USING MACHINE LEARNING TECHNIQUES TO PREDICT HOUSE PRICES IN MELBOURNE

DATA SCIENCE 871

Stellenbosch

UNIVERSITY
IYUNIVESITHI
UNIVERSITEIT

Amy Marais
20933835

Contents

List of Tables

Introduction

I am interested in predicting the selling prices of houses in Melbourne, Australia, and I explore a range of supervised machine learning techniques in RStudio to do so. I start by fitting and evaluating linear regression models and discuss the relevant ridge and lasso penalties. Next, I construct of a simple decision tree, and subsequently try to make predictions more accurate by building and evaluating a random forest learner. Lastly, I implement gradient boosting techniques to get the most accurate predictions as possible. The advantages and shortcomings of techniques are discussed throughout.

Data Exploration

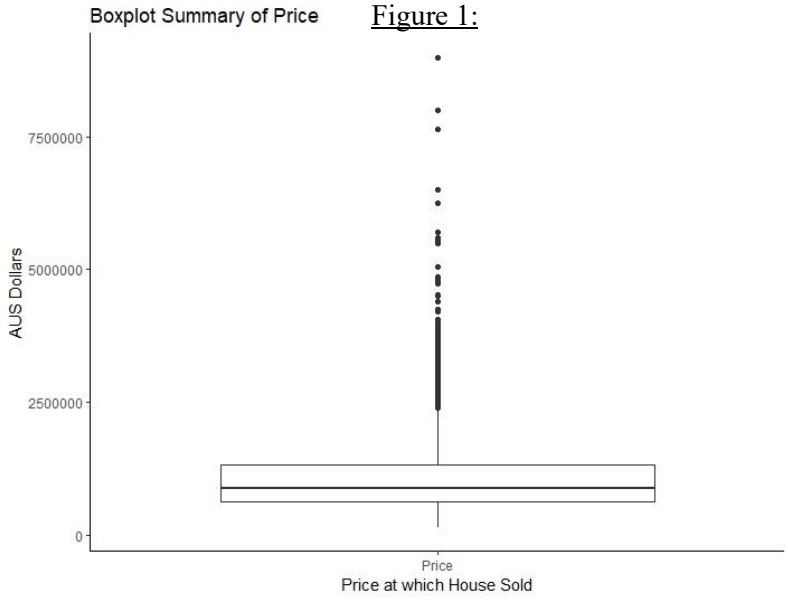
My dataset is a snapshot of the Melbourne housing market that has been scraped from publicly available results on Domain.com.au. The dataset includes address, type of real estate, suburb, method of selling, the number of rooms, price at which the property sold, the name of the real estate agent, the date of sale and distance of the house from the CBD. The appendix has a list of all column names and unique entries, as well as their descriptions. Where appropriate, the dataset is split into training and testing datasets in a 70-30 ratio, using a simple random sampling method. The numerical data can be summarised as follows:

Table 1: Data Summary

	Price	Rooms	Distance	Beroom2	Bathroom	Car Port	Land size	Building Area	Year Built	Latitude	Longitude	Property Count
Minimum	131000	1	0	0	1	0	0	0	1196	-38.16	144.5	381
25th Percentile	620000	2	5.9	2	1	1	152	91	1940	-37.86	144.9	4 384
Average	1068828	2.931	9.75	3	1	1	373	124	1970	-37.80	145.0	6 567
Median	880000	3		2.902	1.579	1.574	471	141.6	1964	-37.81	145.0	7 4535
75th Percentile	1325000	4	12.4	3	2	2	928	170	2000	-37.76	145.1	10 175
Maximum	9000000	8		9	8	10	37000	3 112	2018	-37.46	145.5	21 650

The average house in Melbourne sold for 1 068 282 Australian dollars and had approximately 3 rooms, 1 bathroom, 1 to 2 car ports, is located 9.75km from the CBD, has 373m2 open land while the building area is 141.6m2, was built in the 70s and has a property count of 6 567. We

Figure 1:



are interested in predicting house prices so let's get a better understanding of the price statistics by means of a boxplot.

Majority of the houses sold in Melbourne are priced between 620 000 and 1.3 million Australian dollars, but there are several sales of 'luxury' houses that are priced well above this range.

Model evaluation

Before we can start modelling, we need to understand how models are evaluated and compared. Predictive accuracy is generally assessed through loss functions that output "errors". The error is the difference between the predicted and actual values, and there are several loss functions that can be utilised. We can broadly divide our models into either regression or classification models. We want to create a model that predicts house prices, which is a numeric outcome, hence we will be dealing with regression models in this paper. Categorical models predict a categorical outcome, or the probability of a categorical outcome occurring. Regression models are generally evaluated using three loss functions: RMSE, MAE and R-squared.

Root Mean Square Error (RMSE) is a metric used to measure the average magnitude of errors between predicted and actual values. It calculates the square root of the average of the squared differences between predicted and actual values. RMSE is sensitive to outliers and larger errors. Lower RMSE values indicate better model performance, with a value of 0 indicating a perfect fit.

Mean absolute error (MAE) measures the average absolute difference between predicted and actual values. MAE is less sensitive to outliers compared to RMSE since it does not involve squaring the errors. Smaller MAE values indicate better model performance, with a value of 0 indicating a perfect fit. MAE is useful when the focus is on the magnitude of errors rather than their direction.

R-squared measures the proportion of variance in the dependent variable explained by the model. It provides an indication of how well the model fits the data compared to a baseline model (typically the mean of the dependent variable). R-squared ranges from 0 to 1, with higher values indicating a better fit. However, R-squared has limitations, such as being influenced by the number of predictors and not providing information about the accuracy of individual predictions.

To evaluate a machine learning model, you can calculate these metrics using the predicted and actual values from the test dataset. Lower RMSE and MAE values and higher R-squared values generally indicate better model performance.

Regularised Regression

Linear regression is a popular technique used in machine learning to make predictions. It involves fitting a linear equation to the given dataset, where the target variable is modelled as a linear combination of one or multiple predictor variables. There are two main types of linear regression: single linear regression and multiple linear regression.

Single Linear Regression

Single linear regression uses only one predictor variable to model the relationship with the target variable. In this case, we can model the building area as a predictor of the sale price, as described by equation (1) and Figures 2 and 3:

$$(1)\ \text{Price} = \beta_0 + \beta_1 \text{BuildingArea} + \varepsilon$$

Figure 2:

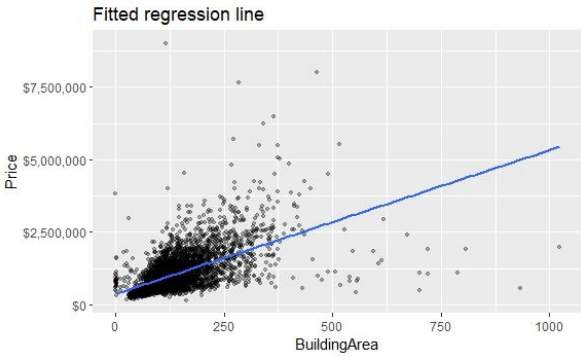
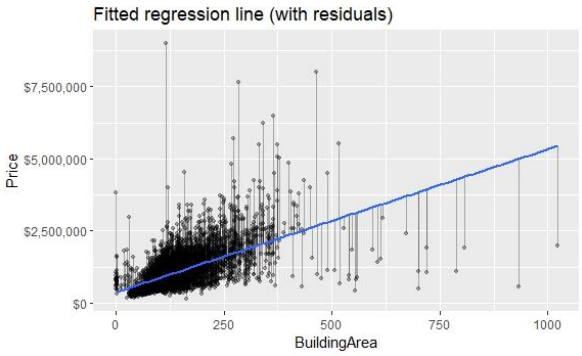


Figure 3:



The regression is performed by means of Ordinary Least Squares (OLS) which implies the sum of the squared residuals is minimised, which necessitates that in-sample fit is optimised. The disadvantage of this model is that it assumes a linear relationship between the predictor and the target variable, meaning that it may not capture complex patterns or interactions between multiple predictors. Linear models in particular exhibit high levels of bias since they are not as flexible as non-linear alternatives. However, the advantage of single linear regression is that it's easy to interpret and is useful when examining the relationship between a single predictor and the target variable to determine a causal effect. From the figures we can see that the size of the building area has a positive effect on the sale [rice of a house, particularly for houses smaller than approximately 500m^2. It seems that the building area does not have a strong influence on price once they become larger than 500 m^2, other factors like the size of the land surrounding the house may be more influence on price after that point.

The results and statistic diagnostics for the linear regression:

Table 2: Simple Linear Regression Results

Regressor	**Estimate**
Building Area	3 658.12
(s.e.)	(92.43)
Intercept	546 131.39
(s.e.)	(15711.89)
Residual s.e.	572500
Degrees of Freedom	4335
F-statistic	1589 on 1 and 4335 df
P-value	< 2.2e-16
R2	0.2683
Adjusted R2	0.2681

Table 3 shows the MAE, RMSE and adjusted R-squared scores for the model after cross validating the sample 10-fold, which we compare to the multiple regression model's scores later in the paper. The model does not have very strong predictive powers, with predictions being off by approximately 400 000 to 600 000 AUS dollars on average. We will likely improve the predictive power by adding more predictor variables to the model.

Table 3: Out of sample performance - Single Linear Regression Model – 10 resamples

	Min	1st Quartile	Median	Mean	3rd Quartile	Maximum
MAE	352778.3	369394.9	382817.2	385938.8	405559.7	419148.9
RMSE	456398.4	521232.7	564183.4	577849.1	621518.4	756851.1
R^2	0.1491363	0.2718527	0.3247291	0.3157315	0.3835738	0.429469

Multiple Linear Regression

Multiple linear regression incorporates multiple predictor variables to model the relationship with the target variable. In this case I model the sale price of houses in Melbourne with the other numeric variables available in the dataset, as represented by equation (2).

$$(2)\ \text{Price} = \beta_0 + \beta_1\text{BuildingArea} + \beta_2\text{YearBuilt} + \beta_3\text{Rooms} + \beta_4\text{Landsize} + \beta_5\text{Distance} + \beta_6\text{Bedroom2} + \beta_7\text{Bathroom} + \beta_8\text{Car} + \varepsilon$$

The advantage of multiple linear regression is that it allows for modelling relationships between multiple predictors and the target variable and that it can capture interactions and dependencies among the predictors. However, the disadvantage is that it assumes a linear relationship between the predictors and the target variable, which may bias results.

Table 4: Multiple Linear Regression Results

Regressor	**Estimate**
Building Area	1.778e+03
(s.e.)	9.306e+01
YearBuilt	-5.087e+03
	1.916e+02
Rooms	8.491e+04
	2.438e+04
Landsize	2.060e+01
	7.268e+00
Distance	-2.868e+04
	1.326e+03
Bedroom2	5.316e+04
	2.400e+04
Bathroom	2.976e+05
	1.342e+04
Car	6.743e+04
(s.e.)	(8.330e+03)
Intercept	1.010e+07
	3.750e+05

Residual s.e.	449100
Degrees of Freedom	4328
F-statistic	662.3 on 8 and 4328 df
P-value	< 2.2e-16
R2	0.5504
Adjusted R2	0.5496

Table 3 shows the MAE, RMSE and adjusted R-squared scores for the model after cross validating the sample 10-fold. The multiple regression model's predictions is off by approximately 300 000 to 450 000 AUS dollars on average. This is an improvement on the simple linear regression model, but the model still does not have very strong predictive powers.

Table 7: Out of sample performance - Multiple Linear Regression Model – 10 resamples

	Min	1st Quartile	Median	Mean	3rd Quartile	Maximum
MAE	269521.1	286592.3	295016.2	296972.9	308978.0	320446.4
RMSE	374530.4	419866.6	446173.6	454922.3	501341.3	513946.7
R^2	0.4571742	0.5212402	0.5508649	0.5447454	0.5733331	0.595775

Ridge and Lasso Penalties

Ridge and Lasso are regularization techniques used in regression models to prevent overfitting and improve the model's generalization performance. They involve adding a penalty term to the regression objective function to control the complexity of the model. Both Ridge and Lasso penalties provide a form of regularization to prevent overfitting and improve model performance. Ridge tends to work well when dealing with multicollinearity, whereas Lasso is beneficial for feature selection by promoting sparsity in the coefficient vector. Elastic Net regularization, which combines both Ridge and Lasso penalties, provides a balance between the two penalties.

Ridge Penalty

Ridge regularization, also known as L2 regularization, adds a penalty term to the regression objective function that is proportional to the sum of squared coefficients. The penalty term encourages the model to have smaller and more balanced coefficients, reducing the impact of individual predictors. Ridge regularization helps to mitigate the issue of multicollinearity (high correlation) among predictors by shrinking their coefficients towards zero. The amount of regularization is controlled by a hyperparameter called the regularization parameter (lambda). A larger value of lambda results in greater shrinkage of coefficients. Ridge regression can be expressed as minimizing the sum of squared errors plus the penalty term:

$$(3)\ \text{Min}(\text{RSS} + \lambda\ \text{sum}(\text{coefficient}^2))$$

Lasso Penalty

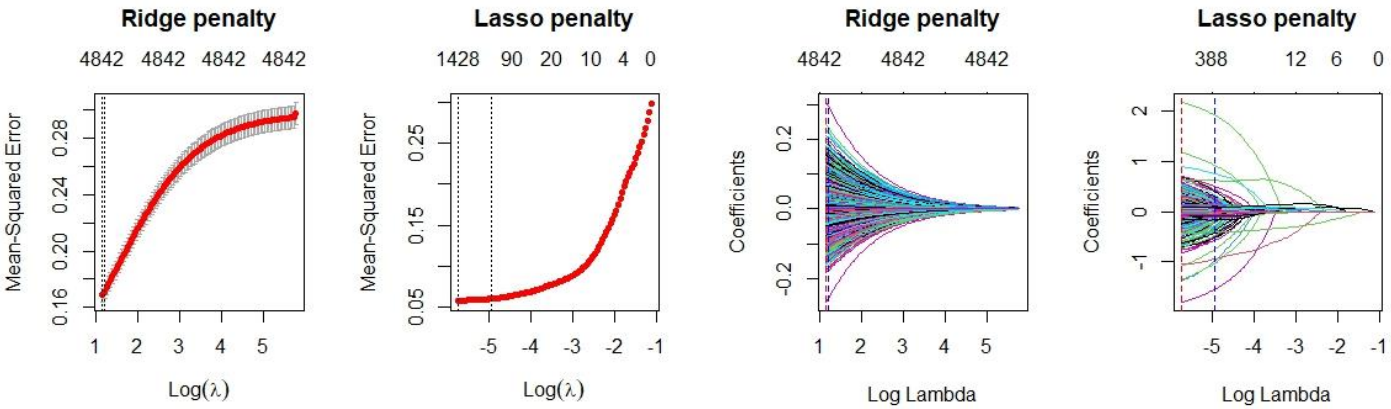
Lasso regularization, also known as L1 regularization, adds a penalty term to the regression objective function that is proportional to the sum of absolute values of coefficients. The penalty term encourages the model to have sparse coefficients by driving some coefficients exactly to zero. Lasso regularization is useful for feature selection as it tends to eliminate irrelevant predictors by setting their coefficients to zero. Similar to Ridge, the amount of

regularization is controlled by a regularization parameter (lambda). Increasing lambda increases the amount of shrinkage and sparsity of coefficients. Lasso regression can be expressed as minimizing the sum of squared errors plus the penalty term:

(4) Min(RSS + λ sum(|coefficient|))

The ridge and lasso penalties can be modelled for the multiple linear regression and are visualised as in composite Figure 4.

Figure 4:



Decision Trees

I start with the set of numeric variables as predictors, and price as the prediction target. I use the number of rooms, the number of bathrooms, how big the lot is, how big the building is, the year it was built and where the lot is as predictors. This model can be described as follows:

(5) Price ~ Rooms + Bathroom + Landsize + BuildingArea + YearBuilt + Latitude + Longitude

This model produces the following simple decision tree in Figure 5.

Max Depth

In decision tree algorithms, the "max depth" parameter refers to the maximum depth or level that the tree can grow to during the training process. It determines the number of splits or branches the tree can have from the root node to the leaf nodes – i.e., it controls the size and complexity of the tree. A higher max depth allows the tree to capture more intricate relationships and interactions between predictors, potentially leading to better training set performance. However, an overly complex model may also be prone to overfitting, where it becomes too specialized to the training data and performs poorly on unseen data, hence it plays a crucial role in managing the trade-off between overfitting and underfitting. Finding the right max depth often involves experimenting with different values and evaluating the model's performance using appropriate evaluation metrics or validation techniques, such as cross-validation.

Our tree is formulated with a maximum depth of 6, since there are 6 nodes from the root node to the lowest leaf, which is also counted as a node. We want to find a balance between

underfitting and overfitting, so I calculate the MAE scores for trees of different maximum depths so we can find the optimum. The results are captured in Table 8.

Figure 5: Decision Tree

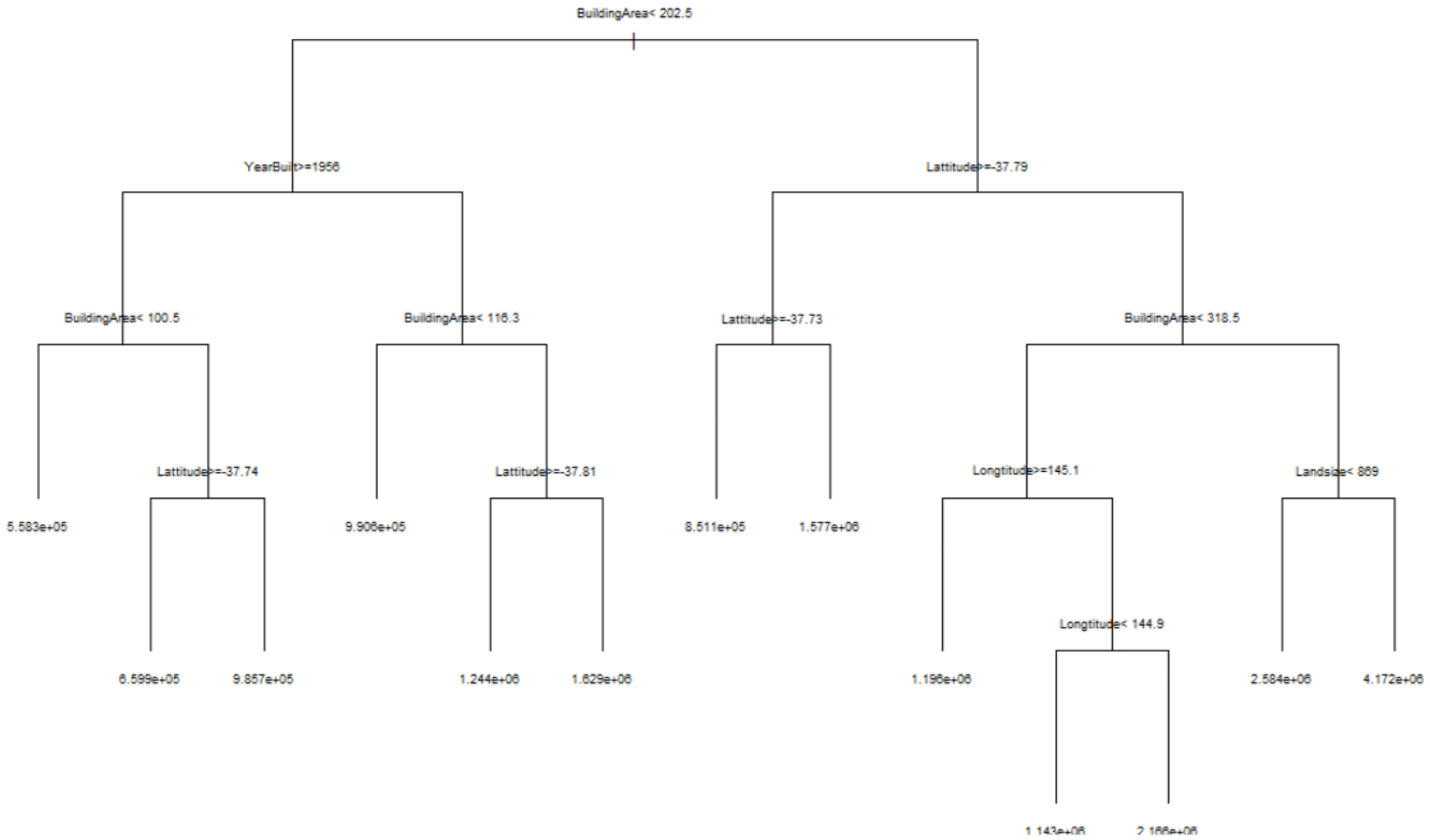


Table 8: Max Depth Assessment

Maximum Depth	Mean Absolute Error
1	412 561.39
2	358 833.59
3	314 415.52
4	302 141.27
5	300 120.25
6	300 120.25
7	300 120.25
8	300 120.25
9	300 120.25
10	300 120.25

The MAE scores converge after a maximum depth of 5 nodes, which has the lowest MAE score, which indicates that our model fitted to the training data, which has 6 nodes, is already fitted rather well. The minimum MAE is 300 120.25, which implies that on average, the model's prediction is off by approximately 300 000 AUS dollars. This is not a very accurate

model. We can try and include more predictors manually to improve the accuracy of the model, or we can create a random forest.

Random Forest

A random forest consists of an ensemble of decision trees. Each tree is built independently using a subset of the training data and a random subset of features. The learner makes a prediction by averaging the predictions of each component tree, and generally has much better predictive accuracy than a single decision tree. Unfortunately, it is quite tricky to produce a visual representation of a random forest. After fitting the random forest model to our training data (that takes the same target and predictors as our preceding individual decision tree), we get the following MAE scores:

Table 9: Random Forest Fit

In-sample fit	Out-of-sample fit
88 938.51	182 424.20

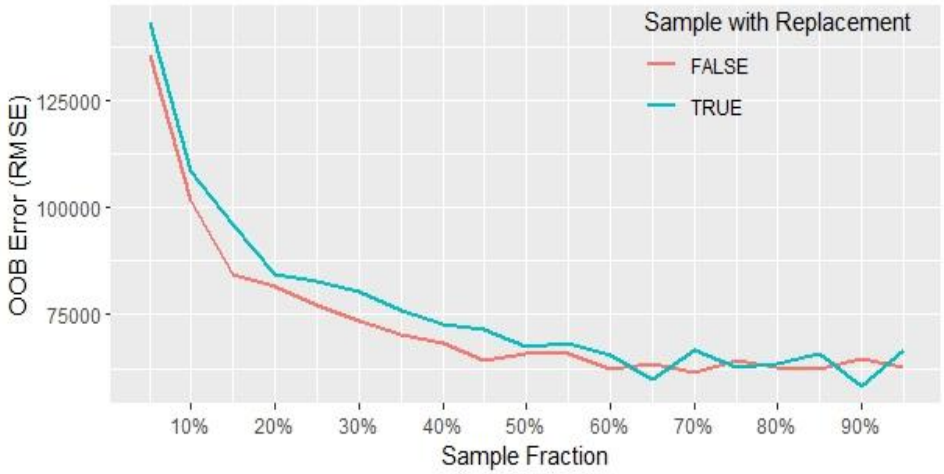
Using a random forest model has dramatically decreased the mean absolute error scores. Now, the model over or underpredicts house prices in Melbourne by approximately 182 000 AUS dollars, as opposed to the previous error of approximately 300 000 AUS dollars. Additionally, the in-sample-fit has greatly improved, and house price predictions are off by less than 100 000 AUS dollars for houses in the training sample. But we can still improve the model's predictive ability even more with hyperparameter tuning and gradient boosting.

Hyperparameter Tuning

Hyperparameters are parameters that are set before the learning process begins and cannot be learned directly from the data. They are parameters that you can adjust to change the nature of the machine learning algorithm, and by extension the bias-variance trade-off. Not all models have hyperparameters, like the OLS regression model, for example. Different hyperparameter values can significantly impact the model's ability to generalize well, avoid overfitting or underfitting, and improve predictive accuracy. There are various methods for hyperparameter tuning, but grid search is commonly used, and is when you allow the computer to search across a range of predefined values and find the best combination. Unfortunately, my computer does not have the processing power to do this successfully. So, while hyperparameter tuning is a crucial part of machine learning processes and improving the accuracy of your model, it could not be completed for this project.

Another way to improve the model's accuracy is to consider different sample size and how to split the sample, and whether to sample with or without replacement or not. Figure 6 below shows when the RMSE is minimised: when there is sampling with replacement at a 65-35 percent split between training and testing data. Alternatively, the RMSE is consistently low from approximately the 60-40 percent split onwards when there is no replacement, so there is no real need to resample our Melbourne housing dataset that is split in a 70-30 ratio by simple random sampling without replacement.

Figure 6: Sampling Schedule



Gradient Boosting and Descent

Gradient boosting is an ensemble learning technique that combines multiple weak prediction models (typically, and in this case, decision trees) to create a strong predictive model. The main idea behind gradient boosting is to iteratively build new models that focus on the errors or residuals of the previous models, thereby reducing the overall prediction error. The models are built in a sequential manner, with each subsequent model trying to correct the mistakes of the previous models. The final prediction is obtained by aggregating the predictions of all the models in the ensemble. The process typically involves specifying the boosting parameters and fitting the ensemble on the training data.

Gradient descent is an optimization algorithm used to minimize an objective function by iteratively updating the model parameters in the direction of steepest descent. It is commonly used in training machine learning models to find the optimal set of weights or coefficients that minimize the difference between predicted and actual values. The algorithm calculates the gradient of the objective function with respect to the model parameters and updates the parameters iteratively by taking steps proportional to the negative gradient. There are different variants of gradient descent, such as batch gradient descent, stochastic gradient descent (SGD), and mini-batch gradient descent, each with its own characteristics and trade-offs.

Running a basic GBM model results in an RMSE of 332 333.70. The XGBoost requires some additional data preparation, so I encode the categorical variables numerically, and go through a series of grid searches to find the model hyperparameters. The minimum test CV RMSE is 320 935.30 for the GXBoost model for 6000 iterations, which is a slight improvement on the basic GBM model.

I assess whether overfitting is limiting the model's performance by performing a grid search that examines various regularisation parameters for 4000 iterations. The results are captured in Table 10.

Table 10: GXBoost CV 10-folds

Iteration	Train RMSE Mean	Train RMSE S.D.	Test RMSE Mean	Test RMSE S.D.
1	1 264 647.7	7 026.255	1 263 170.4	61 672.95
2	1 263 194.7	7 015.882	1 261 714.5	61 708.83
3	1 261 728.7	6 999.672	1 260 245.1	61 750.21
4	1 260 263.3	7 012.259	1 258 776.9	61 764.71
5	1 258 811.1	7 003.415	1 257 321.7	61 799.09
…	…	…	…	…
3996	479 595.7	7 733.994	477 492.7	74 564.42
3997	479 566.8	7 734.855	477 463.9	74 565.69
3998	479 539.7	7 733.956	477 437.8	74 566.77
3999	479 512.6	7 734.814	477 410.9	74 566.88
4000	479 485.6	7 735.126	47 7384.7	74 567.07
Best Iteration				
4000	479 485.6	7 735.126	477 384.7	74 567.07

I create the final GXBoost model with 7 features and 4000 iterations and get the results in table 11.

Table 11: GXBoost Final Model

Iterations	**Train RMSE**
1	1 255 688.7
2	1 245 734.0
…	
3999	230 119.1
4000	230 114.1

The final GXBoost model has a significantly lower in-sample RMSE than its preceding versions. However, the GXBoost model does not have better predictive accuracy than the random forest model, that has an in-sample fit of 88 938.51, and an out of sample fit of 182 424.20. More hyperparameter tuning is required to further improve the GXBoost model, as hyperparameter tuning often takes on an iterative process where different parameter values are tweaked and tried.

Conclusion

There are various machine learning models that we can use to predict housing prices in Melbourne, Australia. The linear regression models are simple and easy to interpret, but have a rather high RMSE. The decision tree has similar predictive capabilities as the regression models, but the random forest significantly improves in predictive accuracy. When used to predict prices in the sample, it is only off by approximately 80 000 AUS dollars on average, while price predictions on new data is off by approximately 180 000 AUS dollars. The model could be made more accurate with hyperparameter tuning and boosting methods, although this becomes rather complex and computationally intensive, and could not successfully improve

upon the random forest's predictive accuracy. The most accurate machine learning model for predicting house prices in Melbourne, given my dataset, is the random forest model.

Appendix

The descriptions for the column names of the Melbourne housing dataset are as follows:

Suburb: Suburb

Address: Address

Rooms: Number of rooms

Price: Price in Australian dollars

Method:
S - property sold;
SP - property sold prior;
PI - property passed in;
PN - sold prior not disclosed;
SN - sold not disclosed;
NB - no bid;
VB - vendor bid;
W - withdrawn prior to auction;
SA - sold after auction;
SS - sold after auction price not disclosed.
N/A - price or highest bid not available.

Type:
br - bedroom(s);
h - house, cottage, villa, semi, terrace;
u - unit, duplex;
t - townhouse;
dev site - development site;
o res - other residential.

SellerG: Real Estate Agent

Date: Date sold

Distance: Distance from CBD in Kilometres

Regionname: General Region (West, North West, North, North east …etc)

Propertycount: Number of properties that exist in the suburb.

Bedroom2 : Scraped # of Bedrooms (from different source)

Bathroom: Number of Bathrooms

Car: Number of car spots

Landsize: Land Size in Metres

BuildingArea: Building Size in Metres

YearBuilt: Year the house was built

CouncilArea: Governing council for the area

Lattitude: latitude of the property

Longtitude: longitude of the property

References

Bergstra, J., & Bengio, Y. (2012). *Random search for hyper-parameter optimization*. Journal of Machine Learning Research, 13(Feb), 281-305. Available at: https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf?source=post_page----------

Boehmke, B. and Greenwell, B.M., 2019. *Hands-on machine learning with R*. CRC press. Available at: https://bradleyboehmke.github.io/HOML/

Breiman, L. (2001). *Random forests. Machine Learning*, 45(1), 5-32. Available at: https://link.springer.com/content/pdf/10.1023/A:1010933404324.pdf

Friedman, J. H. (2001). *Greedy function approximation: A gradient boosting machine.* Annals of Statistics, 29(5), 1189-1232. Available at: https://www.jstor.org/stable/2699986

McDermott, G. (2021). *Data Science for Economists and Other Animals*. Available at: https://grantmcdermott.com/ds4e/index.html

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer. Available at: https://d1wqtxts1xzle7.cloudfront.net/31156736/10.1.1.158.8831.pdf?1366444917=&response-content-disposition=inline%3B+filename%3DThe_elements_of_statistical_learning_dat.pdf&Expires=1687574893&Signature=VjwFYSKKlISB8o3nnFCqU~9WwtGjOFJCJhDSa--Umcy6-~LwjNO24ABVsG0xDFfO1opZHPVXyoF1ejKBWtK3CpL3vgTggi3Jg-VxFdvVEAGzWjCtlrkhrTh65d~0rfcLWMSVF6B~UiMb6S6anIJQROk8qF1uu101jk0-HIL~p2dln8YDI9nFfFhXpNWhZCNy5pQ2o8rl0OrnO7PS6VrX0ut7qMK2AjMnJ~z7R5okOBhL98szgm6ouhNaV0jNaEIl~hN5qzEilhIxpPqcsxlaVhA1iC6YZdvm5jSmMkqnv~On6goD6kIspblBT0DsQG6Ysg0GSaRi9Nz2NJ3FBmm4Pg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning*. Springer. Available at: https://www.academia.edu/download/60707896/An_Introduction_to_Statistical_Learning_with_Applications_in_R-Springer_201320190925-63943-2cqzhk.pdf

Tibshirani, R. (1996). *Regression shrinkage and selection via the lasso*. Journal of the Royal Statistical Society: Series B (Methodological), 58(1), 267-288. https://cir.nii.ac.jp/crid/1370004236282599299