3. Feature Engineering

In order to get a good quality dataset and improve the performance of machine learning model, some of the feature engineering methods implemented are as follows:

4.3.1 FEATURE EXTRACTION

Below were the features constructed:

month: Extracted from scrape_id (and converted to date type)

property_type: Ranked the top 5 in terms of frequency (Apartment, House, Hostel, Condominium, Serviced apartment), the rest were labeled as *Others*

num_amenities, amenities_length: Converted *amenities* from string to list (entries "translation missing: en.hosting_amenity_49" & "translation missing: en.hosting_amenity_50" removed); the former represents the number of items in the list while the latter represents the average word length in the list.

tot_arrival: Total number of arrivals for given month (excluding Chinese nationals as Airbnb is illegal in China) from Visitors Dataset.

Apart from the numerical/categorical features, image feature extraction was done by the convolutional neural network and deep learning models like InceptionV3.

4.4 Exploratory Data Analysis

4.4.1 OBSERVATIONS FOR CATEGORICAL DATA

Below were the observations for categorical features prior to one-hot encoding (as shown in Figure A3 in Appendix):

Superhosts' listing prices are higher than normal hosts, which is in line with what we expected.

Properties in the central region have highest mean compared to the rest; North has the lowest.

Highest prices can be seen from service apartments, which is expected as well as they can cater for more guests and have more amenities. Lowest prices are from hostels.

Shared rooms have lowest mean prices, followed by private rooms.

Airbeds have lowest prices, followed by futons. Surprisingly, pull-out sofa had higher means compared to real beds. This may be because there were only 47 listings with pull-out sofa as compared to 43,930 with real beds.

Group of listings with Super strict policy of 30days has a much higher mean compared to the rest. Looking into the characteristics of this group (which can be seen in Figure A4 in Appendix), it was found that all of them were in the Central Region and most (99%) were serviced apartments (which as mentioned earlier were both pricier compared to other regions/property types). Finally, all the hosts were superhosts, with review scores at an average of 97.3. This makes sense as this policy is offered only to selected hosts, probably top-notch ones. All these could explain why this category's baseline is much higher than the rest, but we believe that by itself, this category is not really an important feature.

Price is comparable across the different months.

4.4.2 Observations For Numerical Data

Further observations for numerical features have also been found as below (as shown in Figure A5 in Appendix):

More guests can be included in the same unit results in higher price.

Among bedrooms, bathrooms and beds, hosts preferring to set a higher price with more bedrooms and beds but adding more bathrooms does not make a difference on price.

Some of the Airbnb posts apply security deposit and cleaning fees, and both have positive correlation with price. It could because when listing price is higher, which might represent a better place, the host would like to charge higher deposit and cleaning fee to maintain the rooms.

More review and higher review scores contributed to a relatively higher price.

Price of listings increases with the number of amenities.

A listing's price does not appear to be affected by its availability in the next 30 days nor by number of visitor arrivals for that month.

4.4.3 Observations For Text Data

Observation on text feature of "amenities" and "description" based on word cloud package:

Across whole price range, hair dryer, WIFI, air conditioning and iron were found necessary and common. High-pricing listings normally provide friendly workspace and hangers.

Therefore, in the listing description, high pricing listings are normally emphasizing on friendly workspace and hangers to the customer to match its higher price.

5. Methodology

5.1 General Modelling Concept for Price Prediction

In this project, we used different models to handle the different type of inputs. One group of models mainly dealt with numerical/categorical features. The other models dealt with image feature extractions and learnings. We also created a CNN model that concatenated the numerical/categorical features together with the image features. Figure 2 below is an overview of the models used based on the different type of features.

Since we need to observe the output of each step in machine learning process, hence we do not use pipeline.

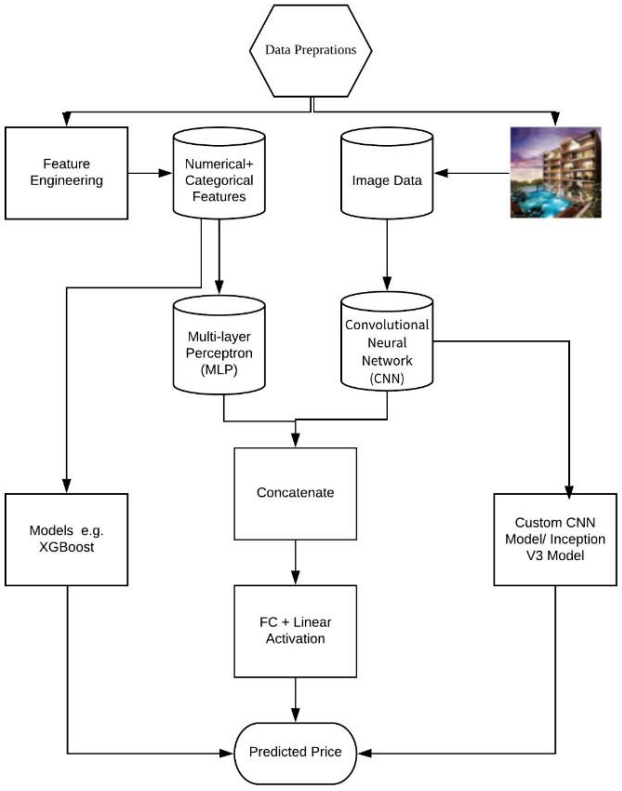


Figure 2. Airbnb Price Prediction Workflow

5.2 Models

5.1.1 Multiple Regression

Multiple regression is the most common and powerful model for supervised numerical data prediction. During the regression process, the important factors are highly weighted while the non-critical factors will be penalized with a small coefficient. It can provide an explainable model to study the factors on listing price while giving a moderately accurate prediction result.

5.1.2 Gradient Boosting Machine

This is a boosting algorithm that identifies shortcomings of a weak learner by gradients. The key idea is to set the target outcomes for the next model in order to minimize the error.

The name Gradient Boosting arises because target outcomes for each case are set based on the gradient of the error with respect to the prediction. Each new model takes a step in the direction that minimizes prediction error, in the space of possible predictions for each training case.

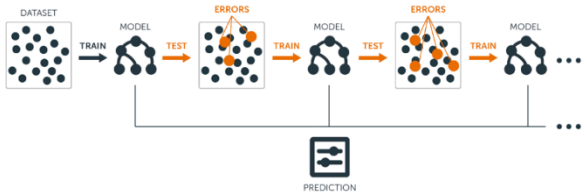


Figure 3. General Illustration of Boosting Method (Boehmke, 2018)

5.1.3 XGBoost

This uses gradient boosting decision tree algorithm. In gradient boosting, new models that predict the residuals or errors of prior models are created, and then added together to make the final prediction. It uses a gradient descent algorithm to minimize the loss when adding new models (Khandelwal, 2017).

The benefits of XGBoost are its execution speed and model performance. With this, we will be able to obtain estimates of feature importance among the numerous features that are available, and at the same time obtain a numerical prediction result.

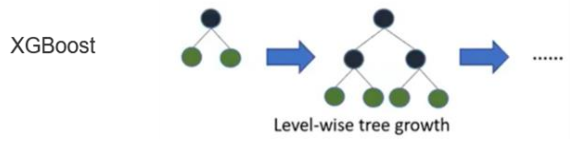


Figure 4. Diagram of Level-wise tree growth

5.1.4 Random Forest

This model uses a technique called bootstrap aggregation, commonly known as bagging. The model is made up of many decision trees and uses 2 key concepts as below (Will, 2018).

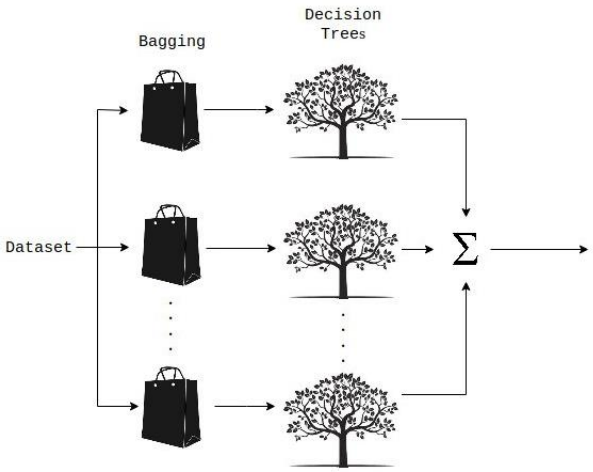


Figure 5. Illustration of Random Forest Regression (Krishni, 2018)

Random sampling of training data points when building trees; and

Random subsets of features considered when splitting nodes.

5.1.5 Neural Networks

Neural networks are multi-layer networks of neurons that are used to perform tasks like classification or numerical regressions. The network is organized in three interconnected layers: input layer, hidden layers that may include more than one layer, and output layer. The training process of a neural network, at a high level, is to define a cost function and use gradient descent optimization to minimize it.

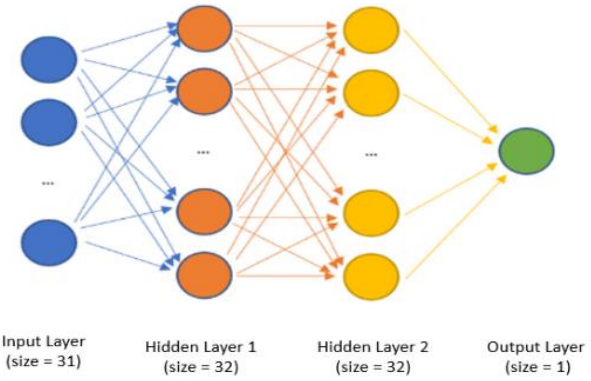


Figure 6. Neural network architecture (Lee, 2019)

A multilayer perceptron (MLP) has three or more layers. It utilizes a nonlinear activation function to classify data that is not linearly separable. Every node in a layer connects to each node in the following layer making the network fully connected.

A convolutional neural network (CNN) contains one or more convolutional layers, pooling or fully connected, and uses a variation of multilayer perceptrons. Convolutional layers apply convolution operation to the input allowing the network to be deeper with much fewer parameters, thus often used for image processing.

Deep neural network requires huge dataset to train. However, in practice, it's not easy to get that large dataset, and it's also computationally expensive to train the network from scratch due to the complexity of the model. Thus, transfer learning is often used where we use pre-trained network weights as initializations and or a fixed feature extractor which could help ease the pain for solving computer vision problems.

Keras provides a set of state-of-the-art deep learning models along with pre-trained weights on ImageNet. These pre-trained models can be used for image classification, feature extraction, and transfer learning. The InceptionV3 model is one of the best models among those.

The InceptionV3 model consists of two parts: feature extraction part with a convolutional neural network and classification part with fully connected and softmax layers. The model extracts general features from input images (which are the first picture in each of the listings) in the first part, and then classifies them based on those features in the second part (Prakash, 2017).

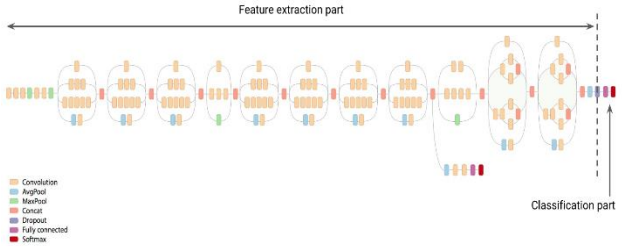


Figure 7. Schematic illustration of image processing model

In this project, we developed a CNN (convolutional neural network) using Keras with a tensorflow backend on our own. And we also explored the InceptionV3 model based on transfer learning concept to leverage on the pretrained weights to improve the feature extraction.

We constructed the CNN models with different layers including Conv2D, pooling, dropout, fully connected layer and with different activation functions, which are mainly ReLU (Rectified Linear Unit), SELU (Scaled Exponential Linear Unit) and linear. Like ReLUs, SELUs has no problem with vanishing gradients and has the advantage of its internal normalization and avoiding dying ReLU issues (Timo, 2018). Linear activation function is used for the output layer to perform the regression for price prediction.

The model structure for the initial CNN model (based on image data alone), MLP and the second CNN model (combining image data with other features) could be found in A7, A8, and in Appendix.

The InceptionV3 models are trained in following 2 steps: Firstly, transfer learning: Construct the model using all the pretrained weights on "ImageNet" for all layers excluding the top layers to do transfer learning and use our own defined top layers for regression.

Secondly, fine tuning: Construct the model freezing earlier layers (weights reused) but unfreezing the last 2 inception blocks for training (from layer 249 onwards in this case). The model summary for last few layers could be found in A10 in Appendix.

5.3 Model Parameter Tuning

To improve the performance of the model, we are using GridSearchCV for parameter tuning.

For deep learning models, parameter tuning was done manually with iterative loop runs, such as layers, dropout, batch size etc.

5.4 Model Accuracy Evaluation

The accuracy of the prediction was evaluated by comparing the predicted value with the actual value in the testing dataset.

Score evaluation techniques, such as MSE, MAE and R2 were used for accuracy checking in numerical / categorical models. They gave an idea of how close the prediction is to the best fit. At the same time, MSE and MAE did a good job of penalizing large errors.

For deep learning models, we have mainly used Adam optimizer with a learning rate of 1e-3, decay of 1e-3 for our custom CNN model, and RMSProp optimizer, SGD optimizer with learning rate of 0.00001 for the InceptionV3 model transfer learning and fine tunings. We used MSE as the loss function and MSE/MAE as validation metrics to show how the model does on the training set, validation set and test set. For validation and test, we used 20% test split. We have also tried out different Epochs ranging from 5 to 200 and plotted the training history graph for training and validation data to observe overfitting issue. A visualization of the model performance was also done by comparing the predicted price vs actual price for listings with their corresponding image (examples can be seen on A14-A17).

6. Results

Table 2 below demonstrates individual model performance after data cleaning, feature engineering and certain amount of hyperparameter tuning. The best model is observed to be Gradient Boost with mean squared error at 0.016 for the log price prediction.

Table 2. Model Results

Model Type	Meaning	MSE
Regression	OLS	0.153
	Step Forward Multiple Regression	0.153
Neural Networks	Neural Network	0.080
Ensemble Models	Random Forest	0.017
	Gradient Boost	0.016
	XGBoost	0.017
Deep Learning	Customized CNN	0.165
	InceptionV3	0.266
	CNN + MLP	0.118

6.1 Regression Model

In multiple regression, OLS (Ordinary Least Squares) regression was first implemented. This model generates a very clear summary to list p-value, standard error, and t-value for each independent feature to understanding the importance features affecting on the target.

Then features with high p -value ($p>0.05$), which means they had lower correlation to the target, have been removed to get the final OLS model to reduce the model complexity.

The second method for feature selection in regression model is 'Step forward variable selection' using multiple regression model. Starting with an empty pool of independent variables and with R2 initialized as 0, features in the dataset were added into the empty pool one by one. After each new feature was added, all features in the pool were used to fit into regression model on training data, in order to make prediction on test data. If test R2 score increased, the feature would be kept; otherwise it would be dropped. This process is repeated until all features have been brought to the pool once for testing.

After doing so, 34 features were finally selected out of the 41. This method allowed us to filter out those non-correlated data with price and get a less complex model.

The two methods used for feature selection gave similar results (as shown in Figure A6 in Appendix), the test accuracy with MSE is 0.153. Among all the models, accuracies for these two methods were relatively lower. This could be due to the regression model is too simple to explain this problem compared to the rest. However, a regression model is always able to give a direct and intuitive explanation for the results, so it is always good to be set as a baseline.

6.3 Neural Networks

Keras Sequential model from Keras was imported. We then defined our neural network's architecture, with input layer size equal to the number of features (31), followed by 2 hidden layers of 32 neurons each (with ReLU as activation), and output layer of 1 neuron. We used 'adam' as optimizer and 'mean_squared_error' as loss with model.compile. We then trained our model with the training data with model.fit, and found that batch_size = 10, epochs = 50 gave the best metrics (R2 of 0.84, MAE of 0.2092 and MSE of 0.0795).

6.4 Ensemble

Here is how we ran the ensemble models to get a good prediction accuracy.

6.4.1 Without Parameter Tuning

First, we simply ran the models without any parameter tuning. We observed that Random Forest has the highest prediction accuracy among other models.

	Model Score	R Squared	Mean Absolute Error	Mean Squared Error
RandomForestRegressor	0.996067	0.973274	0.054730	0.013517
GradientBoostingRegressor	0.801696	0.806911	0.231423	0.097653
XBoostRegressor	0.943574	0.927089	0.134297	0.036874

Figure 8. Model performance without parameter tuning.

6.4.2 With Parameter Tuning

We ran hyperparameter tuning for Gradient Boost and XGBoost. We excluded Random Forest from this step, since the standard parameter already has a good output (99% of accuracy).

We then performed stacking and used the output as the feature to run the models. Following this, we ran the models with the best parameters obtained from hyperparameter tuning. We observed that prediction accuracy of both Gradient Boost and XGBoost improved.

	Model Score	R Squared	Mean Absolute Error	Mean Squared Error
RandomForestRegressor	0.994564	0.966616	0.071640	0.016884
GradientBoostingRegressor	0.980879	0.968062	0.067784	0.016152
XGBoostRegressor	0.990106	0.965500	0.072418	0.017448

Figure 9. Model performance with parameter tuning.

6.4.3 Feature Ranking

After optimizing our models, we then determined the most important features that could affect price. The following features – *bedrooms*, *room_type_Shared_room*, and *room_type_Private_room* were consistently in the top 5 among all three ensemble models (as shown in Figure A17 in Appendix).

6.5 Image CNN

InceptionV3 was first employed to fit image data with prices. InceptionV3 utilizes pre-trained neural network layers downloaded from ImageNet to conduct fitting with 30 epochs and reach fitting optimization range in 5 epochs. Its result is no better than any ensemble method, ending with over 0.26 in mean squared error (MSE).

Customized CNN for image-only data is trained with 30 epochs. It was observed to have a better result in MSE at 0.165.

A further MLP consisting of all essential listing features mentioned in Table 1 was introduced to the neural networks and the model was re-trained with higher epochs considering both image data and the aforementioned essential features. The MSE of the model was further improved to 0.118, despite still being worse off than XGBoost.

Current assessment is based on the first picture of each listing with 1000 image samples due to RAM constraints in Google Colab. The model might be trained to be more accurate with more images taken from the listings. This also suggests that the essential features of the listing, such as location, are more important than image data.

6.6 Comparisons and Recommendations

Comparing different models, we observed ensemble methods give the best predictions among all. They have advantages of less time spent for tuning and training model, feature importance ranking as well as their unique built-in characteristics for better fitting, hyperparameter tuning and regularization.

Convolutional neural networks are potential good prediction models as well, customized CNN can enable image data to be combined in training with other data features. InceptionV3 has built-in image feature extractions from Pre-trained ImageNet, which makes prediction more accurate than conventional CNN.

However, for neural networks, developing machine learning systems capable of handling mixed data was extremely challenging as each data type may require separate pre-processing steps, including scaling, normalization, and feature engineering. In addition, training a complex neural network was computationally heavy, not to mention the parameter tunings to achieve the best performance. In this project, we used the GPU provided by Google Colab, however, the session had unstable RAM allocation and limited the image samples that could be downloaded for analysis, which potentially caused overfitting and suboptimal performance for the deep learning models.

7. Conclusion

7.1 Modelling Application

The fine-tuned Gradient Boost model performed the best among all the models and it also provided feature rankings which determines the price of the Airbnb listing. This provides a good reference of the price range for the property owners to consider putting up in the Airbnb listing price. Moreover, the model could be applied in other applications related to predictions, not only in housing price prediction.

7.2 Business Insight

From data exploration, we observe several noticeable tips that can be recommended for property owners to set their price or opportunity to beef up their pricing.

It is found that normally brighter pictures, room images with window views have higher pricing. The owners should allow as many guests as possible, even if the number of accommodators is fixed. To increase pricing potential, owners can make their room more work-friendly and provide hangers. Owners' pricing isn't strongly affected by overall tourism volume given normal variation. Owners don't necessarily need to invest in additional beds, because number of bedrooms are more important than the beds needed. If the owner insists to increase the number of beds in the room, real beds are better than pull-out sofa.

References

Boehmke, B. (2018). *UC Business Analytics R Programming Guide*. Retrieved from University of Cincinnati website: http://uc-r.github.io/gbm_regression.

Böhm, T. (2018). *A first Introduction to SELUs and why you should start using them as your Activation Functions* Retrieved from Towards Data Science website: https://towardsdatascience.com/gentle-introduction-to-selus-b19943068cd9.

Brownlee, J. (2016). *Regression Tutorial with the Keras Deep Learning Library in Python*. Retrieved from Machine Learning Mastery website: https://machinelearningmastery.com/regression-tutorial-keras-deep-learning-library-python/.

Brownlee, J. (2019). *How to use Data Scaling Improve Deep Learning Model Stability and Performance* . Retrieved from Machine Learning Mastery website*:* https://machinelearningmastery.com/how-to-improve-neural-network-stability-and-modeling-performance-with-data-scaling/.

Jay, P. (2017). *Transfer Learning using Keras.* Retrieved from Medium website: https://medium.com/@14prakash/transfer-learning-using-keras-d804b2e04ef8.

Jeo, Y. (2016). *Average Singapore Airbnb host makes about $5,000 a year*. Retrieved from The Singapore Times website: https://www.straitstimes.com/singapore/housing/average-singapore-airbnb-host-makes-about-5000-a-year.

Khandelwal, P. (2017). *Which algorithm takes the crown: Light GBM vs XGBOOST?* Retrieved from Analytics Vidhya website: https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/.

Koehrsen, W. (2018). *An Implementation and Explanation of the Random Forest in Python*. Retrieved from Medium website: https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76.

Krishni, Nov 27, 2018, *A Beginners Guide to Random Forest Regression*. Retrieved from A Medium Corporation website: https://medium.com/datadriveninvestor/random-forest-regression-9871bc9a25eb.

Lee, J. (2019). *Build your first Neural Network to predict house prices with Keras*. Retrieved from A Medium Corporation website: https://medium.com/intuitive-deep-learning/build-your-first-neural-network-to-predict-house-prices-with-keras-eb5db60232c.

Ponti, M., Ribeiro, L. & Nazare, T., Bui, T. & Collomosse, J. (2017). *Everything You Wanted to Know about Deep Learning for Computer Vision but Were Afraid to Ask*. 17-41. 10.1109/SIBGRAPI-T.2017.12.

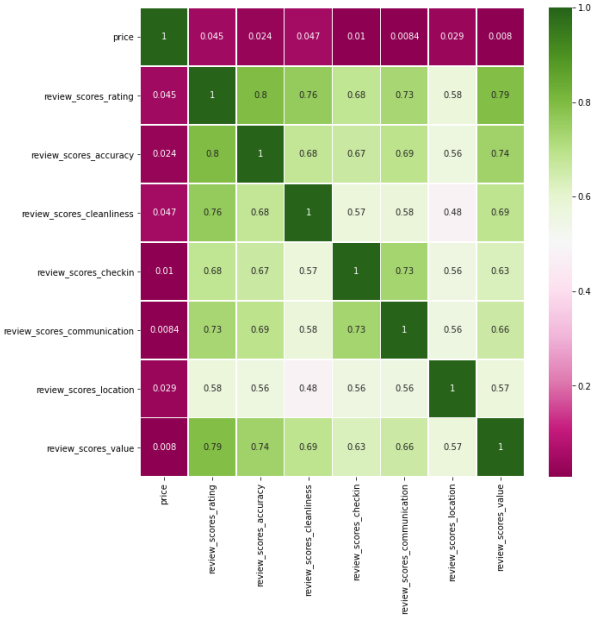
Raj, B. (2018). *A Simple Guide to the Versions of the Inception Network*. Retrieved from A Medium Corporation website: https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202.

Rosebrock, A. (2019). *Keras: Multiple Inputs and Mixed Data*. Retrieved from pyimagesearch website: https://www.pyimagesearch.com/2019/02/04/keras-multiple-inputs-and-mixed-data.

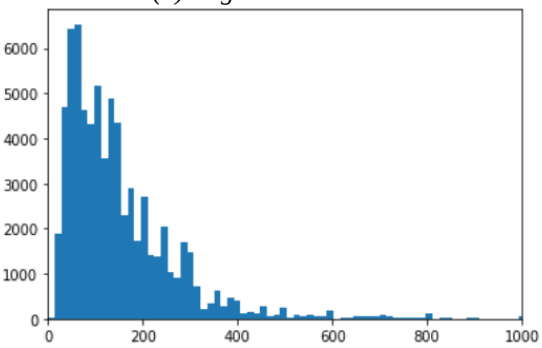
Singapore Tourism Board Overview. (2019). *STB Overview*. Retrieved from STB website: https://www.stb.gov.sg/content/stb/en/about-stb/overview.html.

Appendices

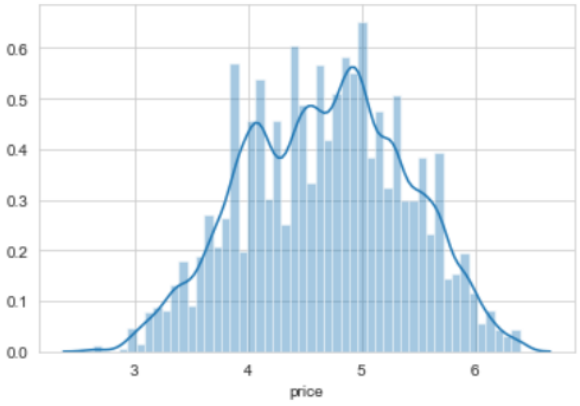
A1.Correlation heatmap of price and review scores



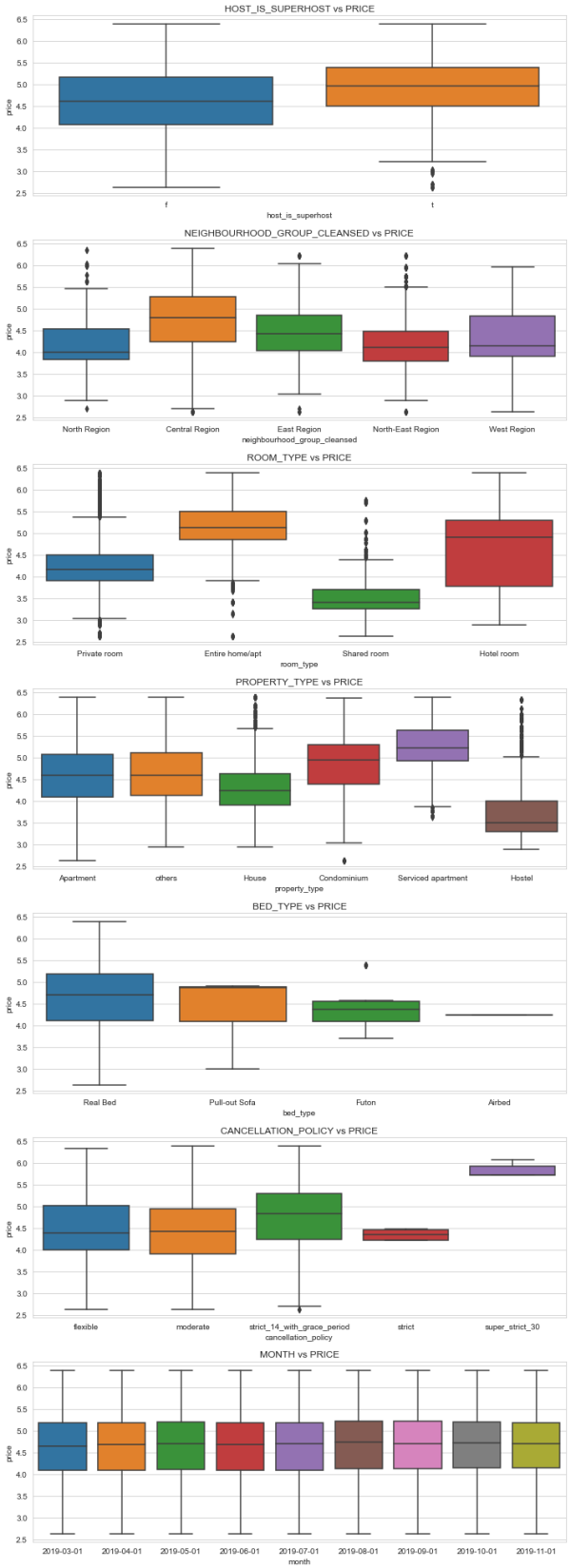
A2 (a)Original Price Distribution



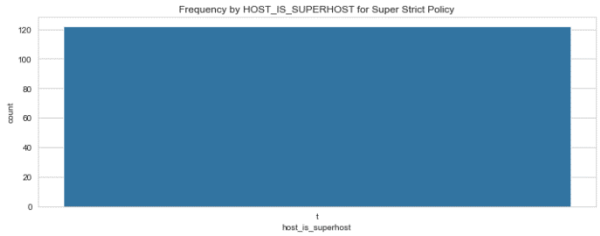
A2 (b)Price Distribution after Transformation

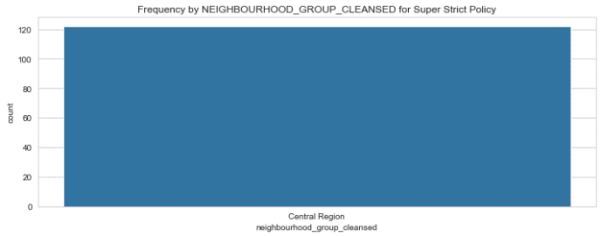


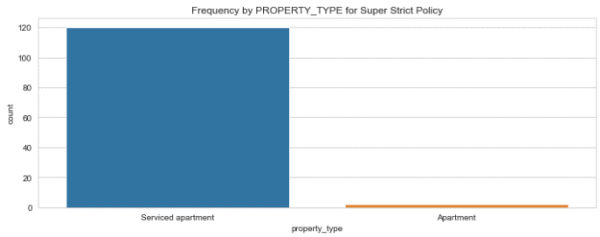
A3. EDA - Boxplots of Price by Categorical Features



A4. Characteristics of Group Cancelation Policy "Super Strict Policy 30days"





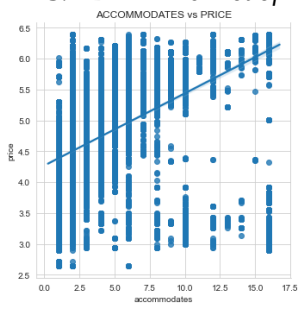


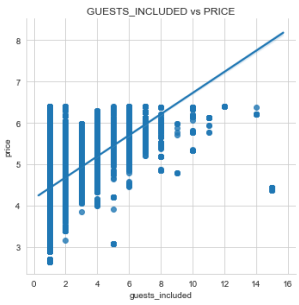

```
In [118]: df_superstrict30.describe()
          #Mean ratings are quite high (97)
```

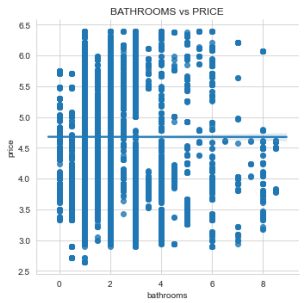
Out[118]:

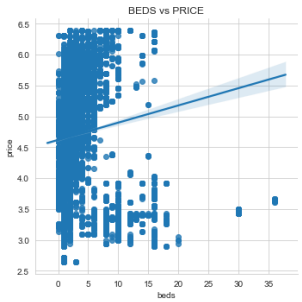
	accommodates	bathrooms	bedrooms	beds	price	security_deposit	cleaning_fee	guests_included	availability_30	review_scores_rating
count	122.000000	122.0	122.000000	122.0	122.000000	122.0	122.0	122.000000	122.000000	122.000000
mean	2.262295	1.0	0.459016	1.0	5.823816	0.0	0.0	1.336066	21.713115	97.319672
std	0.586392	0.0	0.500372	0.0	0.120630	0.0	0.0	0.474310	9.446220	6.119021
min	1.000000	1.0	0.000000	1.0	5.713733	0.0	0.0	1.000000	0.000000	40.000000
25%	2.000000	1.0	0.000000	1.0	5.713733	0.0	0.0	1.000000	19.000000	97.000000
50%	2.000000	1.0	0.000000	1.0	5.717028	0.0	0.0	1.000000	24.000000	99.000000
75%	3.000000	1.0	1.000000	1.0	5.924256	0.0	0.0	2.000000	29.000000	100.000000
max	3.000000	1.0	1.000000	1.0	6.077642	0.0	0.0	2.000000	30.000000	100.000000

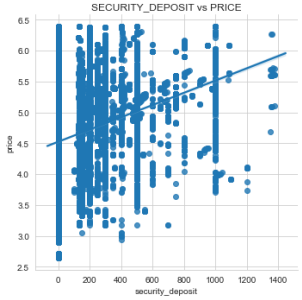
A5. EDA – Line Plot of Numerical Features vs Price

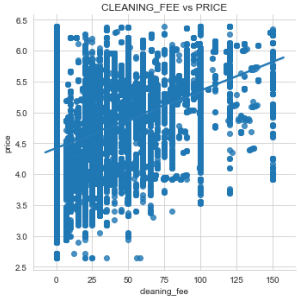


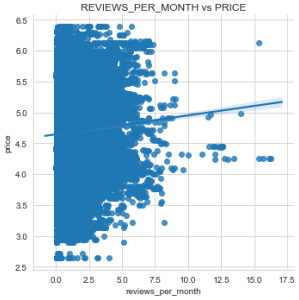


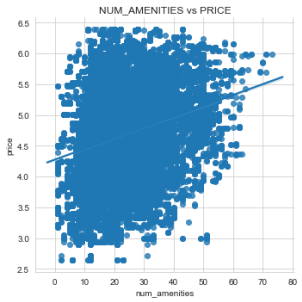


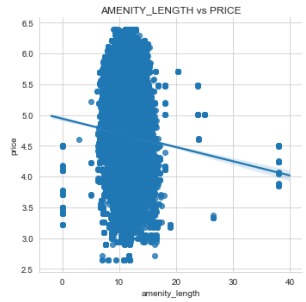


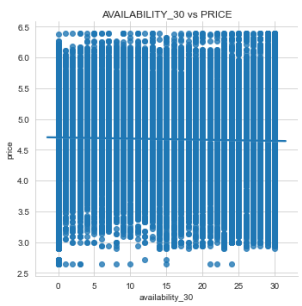


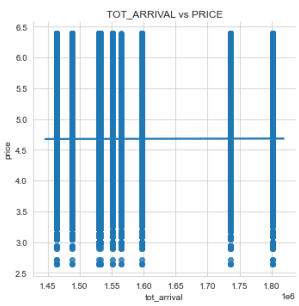












A6. All model results

Model Types	Model Name	R2	MAE	MSE
Regression Model	OLS	0.70	0.29	0.15
	Step Forward Multiple Regression	0.70	0.29	0.15
Neural Network	Neural Network	0.84	0.21	0.080
Ensemble Model	Random Forest	0.97	0.07	0.017
	Gradient Boost	0.97	0.07	0.016
	XGBoost	0.97	0.07	0.017
Deep Learning	Custom CNN (Image alone)		0.30	0.17
	InceptionV3 (Image alone)		0.31	0.18
	Custom CNN (Image + other features)		0.25	0.12

A7. Model Summary for initial CNN model (image only)

Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 112, 112, 32)	896
max_pooling2d_2 (MaxPooling2	(None, 55, 55, 32)	0
conv2d_6 (Conv2D)	(None, 55, 55, 64)	18496
average_pooling2d_2 (Average	(None, 53, 53, 64)	0
conv2d_7 (Conv2D)	(None, 27, 27, 128)	73856
dropout_4 (Dropout)	(None, 27, 27, 128)	0
conv2d_8 (Conv2D)	(None, 27, 27, 64)	73792
flatten_2 (Flatten)	(None, 46656)	0
dense_5 (Dense)	(None, 128)	5972096
dropout_5 (Dropout)	(None, 128)	0
dense_6 (Dense)	(None, 64)	8256
dropout_6 (Dropout)	(None, 64)	0
dense_7 (Dense)	(None, 31)	2015
dense_8 (Dense)	(None, 1)	32

Total params: 6,149,439
Trainable params: 6,149,439
Non-trainable params: 0

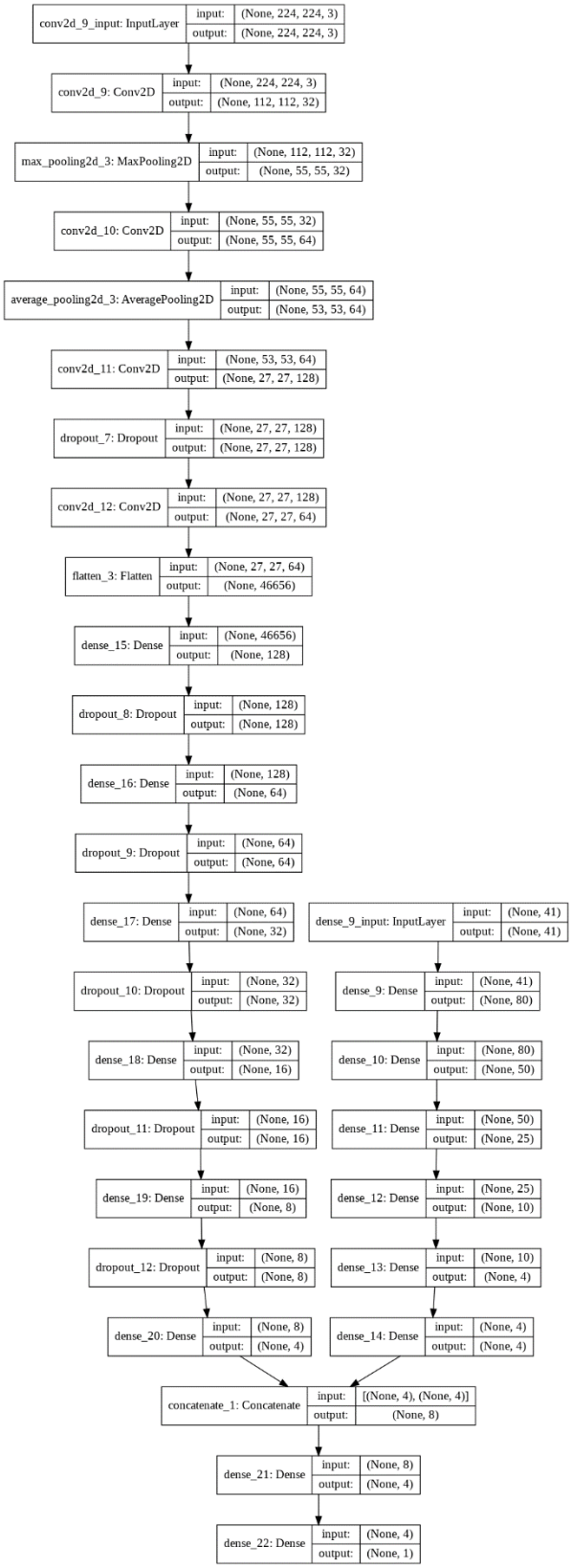
A8. Model Summary for NN (non-image features)

Model: "sequential_8"

Layer (type)	Output Shape	Param #
dense_29 (Dense)	(None, 32)	1344
dense_30 (Dense)	(None, 32)	1056
dense_31 (Dense)	(None, 32)	1056
dense_32 (Dense)	(None, 1)	33

Total params: 3,489
Trainable params: 3,489
Non-trainable params: 0

A9. Model Graph for 2nd CNN model (Combine image with other features)

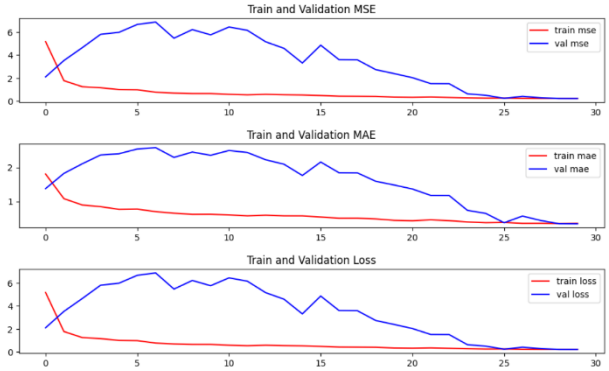


A10. Model Summary for InceptionV3 Model Top Layers

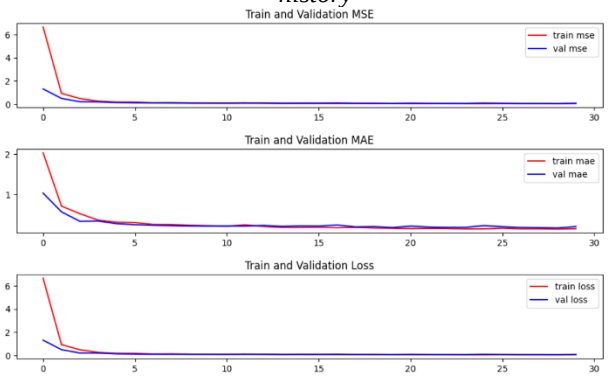
activation_94 (Activation)	(None, None, None, 1	0	ba
mixed10 (Concatenate)	(None, None, None, 2	0	ac mi co ac
dropout_45 (Dropout)	(None, None, None, 2	0	mi
global_average_pooling2d_1 (Glo	(None, 2048)	0	dr
dropout_46 (Dropout)	(None, 2048)	0	gl
dense_122 (Dense)	(None, 1024)	2098176	dr
dropout_47 (Dropout)	(None, 1024)	0	de
dense_123 (Dense)	(None, 1)	1025	dr

Total params: 23,901,985
Trainable params: 2,099,201
Non-trainable params: 21,802,784

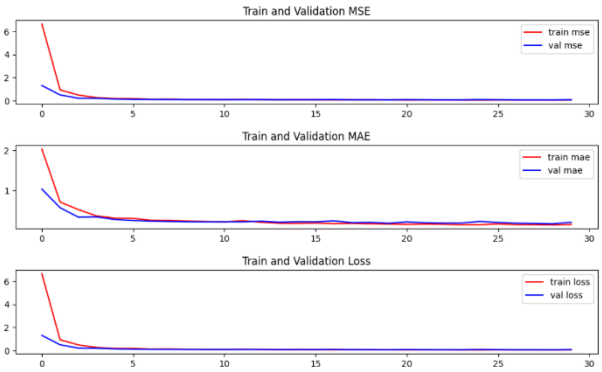
A11. Initial CNN model (Image alone) training/validation



A12. Second CNN model (image with other features combined) training/validation history
history



A13.InceptionV3 training/validation history



A14. CNN Model (image alone) Predicted Price vs Actual Price for selected listings

A15.CNN Model (image with other features) Predicted Price vs Actual Price for selected listings

A16. InceptionV3 Model (image alone) Predicted Price vs Actual Price for selected listings

A17. Tuned InceptionV3 Model (image alone) Predicted Price vs Actual Price for selected listings

A17. Ensemble Feature Rankings