Predicting Property Prices in New York City using Regression Analysis

---

Final Project
EECE 5644 - Intro to Machine Learning and Pattern Recognition
Alex Sadler
Section: Jennifer Dy
April 24, 2019

1 Introduction

With the recent price uncertainty in one of the largest and most expensive housing markets, New York City, I became interested in what factors affect property prices the most and how to build a model that accurately predicts prices [1]. A dataset found online, provided by the New York City Department of Finance titled "Rolling Sales Data," included relevant information for every housing property transaction over the past 12 months in New York City [2]. This dataset is suitable for determining how to optimally predict prices from a number of features since it includes information such as square footage, neighborhood, price, and more.

Furthermore, because this dataset has the most recent New York City property sales data (with only a one-month delay), is one of the few that are publicly available online, and is regulated by the Department of Finance, this project will allow for significant hands-on, practical experience in completing a machine learning project, from data collection to machine learning implementation and analysis. While some parts of the preprocessing will be challenging, it will help develop my programming skills and give me insight into what would actually need to be done in practice to complete a predictive modeling project such as this. Additionally, since I'm a Mathematics and Finance major (with a minor in Computational Data Analytics), I'm particularly interested in the role of predictive/statistical models in the financial industry.

For this project, the price of the property is the target/dependent variable, and the other variables will be the features/independent variables. In attempts to predict housing prices, it is common to apply the hedonic model developed by Rosen [3]. Under this model, price is assumed to be a function of certain characteristics such as location, size, neighborhood, and other features. The hedonic model typically uses linear regression analysis to find a function that explains the relationship between the inputs and price. After finding the linear function, the coefficients of the model are analyzed to determine which features have the greatest predictive power for the price.

While the hedonic pricing model is and has been an extremely popular framework for predicting housing prices, it has a number of drawbacks [4]. One of the main issues with hedonic pricing is choosing the functional form. Linear regression is the most commonly chosen functional form, but this may be incorrect for certain markets and would result in inconsistent estimates. Another issue relates to market segmentation. More specifically, in

any given market, there are likely different submarkets, and hedonic pricing provides no guidance in identifying or analyzing the different component markets.

Despite the drawbacks of hedonic pricing, it remains a widely used tool and provides a valid way of predicting housing prices from a set of features. Therefore, the first methods discussed in this paper are regression models. Linear regression is implemented first, followed by lasso and ridge regression. These models will then be compared to determine which provides the best predictive capabilities on test sets and is the simplest model. After these linear regression models, nonlinear algorithms such as decision trees, random forests, neural networks will be implemented and analyzed. The aim of this project is to create a property price prediction model using regression analysis that optimally predicts housing prices in New York City and to contribute to the existing research of comparing regression models for general property price predictions.

2 Data Acquisition and Pre-processing

In this paper, I use property sales data for the past twelve months in New York City, provided by the New York Department of Finance. Data from each borough is separated on the website but was concatenated into one table in order to analyze all of New York City transactions. It is worth noting that the models discussed throughout this paper could easily be applied to the individual boroughs, but the data is combined for simplicity and to predict prices for the city overall. The raw dataset had 81,747 rows and 21 columns.

For each transaction, the data offer information on 11 numerical variables: block, lot, easement, zip code, residential units, commercial units, total units, land square feet, gross square feet, year built, and sale price; 9 categorical variables: borough, neighborhood, building class category, tax class at time of sale, tax class at present, building class at present, address, apartment number, and building class at time of sale; and 1 date variable: sale date. Out of the 21 variables, six were removed from the dataset due to having no values or because they could not be processed with the algorithms I plan to use: easement, address, apartment number, building class at time of sale, and building class at present, and date.

The remaining categorical variables (borough, neighborhood, building class category, tax class at time of sale, tax class at present) were transformed into numerical quantities using one-hot encoding. This provided the benefit of including all of the variables in the analysis but at the cost of increased computational complexity with a sparse and significantly larger matrix. Before one-hot encoding, there were 18 features. After, there were 324 features.

The next step in pre-processing was to remove N/A values. Since there were only 64 rows that had at least one N/A value, all of them were removed. There were also 792 duplicate rows that were removed. After basic data cleaning and preparation, there was still more pre-processing required to make the dataset ready for machine learning algorithms. Since this dataset included information on all transactions and not just property sales, many entries in the dataset had a sale price of $0. Thus, rows that had a price less than $1,000 or more than $10,000,000 were filtered out to exclude transfers of property or extreme property sales values. While the "missing" housing prices could have been imputed with another value such as the mean or median, these rows were removed since imputing the target variable may skew the results and add noise to the results [5]. Additionally, two variables, gross

square feet and year built, had zero values for 28% and 6% of rows, respectively. Since the zero values for both of these variables were likely due to data collection or reporting errors, they were recast to N/A values to be imputed later in the pre-processing stage.

After completing most of the pre-processing, the dataset was split into training and test sets using stratified sampling based on the distribution of properties across the five boroughs. This was motivated by the substantial price differences in average property prices between the boroughs, and the distribution of number of properties was not evenly distributed. The training set was then used to impute the N/A values for gross square feet and year built using each variable's training set median. The mean and standard deviation of the sale price were then \$1,001,654 and \$1,178,061. The final step of the pre-processing stage was to scale the features. The means and variances of each feature of the training set were computed and used to normalize both the training and test set. This ensured variables with higher values would not be treated as more important than variables with lower values based on arbitrary differences in scales [6]. Additionally, scaling accelerates certain algorithms, such as gradient descent, so that they converge faster [7] Altogether, the pre-processing stage transformed the 81,747 x 21 raw dataset to a cleaned and algorithm-ready 54,730 x 325 dataset.

3 Methodology

3.1 Linear Models

With the pre-processing complete, the dataset was ready for machine learning implementation. The first set of predictive models that will be used and analyzed are linear regression models under the hedonic pricing model, as shown in equation (1).

$$y = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n \tag{1}$$

Where y is the predicted price of a property, $x_1, x_2, \cdots, x_n$ are the features of a property, and $\theta_0, \theta_1, \cdots, \theta_n$ represent the parameters/coefficients of the model used to predict housing prices. For this dataset, the independent variables are represented by $x_1 \cdots x_{324}$, which represent the dataset's original numerical variables and the one-hot encoded categorical variables, and y represents the sale price of a given transaction.

After basic linear regression is implemented, the other methods that will be used are lasso and ridge regression. In analyzing the performance of each of these models, cross-validation will be used to see how each model performs with different subsets of the dataset. The means and variances of the results will compared, and the most promising models will be investigated further. The models' performance will be compared using the root of the mean squared error (MSE shown in formula (2)). The root mean squared error (RMSE) is a natural and intuitive choice for the performance measure as it measures the average deviation of the estimates from the observed values.

$$MSE = \frac{1}{m} \sum_{i=1}^{m} (\hat{y_i} - y_i)^2 \tag{2}$$

Where $\hat{y_i}$ is the predicted value from the model, y_i is the observed value, and m is the number of data points.

3.1.1 Linear Regression

The first regression model to be implemented is ordinary least squares multivariate linear regression. This type of regression has the same hypothesis as the hedonic pricing model, as the relationship between the input variables and the output variable (price) is assumed to be linear as given in equation (1). Additionally, this form of regression is one of the earliest forms of regression, which has been used as a predictive technique since the early 1800's [8]. Ordinary least squares regression is a very popular method for predicting an output variable from a set of input variables. Some of its main advantages is that it is a simple model, is easy to implement, and is often easy to interpret.

However, some of this model's strengths are also drawbacks. Because it only attempts to discern linear relationships between variables, ordinary least squares cannot uncover non-linear relationships. Another issue with ordinary least squares, which is especially pertinent with this dataset given how sparse most of the feature matrix is, deals with prediction accuracy. Since the model treats all of the variables equally, and it is likely many variables in the dataset (especially many of the one-hot encoded categorical variables) do not have strong predictive capabilities, the model will have a tendency to overfit the training data [9]. While the bias for this model will be fairly low, it will likely have high variance and will be unable to make accurate predictions for unseen data.

All of the variables will be used to predict the housing prices for this model, and both the cost function and performance measure will be the mean squared error as shown in equation 2. The parameters/coefficients will be computed in Python using the LinearRegression tool from the Scikit-learn library [10]. This tool uses the normal equation as an analytic solution to find the parameter vector θ (equation 3).

$$\theta = (X^T X)^{-1}(X^T y) \tag{3}$$

3.1.2 Ridge Regression

Linear regression is able to quickly solve for the optimal model that predicts housing prices for the given training data; however, linear regression is prone to overfit the data, especially when the number of features is large. Linear regression will compute and return the coefficients for all 326 variables and will provide little guidance on determining which coefficients are the most important in the model. As all variables are treated equally, the model will likely have low bias and high variance. Options for addressing an overfitting model are to reduce the number of features and/or regularization. Since the number of features is very high, this section and the next on lasso regression address regularization to find better fitting models. Ridge regression can be used to increase the bias slightly while decreasing the variance [11].

Ridge regression is very similar to ordinary least squares regression, except that the coefficients are estimated by minimizing the mean squared error plus a regularization term, as shown in equation (4).

$$Cost(\theta) = \frac{1}{m}\left[\sum_{i=1}^{m}(\hat{y}_i - y_i)^2 + \lambda \sum_{j=1}^{n} \theta_j^2\right] \tag{4}$$

Where λ is the regularization parameter. The regularization term helps to control a trade-off between two objectives. The first objective is to fit the training data well. The second

objective is to reduce the size of the parameters to avoid overfitting. By including the regularization term in the cost function, ridge regression forces the model coefficients towards zero, where higher valued weights/coefficients will be penalized more than smaller coefficient terms since the coefficients are squared in the cost function. To select an optimal value for λ, I will use grid search (where the grid values range from 1 to 100 with steps taken in approximately multiples of three) with 10-fold cross validation. The optimization algorithm used to fit the model is automatically selected in Scikit-learn based on the dataset.

3.1.3 Lasso Regression

The final linear model implemented is lasso regression. It is nearly identical to ridge regression, except that the absolute value of the coefficients is used in the regularization term instead of squared values (equation (5)). Using the absolute value instead of the square of the coefficients has the effect of forcing all terms towards zero and some to exactly zero. Therefore, lasso regression performs both shrinkage estimation and variable selection [12]. Given that this dataset has 324 features, using lasso regression and having coefficients equal to 0 should help with model interpretability. Additionally, it shares many of the same advantages as ridge regression over ordinary least squares regression, especially in that it often improves overall model accuracy [9].

$$Cost(\theta) = \frac{1}{m}\left[\sum_{i=1}^{m}(\hat{y}_i - y_i)^2 + \lambda\sum_{j=1}^{n}|\theta_j|\right] \tag{5}$$

Similarly to the methods used for ridge regression model tuning, the regularization parameter λ will be found using grid search (with the same grid values as in ridge regression) with cross-validation. The default algorithm Scikit-learn uses to fit the best model is coordinate descent. While few other models use coordinate descent to fit the data, this algorithm has the benefits of simplicity, speed, and stability [13].

3.2 Nonlinear Models

The linear regression models in the previous section are relatively simple to implement, quick to run, and easy to understand. While these models have their merit, they also suffer from a significant drawback: their assumed functional form. Essentially, when using a set of input features to predict the price of a piece of property, the hedonic pricing model is being used. This is a significant assumption of the relationship between the inputs and output, and only allows for linear relationships and explanations. Instead of using the hedonic pricing model (or slight variants/modifications of it), other nonlinear regression models, which have been used successfully in other housing markets, are used to optimally predict the price of property in this section [14].

3.2.1 Decision Trees

Decision trees are the first nonlinear model used to predict property prices in this paper. Instead of building a single predictive formula for all 300+ variables (which was done in 3.1) an alternative method is to first assume that the characteristics of property are functionally

separable and then group properties together to form a hierarchal structure (a decision tree) – a method that was first applied to housing in 1973 by Apps [15]. Under this framework, the price of property depends on a set of intermediate variables and functions.

With decision trees, the first step is to recursively partition the predictor space. After the feature space has been segmented into a number of distinct regions, predictions can be made for a datapoint depending on which region is classified as. To determine which region a datapoint belongs in, the datapoint starts at the root node and is classified based on a sequence of rules. To predict the final value based on the classification, a simple average is taken of the output values in a given region. Scikit-learn uses the Classification And Regression Tree (CART) algorithm to train decision trees. It first splits the training set optimally into two subsets using a single feature, and then it repeats this step at each level until it reaches a maximum depth or if splits do not improve the model's performance [16].

One of the main advantages of using decision trees, whether it is for classification or regression, is model interpretability. Since the trees can be displayed graphically, the sequence of rules can be interpreted quickly and easily, even by non-experts. For this dataset, since there are a large number of variables, decision trees can help to determine which features are the most important for predicting prices. This will also help for predictions, as each new datapoint inputted into the model can be quickly sorted based on a sequence of rules.

Some of the main problems with decision trees are model robustness and predictive accuracy. Slight changes in the dataset can often lead to a very different set of rules. These issues can be addressed by adding regularization; however, that adds the issue of model and computational complexity since there are many hyperparameters in decision trees. Using grid search with cross-validation for all of the hyperparameters would be very computationally expensive, so only three are tuned: the maximum depth (with values from 1 to 25 in steps of 5), the minimum number of samples required to split an internal node (ranging from 0.01 to 0.30), and the minimum number (or fraction) of instances in a leaf (ranging from 0.01 to 0.30). These were chosen as they have been found to be the most responsible hyperparameters for the performance of final trees in empirical studies of hyperparameters [17].

3.2.2 Random Forests

Decision trees are useful and easily interpretable models that can be used to predict property prices, but they are prone to having fairly high variance. In general, a popular way to get better results than a single predictor is to use ensemble methods. Ensemble methods combine multiple learning methods into one predictive model to produce a single optimal output. The general idea is that using the results from multiple models will yield better performance than a single model could. Ensemble methods can be used in the context of decision trees to train the model on many different random subsets of the training set with replacement, which is known as bagging, and averaging the predictions of all trees. This is the general process that random forests follow.

Random forests are random in two ways. First, each tree is based on a random subset of the observations, and second, each split within the tree is based on a random subset of the variables [18]. Since trees are quite unstable, the randomness will create different predictions for each tree, and then the final predictions are averaged across the trees. Thus, the model is much less prone to overfitting the data than any single tree is. This can increase the bias of

the model but at the benefit of substantially lowering the variance and improving the overall model's predictions. Although random forests often have superior performance compared to decision trees, they are not as easily interpretable since averages are taken over a large of number of trees.

Similar to the model tuning methods discussed in other sections, grid search and cross validation are used to find optimal hyperparameters. The only hyperparameters tuned for the random forests implementation are the number of trees (where the possible values range from 1 to 200 in multiples of 50) and maximum tree depth (ranging from 1 to 25 in roughly steps of 5). These were chosen since having a large number of trees will hopefully mitigate any specific tree overfitting the training data and due to computational costs (since grid search with cross-validation for random forests can take a long time).

3.2.3 Neural Networks

The final nonlinear model to be implemented is a fairly simple neural network. Many of the motivations for using a neural network to predict housing prices is similar to that for the decision trees. While the hedonic pricing model is a common base framework to use for predicting prices, other methods may be more suitable, especially with the increasingly complex datasets available. In this dataset with 324 variables and over 50,000 examples, it is worth considering other more complex models that may be able to find nonlinear relationships between variables.

The model used is a feedforward neural network with 324 inputs, 2 hidden layers, 324 hidden neurons, and 1 final output (the predicted property price). The activation function used is ReLU for its speed and efficiency [19]. There are many other choices that could have been used for this model, which is one of the drawbacks of neural networks. Deciding how many hidden layers to use, how many neurons in each layer, and experimenting with all of the parameters to find the optimal model is typically not feasible given how many different possibilities there are. Thus, while this model is likely not optimal, it is a reasonable starting point for building more complex neural networks. Regarding implementation, the neural network is implemented in Python using Keras with TensorFlow backend, and early stopping is used to stop the model if it has not converged after 200 iterations.

In addition to the issues with model complexity, the process and results from the neural network are hard, if not impossible, to interpret. In fitting complex relationships to the model, interpretability is sacrificed for greater predictive power. The model may be able to find relationships not considered by other models, but with other models it is possible to see roughly how the regression models make predictions.

4 Experiments and Results

The first part of fitting the models discussed in section 3 was to obtain the optimal hyperparameters each model used to predict housing prices. For each model, except for the neural network, a range of values was altered using grid search, and then for each value, cross-validation was performed to see how well that parameter performed on the given training/validation data. For ridge and lasso regression, the options for the regularization param-

eter λ were 1, 3, 10, 30, and 100. For decision trees, the maximum number of layers ranged from 1 to 25 in multiples of 5 and for both the minimum number of samples to split and minimum number of samples in a leaf, the values ranged from 0.01 to 0.30 in multiples of 0.5. For random forests, the maximum depth was set to 15 and the values for number of estimators ranged from 1 to 200 in multiples of 50. No hyperparameter tuning was conducted for the neural network. Table 1 shows a summary of the hyperparameters used throughout the models, where cell values denote the optimal values selected from a set of values, and where blank cells denote if a hyperparameter did not apply to a model or was not optimized.

Table 1: Hyperparameters Summary

Model	λ	Max Depth	Min Split	Min Leafs	Number of Trees	Hidden Layers	Neurons per Layer	Activ. Function
Linear Regression								
Ridge Regression	30							
Lasso Regression	10							
Decision Trees		15	0.01	0.01				
Random Forests		15			150			
Neural Networks						2	324	ReLU

After using grid search with cross-validation to find optimal hyperparameters, each model was used to evaluate the performance of making predictions for property sales prices. The models used various cost functions to fit the data, but they were all compared using the RMSE and R-squared. A summary of the performance for each model for training and test sets is shown in table 2. All models had lower RMSE of predicted property prices than the standard deviation of housing prices, which was 1,178,061.

Table 2: Performance Summary

Model	Train RMSE	Test RMSE	Train R^2	Test R^2
Linear Regression	880,410	833,052	0.45	0.48
Ridge Regression	880,320	832,624	0.45	0.48
Lasso Regression	880,425	832,774	0.45	0.48
Decision Trees	797,636	768,131	0.55	0.55
Random Forests	418,550	615,539	0.88	0.71
Neural Networks	829,786	796,625	0.01	0.08

The linear models had nearly identical performance for both RMSE and R-squared across training and test samples. In all cases, the test set R-squared was slightly higher than the training set R-squared. Some of largest coefficients (which can be used to compare the relative importance of variables) across all three of the linear regression models were residential property tax class at present, cooperatives property tax class at present, and residential property tax class at time of sale.

The nonlinear models had much more varied performance than the linear models did. The decision trees implementation had lower RMSE scores and higher R-squared values

than the linear models, and including regularization helped to prevent overfitting. In this case, similar to the linear models, the test RMSE was actually lower than the train RMSE. Random forests had the best performance out of any model, but performance on the test set was much worse than performance on the training set. The top three most important features for the random forest were gross square feet, whether a property was in Manhattan or not, and block number, with Gini importance values of 0.41, 0.13, and 0.12, respectively. The neural network had the lowest R-squared values of any model, but the RMSE for the train and test sets were approximately average.

5 Discussion and Conclusion

In this paper, several types of models were used and compared to determine which was able to predict property prices most accurately. The first group of models, linear models, were implemented using the hedonic pricing model, where the price of a property is assumed to be a combination of a set of input values. This framework is often used as a baseline comparison for regression models, since this model has been used longer than any other to predict prices, and it is simple to use and understand.

Ordinary least squares was the first regression model, and it performed reasonably well. However, using some type of regularization is typically recommended to prevent overfitting, so ridge and lasso regression were experimented with as well. After using grid search with cross validation to tune the regularization parameters, ridge and lasso regression surprisingly still did not perform better than ordinary least squares. This may be due to the very large number of features in the model, with many possibly not having that large of an impact on the predicted price. Additionally, even with a fairly large regularization parameter value of 10 for lasso regression, the model only reduced the number of non-zero coefficients to 321 from 324. Thus, the model did not perform as well as expected in regards to its feature selection capabilities.

All linear models had an R-squared of 0.45 on the training set and 0.48 on the test set. The improved performance on the test set was likely due to the similarities in the two sets, and overall, these models had low variance. Even though the performance results were very similar for all models, they each differed slightly in the coefficients rankings. For example, the most important feature for linear and lasso regression were tax class at time of sale 1 (the tax class for residential properties), while ridge regression's most important feature was the rental building class category. These variables were not expected to be the most important variables (especially since they were one-hot encoded variables), so other methods helped to explore important features.

The next set of regression models were nonlinear: decision trees, random forests, and neural networks. The hedonic pricing model in the previous section has the benefits of simplicity and intuitiveness; however, it assumes that the relationship between variables is linear. Nonlinear methods can help to find complex relationships and can have better predictive power than linear models. Using decision trees had notably better performance than the regression models, measured both by RMSE and R-squared. Instead of using a single equation to explain the price, decision trees create a sequence of rules that new data points can be passed through. With housing prices, this seems like a reasonable way to

predict prices and provides good interpretability. The most important feature clearly was gross square feet with a Gini importance value of 0.63. While this model uses this variable much more than any other to predict prices, it is reasonable since square footage in such a compact city is extremely valuable. Large homes in New York City will be far more valuable than smaller homes, whereas in rural, spacious areas this is not always the case.

Decision trees are useful for predictions and to understand what the most important features of a dataset are, but they are prone to overfitting the training data. Instead of using a single tree and tuning the hyperparameters further, an ensemble of decision trees was used to make predictions, which is also known as a random forest. This method proved to be the best model with a RMSE and R-squared on the test set of 615,539 and 0.71, respectively. The RMSE for the random forests model is nearly half the value of the standard deviation of the price. While 615,539 is still a fairly large RMSE, it can be useful to use this to predict a range of prices for properties. In this model 150 trees were used, each with a maximum depth of 15.

This model had all of the benefits of decision trees discussed above with the added benefit of additional randomness and that it averaged results across a large number of trees. The most important features, as discussed in section 4, were gross square feet, whether a property was in Manhattan or not, and block number. This gives significant insight into the determinants of housing prices that was not possible with the original dataset, since boroughs were a categorical variable that could not be processed initially. Therefore, it may be better to implement the models used in this paper for each borough, rather than all of them together. There are significant differences between boroughs, and it may not have been appropriate to treat all of the boroughs in New York City as a homogenous geographic unit.

The final model used was a neural network. This model was the most complex and provided the least amount of interpretability with only average performance results in regards to RMSE. However, the neural network used was a very basic one, and no hyperparameter tuning was conducted. Further research would need to be conducted to determine a more optimal neural network to predict housing prices.

Overall, this final project provided significant insight into implementing machine learning models, starting from basic data acquisition and pre-processing to hyperparameter tuning and model evaluation. Many different models were implemented starting from basic OLS linear regression to neural networks. Basic models under the hedonic pricing model provide a good starting point for comparing regression techniques, but given how much larger and more complex datasets are today than when hedonic pricing was initially created, other models, such as decision trees, random forests, and neural networks, can provide strong predictive power sometimes with better interpretability. In general, this paper's contribution lies in the comparison of using various models to predict property prices, regardless of the geographical area.

References

[1] J. Barbanel, "Manhattan apartment sales in 2018 sink to low hit after financial crisis manhattan apartment sales in 2018 sink to low hit after financial crisis," *Wall Street Journal*, 2018.

[2] "Rolling sales data," NYC Department of Finance, February 2019.

[3] S. Rosen, *Hedonic Prices and Implicit Markets: Product Differentiation in Pure Competition.* Journal of Political Economy, 1974, vol. 82.

[4] C. K. Wing, *A Critical Review of Literature on the Hedonic Price Model.* International Journal for Housing Science and Its Applications, June 2003, vol. 27.

[5] P. T. von Hippel, *Regression with Missing Ys: An Improved Strategy for Analyzing Multiply Imputed Data.* Sociological Methodology, May 2007, vol. 37.

[6] D. W. Marquardt and R. D. Snee, "Ridge regression in practice," *The American Statistician*, vol. 29, February 1975.

[7] T. Li, B. Jing, N. Ying, and X. Yu, "Adaptive scaling," *Annals of Applied Statistics*, September 2017.

[8] X. Yan, *Linear Regression Analysis: Theory and Computing.* World Scientific, 2009.

[9] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of Royal Statistical Society*, vol. 58, 1996.

[10] "scikit-learn." [Online]. Available: https://scikit-learn.org/stable/index.html

[11] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning.* Springer, 2013.

[12] M. R. Osborne, B. Presnell, and B. A. Turlach, "On the lasso and its dual," *Journal of Computational and Graphical Statistics*, vol. 9, no. 2, June 2000.

[13] T. T. Wu and K. Lange, "Coordinate descent algorithms for lasso penalized regression," *The Annals of Applied Statistics*, vol. 2, no. 1, March 2008.

[14] G.-Z. Fan, S. E. Ong, and H. C. Koh, "Determinants of house price: A decision tree approach," *Urban Studies*, 2006.

[15] P. F. Apps, "An approach to urban modelling and evaluation. a residential model: 1. theory," *Environment and Planning*, 1973.

[16] A. Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow.* O'Reilly, 2017.

[17] R. G. Mantovani, T. Horváth, R. Cerri, S. B. Junior, J. Vanschoren, and A. C. P. L. F. de Carvalho, "An empirical study on hyperparameter tuning of decision trees," 2018.

[18] U. Grömping, "Variable importance assessment in regression: Linear regression versus random forest," *The American Statistician*, vol. 63, no. 4, 2009.

[19] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," *ICML*, 2010.