Interpretable Machine Learning Models for Housing Price Prediction and Analysis

Chen Anqi[1], Chen Su[2], Wang Pinyuan[3], Liu Aiqi[4], Zhu Enyao[5], Bryan Wang[6]
[1]A0188533W, [2]A0188119W, [3]A0177848H, [4]A0204760M, [5]A0206188B, [6]A0216450N
CS3244 Machine Learning Project Team 36, School of Computing, National University of Singapore
{anqichen, e0323703, wang.pinyuan, liuaiqi, e0426111, bwang}@u.nus.edu

Abstract

We propose an app for interpretable prediction of housing prices in the Singapore housing market, and conduct detailed experimentation to illustrate the proposed usage and functionalities of our application to give end-users greater insights into the factors that affect housing prices. We empirically examine and compare three machine learning models: neural network, random forest and XGBoost. XGBoost is chosen for its best performance. A novel method proposed in interpretable machine learning, the Accumulated Local Effects (ALE) plot, is used to analyze and interpret the marginal effects of factors in our housing price model. Several interesting findings on the marginal effect on prices with respect to the underlying factors are also discussed.

Motivation & Problem Statement

According to the Department of Statistics Singapore, 90.4% of Singaporeans own a house, ranking second highest in home ownership rates worldwide. The housing market in the land-scarce and ultra-urbanised country is especially active and of significant domestic importance. Having an accurate model for house price prediction has proved important to and in high demand by both buyers and sellers in the market. However, existing machine learning models suffer from a common problem of limited functionalities and lack of interpretable analysis. In Singapore's urban context, a multitude of features can be factored into the housing market, such as *distances to nearest amenities*, and insights into these factors would greatly benefit all stakeholders in the market, empowering them to make better choices and more informed judgments.

We therefore propose an application which provides accurate price prediction and interpretable analysis on the price determining factors. Our proposed app fills existing information gaps and asymmetries, thus making property trading more convenient for both buyers and sellers and ultimately improving the efficiency of the real estate market in Singapore. To achieve this, our app utilises novel methods in interpretable machine learning to give users greater insights and deeper understanding of price models, which similar offerings today do not provide.

Overview of App Functionalities

We propose the following four functionalities for our app.

(1) Our app predicts the price of an existing property based on the user's input of an address and the type of flat.

(2) Users can specify changes in certain factors, which we call *exploratory factors*, to obtain insights on the marginal effect of the factor on property price. For example, users can determine the marginal effect on price for a house 100m away from a school relative to the average. Users can also specify certain *baseline factors* to direct their search; for example, if they are only interested in houses in a particular town, and the marginal effect displayed will be solely among houses in this town.

(3) After viewing the pricing information of a property, users can specify their acceptable price ranges for the marginal effect of certain features. Flats that satisfy the conditions will be shortlisted by our app.

(4) Beyond searching for property listings, users can also gain holistic insights on the current housing market situation. For example, the user may choose any town (eg. Jurong East) in Singapore and view findings on the more influential factors in this town. Our app compares the marginal price change of the town with the national average to determine whether a certain feature is important in determining flat price in this town. For example, if the national average for "distance to MRT station" is a premium of $50 at distance of 100m, whereas the marginal price change for Pasir Ris is $90 per 100m, this means "distance to MRT station" is a relatively more important factor in Pasir Ris.

With the above functionalities, users can customize specific features of their concern in property trading. Through the marginal effects of the features, they will have a more comprehensive understanding on which features are more important in determining the property price, making our model more interpretable and functional.

Experimental Setup

Datasets. We obtain our datasets 'Resale Flat Prices' from data.gov.sg, which contains the resale prices of all flats in Singapore from January 2017, with information about the flat unit (details in subsection 'Feature Engineering') and its resale price. We also use the *geosphere* package in R to obtain the geographical information of various facilities. Our final dataset consists of approximately 93k instances split randomly in an 80:20 ratio of training and test set.

In actual usage, our proposed app should take the most recent data within a predefined time interval (e.g. four years), in order to be representative of prevailing conditions affecting the house price.

Feature Engineering. The features we apply to our model are in two categories:

(1) *Flat information* provided in the 'Resale Flat Prices' dataset: *town*, *number of rooms*, *storey range*, *floor area*, *flat model*, and *remaining lease years*.

(2) *Information of the facilities near a flat unit*: the *nearest distance* to various facilities, including *MRT*, *hawker centre*, *hospital*, *supermarket*, *school*, *sports centre*, *mall*, and *bus interchange*. For the last two facilities, we also included the *number of malls within 3km & 5km*, and the *number of bus stops within 300m & 500m.*

Preprocessing. Most machine learning algorithms accept numerical inputs only. Hence, there is a need to encode categorical features such as *town* and *flat model*. As there is no ordinal relationship of values in *town* and *flat model*, one hot encoder is used for the two categorical variables.

Since the *unit price (per sqm)* of a flat is also of users' concern, we suspect that outputting *total price* by predicting *unit price* first may give better performance than using *total price* as class labels directly during training. Therefore, we pre-compute the *unit price* (*total price* divided by *floor area*) and compare both approaches (details in subsection 'Machine Learning Models Used').

Choice of ML models. The size of our experimental dataset indicates the expected scale of data that would be used in our proposed app. We select two tree-based models (Random Forest and XGBoost) and a neural network model. These models are able to perform regression tasks of outputting a single price based on our proposed features, and their characteristics suit the needs of our proposed app. Further elaboration is provided in the subsequent sections.

Performance metrics. We use *mean absolute error* (MAE), *mean squared error* (MSE) and *root mean squared error* (RMSE) to determine the accuracy of our models. Prices are in the unit of *10k* to give a more interpretable value for errors.

Machine Learning Models Used

Neural Network

Qualitative Advantages. Neural Networks find natural application to regression tasks by outputting a single value at the output layer. The key advantage of neural networks is their ability to learn rich representations easily, and unlike tree-based algorithms, are capable of fitting any continuous function, allowing them to capture complex trends in data.

Design of the Model. Additional data preprocessing steps need to be done for training the Neural Network. All data is normalised with *Standard Scaler* from the *sklearn.preprocessing* library. This step is of particular importance to the Neural Network model since features with larger values will be treated as more important in the training, which is not desired.

We examine two choices for the target label used in our regression task: *total price*, and *unit price (per sqm)*. We first implement a simple Neural Network with 3 layers as the baseline model. Next, we train the model using the two labels. For *unit price*, the *final total price* is computed by multiplying the *floor area*. Then, we use the evaluation metrics to compare the performances.

Method	MSE	RMSE	MAE
Total Price as Label	8.7460	2.9573	2.1998
Unit Price as Label	8.7246	2.9537	2.1902

Figure 1: Error metrics for NN baseline model in two methods

Both labels give similar accuracy on the baseline model. Hence, two fine-tuned models are trained based on the two labels separately. We use *RandomisedSearchCV* with *5-fold cross validation* to obtain a near-optimal set of hyperparameters. It turns out that using *unit price* as a label gives a greater improvement in performance at the end. Different subset combinations of the features are fed into the input layers and we find that using all features together gives the best performance.

The final structure of the model is as follows: The input layer takes in all features with normalized values. After that, we have 3 fully connected hidden layers with 128, 64, 32 units in each layer respectively. Lastly, the output layer contains only 1 unit. The best hyperparameters found are as follows:

Hidden Layer 1: kernel_initializer='normal', activation='selu'
Hidden Layer 2: kernel_initializer='normal', activation='relu'
Hidden Layer 3: kernel_initializer='normal', activation='relu'
Output Layer: kernel_initializer='orthogonal', activation='linear'
Model: loss='mse', optimizer='adam', metrics=['mae','mse'], epoch=300, batch_size=256

Performance Analysis. The following table shows the MSE, RMSE, and MAE scores for the Neural Network model on both validation (the whole training set is used for validation) and test.

Validation		Test	
MSE	5.4353	MSE	7.4936
RMSE	2.3314	RMSE	2.7374
MAE	1.7389	MAE	2.0060

Figure 2: Error metrics predicting total price for Neural Network

Limitation. First, more steps in preprocessing are required compared to tree-based models. Second, the neural network is a *black box* algorithm. It is very hard to interpret which factors are more important in determining the price. The training time is generally longer than tree-based models but it is still within tolerable range.

Random Forest

Qualitative Advantages. Simple decision trees are very sensitive to the data they are trained on, with small changes to the data resulting in huge changes in tree structures. Random forest overcomes this problem by making individual trees of random samples with replacement. Thus, random forest gives more accurate results than decision tree models, because the trees 'protect' each other from their individual errors. Moreover, as compared to neural networks, random forest has a clearer interpretation of the hyperparameters.

Design of the Model. Similar to the Neural Network model, we first compare the accuracy on *unit price per sqm* and *total price* as class labels using baseline random forest regressors.

Method	MSE	RMSE	MAE
Total Price as Label	7.4172	2.7234	1.9585
Unit Price as Label	7.1938	2.6821	1.9307

Figure 3: Error metrics for RF baseline model in two methods

This time, the total price *computed by* the predicted *unit price* is more accurate compared to using *total price* as class label *directly*. We will use unit price as the label for training. Next, we used *RandomisedSearchCV* to train and evaluate random forest models and obtain a near-optimal set of hyperparameter by taking random draws from a predetermined set of hyperparameter distributions.

Performance Analysis. The best parameters for random forest model are as follows:

'bootstrap':True, 'ccp_alpha':0.0, 'criterion':'mse', 'max_depth':100, 'max_features':'auto', 'max_leaf_nodes':None, 'max_samples':None, 'Min_impurity_decrease':0.0, 'min_impurity_split':None, 'min_samples_leaf':1, 'min_samples_split':2, 'min_weight_fraction_leaf':0.0, 'n_estimators': 200, 'n_jobs': None, 'oob_score': False, 'random_state': None, 'verbose': 0, 'warm_start': False

Validation		Test	
MSE	1.2386	MSE	7.1546
RMSE	1.1129	RMSE	2.6748
MAE	0.7978	MAE	2.6748

Figure 4: Error metrics predicting total price for RF

We observe that the performance of the random forest model is slightly worse than Neural Network, and it has a higher tendency to overfit.

Limitations. The large number of trees in the random forest results in significant time and computation needed for random forest models to produce a prediction for a given input in real-life predictions, and may therefore be computationally inefficient for usage and deployment in our proposed app.

Extreme Gradient Boosting

Qualitative Advantages. Extreme Gradient Boosting (XGBoost) is a decision-tree-based algorithm using highly optimized gradient boosting techniques. XGBoost's ensemble learning process iteratively combines new models with prior ones to correct existing errors, thus producing a robust final model. Moreover, its powerful gradient boosting approach utilizes both software and hardware optimization, including parallelized and distributed processing, depth-first tree-pruning, as well as more sophisticated regularization (both LASSO and Ridge) to avoid overfitting or bias. Therefore, XGBoost is able to achieve superior results with fewer computing resources in a shorter time as compared to other models, which is desirable for deployment in our proposed app.

As compared to neural networks, decision-tree-based algorithms excel at tasks where the source dataset is structured and the number of features is on a small-to-medium scale. Therefore, we anticipate that XGBoost would be better suited for our task setting.

Design of the Model. We use *XGBRegressor* from the *xgboost* Python library. Similar to the Neural Network model, we first compare the accuracy on *unit price per sqm* and *total price* as class labels using baseline random forest regressors.

Method	MSE	RMSE	MAE
Total Price as Label	6.5871	2.5665	1.8634
Unit Price as Label	6.5215	2.5537	1.8547

Figure 5: Error metrics for XGBoost baseline model in two methods

Again, the total price *computed by* the predicted *unit price* is more accurate compared to using *total price* as class label *directly*. Hence, we will use *unit price* as the label for training. Next, we used *RandomisedSearchCV* with *5-fold cross validation* to obtain a near-optimal set of hyperparameters. The best hyperparameters we found are as follows:

n_estimators=1000, max_depth=8, eta=0.1, subsample=1, colsample_bytree=0.6, learning_rate=0.08

Performance Analysis. The performance scores of XGBoost on both validation and test sets are tabulated below.

Validation		Test	
MSE	2.9225	MSE	6.3459
RMSE	1.7095	RMSE	2.5191
MAE	1.2774	MAE	1.8179

Figure 6: Error metrics predicting total price for XGBoost

We can see that XGBoost performs the best among all three models. Furthermore, in terms of time efficiency, XGBoost runs the fastest among these models with an average training time of 64s, whereas Neural Networks and Random Forest take 132s and 141s respectively.

Limitations. One limitation of tree-based algorithms in regression tasks is that such models are generally poor at extrapolation. This is because even when predicting continuous values, decision trees still use a *finite* set of rules to output leaf values within a *finite* range. In the case of our flat price prediction task, since the training dataset only contains prices between $140,000 and $1,258,000, we have empirically verified that the decision tree based *XGBRegressor* never predicts a price below $140,000 or above $1,258,000 for any test data. Hence, the model has some difficulty extrapolating target values beyond the range present in the training examples.

Nonetheless, the negative effect of this limitation is mitigated by the good quality of the datasets used, which comprehensively captures the sale transactions in the Singapore housing market. As such, the range of target values in the real-world test data is likely to be covered by that present in the training set.

Accumulated Local Effects (ALE)

Although housing price prediction models already exist, current offerings do not provide adequate insights into how individual factors in the model affect the price, but rather a black-box prediction of the house price. Yet, beyond the output prediction, buyers, sellers and other stakeholders in the housing market alike, often want to understand the effect of various factors on housing prices in Singapore, to make more informed judgments. Our proposed app thus aims to give users greater insight and understanding of the price models that we generate.

Therefore, methods in *interpretable machine learning* are of importance. Our proposed app seeks to explain changes in prices, according to marginal changes of various factors that buyers and sellers will find understandable and relevant, and further identify factors which contribute most to the price of a house in our model.

The **Accumulated Local Effects (ALE)** plot is a method in interpretable machine learning proposed by Apley and Zhu in 2016, which has the ability to accurately and efficiently (computation is possible in *linear time* w.r.t the number of samples) describe and interpret the marginal effects of a feature or a group of features in any given black-box machine learning model, while taking into account possible correlations or interdependencies between features.

Given a model with input instances $\vec{x}$ and output predictions $f(\vec{x})$, the ALE value of a given feature X_1, at a particular value $X_1 = x_1$, is defined as

$$f_{ALE}(x_1) = \int_{x_{1,min}}^{x_1} E[f^1(\vec{x}) \mid X_1 = t]dt - constant$$

where $x_{1,min}$ is a suitable constant lower bound for the range of values of X_1, and we define $f^1(\vec{x}) := \partial f(\vec{x})/\partial X_1|_{X_1=x_1}$ as the gradient which represents the *local effect* of X_1 on f at $X_1 = x_1$, and expectation is taken over the conditional distribution of $\vec{x}$ given $X_1 = t$. The overall effect of *accumulating* the *local effects* of feature X_1 based on the *conditional distributions* given X_1 is to isolate X_1 from any correlation with other features, and focus only on the marginal effect of X_1 on the prediction f. The ALE values are then shifted by a suitable constant so that the average effect, or ALE value, over the *entire* range of values of X_1, is zero (as one would expect).

For models where the gradient f^1 is not directly available, for instance in the tree-based models that we have selected, *Monte Carlo* methods are used to estimate this gradient, by sampling in a small interval, where intervals are taken at fixed percentiles of the range of values for X_1.

Illustration of App Functionalities

We illustrate the functionality of our proposed application from the perspective of a prospective house-buyer.

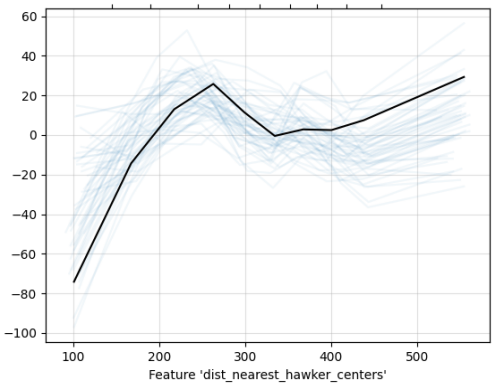
First, the prospective house-buyer may select *baseline features*, which are strict requirements that must be met in their search for a house. For example, he may specify that he is only interested in houses in the town of Ang Mo Kio.

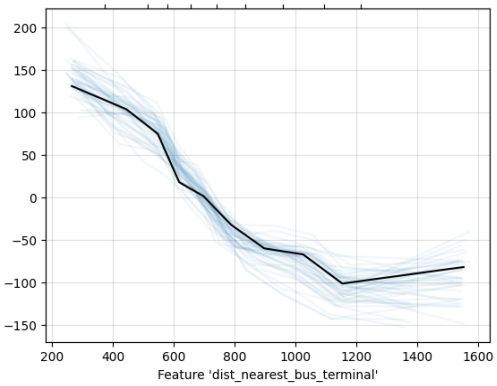
The house-buyer may then specify *exploratory features*, which are features or factors for which they are interested in exploring the marginal effect on the price, given houses that satisfy the *baseline features* they specified.

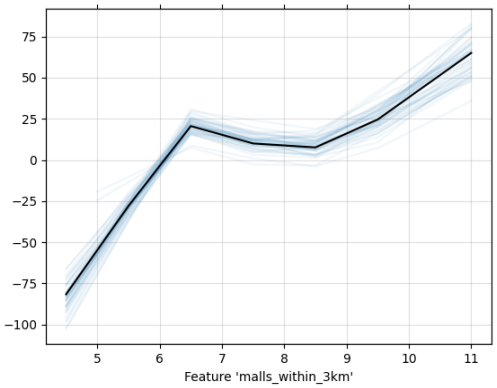
Based on our model of house prices, in this case, the *XGBoost* model, ALE plots are then calculated and

generated efficiently to represent the marginal effect of each *exploratory feature* on the price of houses satisfying their *baseline features*. For example, suppose a user has specified that he is interested in houses in the town of Ang Mo Kio (*baseline feature*), and his *exploratory features* are: i) *nearest distance to hawker center*; ii) *nearest distance to bus terminal*; iii) *number of malls within 3km*; and iv) *distance to nearest school*.

ALE plots are then generated representing the marginal effect of each *exploratory feature* on *price (per sqm)*, among houses satisfying the *baseline features*, that is, considering the input instances where the *baseline features* are equal to the user-specified values.







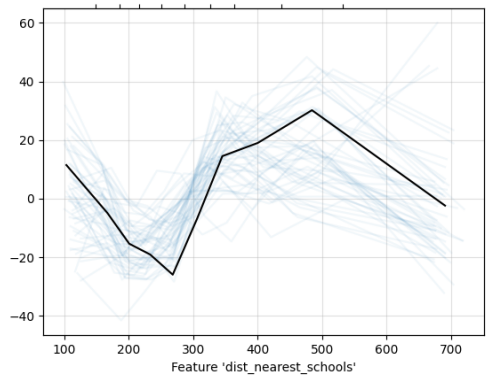


Figure 7: Sample ALE plots generated for four exploratory features, given a baseline feature of town = "Ang Mo Kio".

The y-axis of each plot represents the marginal change (relative to the average prediction) in *price (per sqm)* for the given value of *feature* (x-axis). Each thin line represents an iteration of the 50 *Monte Carlo* simulations used to estimate the gradient, and the bold line represents the obtained estimate.

These plots can be displayed to the user. Sliders can also be provided to allow the user to explore the marginal price change as the values for each feature are changed.

The ALE plots generated give clear insight and interpretation of the predictions generated by our model. For instance, there is a potential price penalty of up to $60 per sqm for houses which are within 100m of a *hawker center* (first plot). This can potentially be attributed to the noise and/or pollution associated with staying in such close proximity of a hawker center. However, a premium of $20 per sqm is expected for staying within medium distance from a hawker center, potentially due to the associated convenience.

Conversely, there is a premium of up to $150 per sqm for staying close to *the nearest bus terminal* in Ang Mo Kio, and savings of up to $50-100 per sqm for living far (>1km) from the nearest bus terminal (second plot). Although this is not plotted above, we additionally observed that while the ALE plot for *distance to nearest MRT station* exhibits a similar trend (as expected), the premium for staying close to the MRT station in Ang Mo Kio is relatively lower, at $50-$100 per sqm. We attribute this to the greater interconnectivity of bus services in Ang Mo Kio as compared to the MRT, leading to bus services having more weight on house values in Ang Mo Kio. We note also that the effect on price of distance features becomes relatively constant at large values, probably because the marginal effect of distance is no longer significant enough to factor into the value of the house.

Finally, an interesting phenomenon occurs with *distance to the nearest school* (fourth plot). Here, there is a penalty associated with being within 200-300m of a school, likely

due to the associated noise or traffic inconvenience, much as with the hawker centers. However, there is additional upward influence on the price as the distance further decreases, and this is likely driven by parents wishing to live near their child's school for their child's convenience.

Based on these insights, users are able to decide on the values of the various *exploratory features* that best fit their needs. For instance, a user may decide that he is only interested in units which are >600m from the nearest bus terminal, as he is not willing to pay the premium associated with staying closer to the bus terminal, but he will also want to consider units at <1km from the bus terminal, as he would still like the convenience of living near to the bus terminal at a negligible marginal price difference. He may also decide that he will not prefer units which are too close to a school (within 150m), as he would want to avoid the premium paid by parents to live near the school.

Based on this, the user can then input specified ranges of values for his *exploratory features*, and based on his inputs, property listings meeting his criteria are then displayed.

From the sellers' perspective, they can also use this information in deciding which aspects of their house are most marketable and are the key selling points or unique points of their house. For example, they may probably no longer want to highlight that their house is only <100m away from a hawker center; neither may they want to focus on marketing their house as being "only 1km" away from the bus terminal, as the effect no longer appears to be significant after the first 400m, but rather to focus on marketing their house as having >10 malls within 3km, for example.

Limitations & Extensions

Further research can be conducted on the interpretability of the model presented in our app. For example, it may be possible to automate aspects of the analysis or visualisation, so that our app can directly show the rank of factors based on their influenceability.

While ALE plots allow in theory for calculation of the marginal effect of several features jointly, visualisation for three or more features is usually not possible due to the limits of graph dimensionality (Molnar, 2021). We can further investigate ways to make visualisation possible for certain types of features, such as distance features.

In reality, factors determining housing price can be more complicated than the features we are choosing. It can be linked to the country's macroeconomic performance, social events, etc. More investigations on feature engineering could be done to improve the accuracy of the model.

Lastly, we have given careful consideration to the ethical issues that may relate to the usage of our model. First, all data used in our model are publicly available, and in particular have been fully anonymised. We need to ensure that all data used in our model after app deployment is similarly accessible in the public domain, and suitably sanitised and anonymised. Second, bearing in mind the target usage of our app in housing, an area which has direct implications for Singapore society, we have also deliberately avoided the use of demographics in our models, which has been known to introduce or reinforce social bias. Finally, we must make clear to our end-users that our app provides analysis and insight on the housing market as a whole and therefore serves as a guide, rather than making judgments subjectively. These considerations will mitigate any potential significant ethical issues arising from the usage of our app.

Conclusion

In our study, we compare the accuracy and applicability of neural networks, random forest and XGBoost models on house price prediction. We discover that the XGBoost model gives the best performance, with the lowest MSE, MAE and RMSE scores. We further explore the application of Accumulated Local Effects (ALE) on our XGBoost model, which analyses the marginal effects of a feature on the housing price. We thereby demonstrate the proposed functionalities and usage of our app, which we believe will significantly benefit the various stakeholders in Singapore's housing market.

Roles of Team Members

Chen Anqi: XGBoost model; **Chen Su**: Neural Network model; **Wang Pinyuan**: Random Forest model; **Liu Aiqi & Zhu Enyao**: Research and fine-tuning on NN model; **Bryan Wang**: ALE analysis; **All members**: Brainstorming functionalities and data preprocessing.

References

Apley, D. W., & Zhu, J. (2020). Visualizing the effects of predictor variables in black box supervised learning models. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 82(4), 1059-1086. doi:10.1111/rssb.12377

Brownlee, J. (2021, February 16). A gentle introduction to xgboost for applied machine learning. https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

Donges, N. (n.d.). A complete guide to the random forest algorithm. https://builtin.com/data-science/random-forest-algorithm

Households - latest data. (n.d.). https://www.singstat.gov.sg/find-data/search-by-theme/households/households/latest-data

Molnar, C. (2021, April). Interpretable machine learning. https://christophm.github.io/interpretable-ml-book/ale.html

Morde, V. (2019, April). Xgboost algorithm: Long may she reign! https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

Yiu, T. (2019, August). Understanding random forest. https://towardsdatascience.com/understanding-random-forest-58381e0602d2