Fundamentals of Data Science and Research Computing Combined Project- Melbourne Housing Snapshot

December 18, 2025

1 Motivation

The Melbourne Housing Snapshot dataset is used to predict property prices in Melbourne from 2016–2017. Asset price prediction is at the intersection of Economics and Data Science and there is substantial heterogeneity of house prices in large cities. I was especially interested in investigating how spatial factors impact house prices in cities since similar sized homes with comparable interiors vastly differ in price depending on the neighbourhood and proximity to the Central Business District (CBD).

The GLM (Generalised Linear Model) and LGBM (Light Gradient Boosting Machine) models are tuned with k-fold cross validation (CV) and trained. I find that the LGBM model outperforms the GLM model with lower relative MAE, RMSE and bias. This is because the LGBM model is more flexible in accounting for non-linear effects of features on the target.

2 Explanatory Data Analysis and Cleaning

Features are decomposed into the following categories. Note, feature names differ to those used in code for ease of explanation.

Feature Category	**Features**
Target	*prices* (AUD)
Time	*day*, *month* and *year* of sale
Locational	*distance* from the CBD (km), *latitude*, *longitude*, *council*, *postcode*, *region*, *suburb*, *propertycount*
Size	*buildingarea* (m2), *bathrooms*, *bedrooms*, *rooms*, *car* spots, *landsize* (m2)
Other	*method* of sale, *type* of home, *yearbuilt*

Table 1: Feature categories and associated variables

2.1 Target Variable

prices have a strong right skew which can be seen through the boxplot and distribution plots in Figure 1 which is due to the infrequent sale of luxury properties leading to several outliers above the upper whisker of the boxplot. To minimise the skew, *prices* were logged to obtain a more symmetrical distribution so that residuals follow the normal distribution more closely. Figure 1 shows the boxplot of $log(prices)$ and distribution of $log(prices)$ fitted against the normal, burr and fisk (log-logistic) distributions. Although the burr and fisk distributions also fit $log(prices)$ relatively well, the normal distribution fits with lowest error. Hence, I selected the gaussian GLM with identity link on $log(prices)$ which corresponds to minimising the Mean Squared Error (MSE) on $log(prices)$.

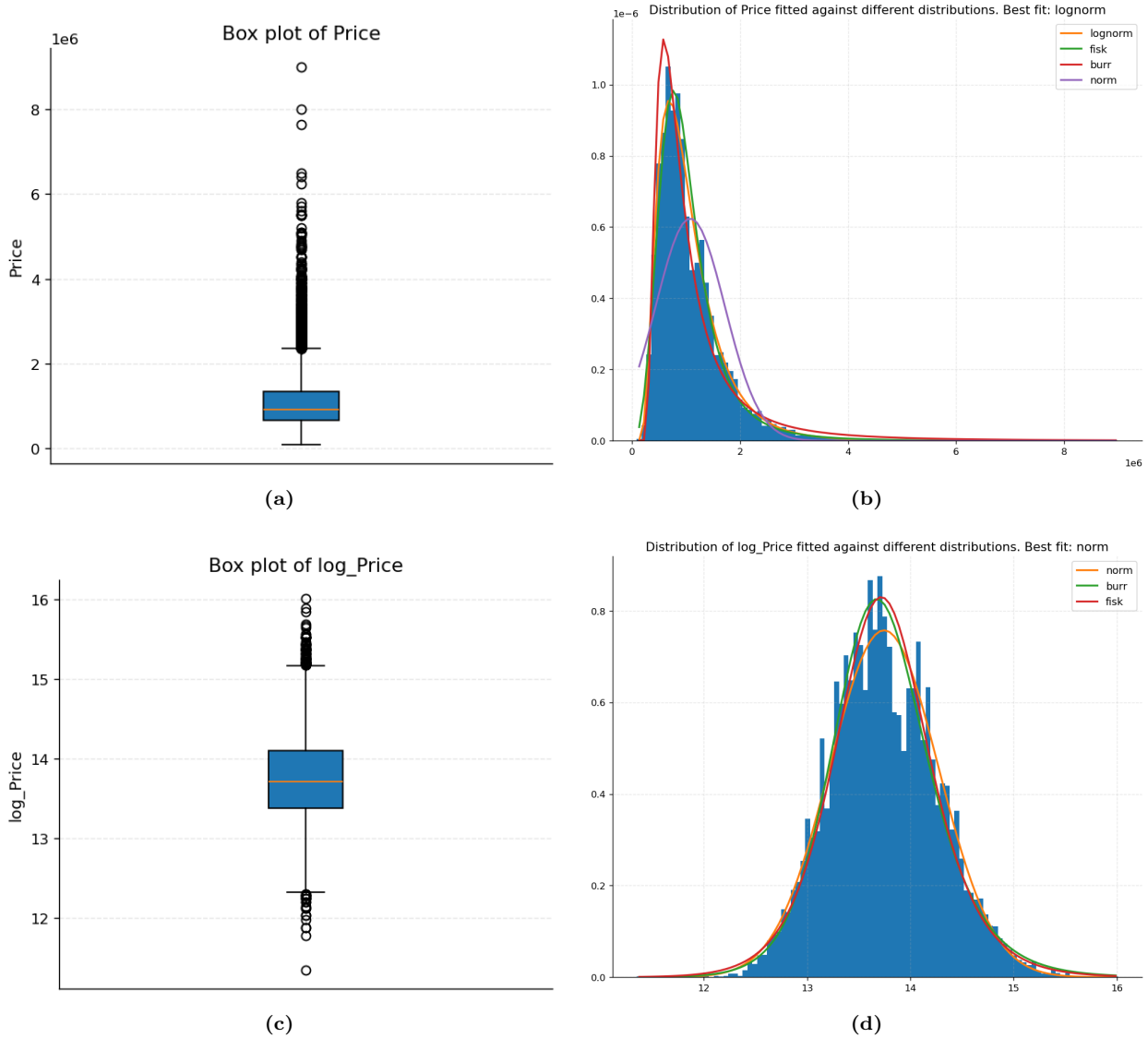


Figure 1

2.2 Time Features

day and *month* were dropped since $log(prices)$ has a similar distribution across days and months. Although the distribution of $log(prices)$ by *year* is also indistinguishable, *year* is not dropped as it is the only feature which encodes annual macroeconomic factors such as consumer confidence which housing markets are sensitive to.

2.3 Locational Features

2.3.1 Council

The dataset contains 10.1% missing *council* data which was imputed. In Melbourne, since several suburbs are a subset of one *council*, I mapped each *suburb* to its respective *council* through observations with no missing *council* data. I used this mapping to impute missing *council* data depending on the *suburb* which the observation fell into.

2.3.2 Council, Postcode, Region, Suburb and Property Count

As shown in Cramer's V correlation matrix (Figure 2), there is a strong multicollinearity between categorical locational features. Including all features in the model is likely to cause overfitting on training data. Although regularisation was conducted to minimise overfitting, *council*, *propertycount* and *postcode* were dropped before modelling since they largely overlap with *suburb*:

- multiple suburbs are a subset of a single *council* and *council* is less granular than *suburb*.
- the number of properties in a *suburb* is identical for all properties in a given *suburb*.
- *postcode* boundaries generally align with *suburb* boundaries and is less granular than *suburb*.

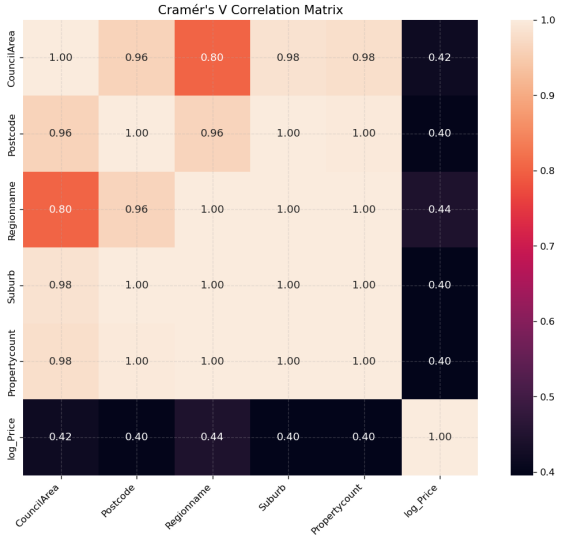


Figure 2

2.4 Size Features

- *buildingarea* was dropped since it faces substantial missingness with 47.5% of observations containing missing data.
- The 0.5% missing data for *car* was imputed with median imputation in feature engineering. Median was chosen as the imputation strategy rather than mean due to the right skew of the feature.

2.4.1 Rooms

The low accuracy of *rooms* and availability of a close alternative (*bedrooms*) means it was dropped. The interpretation of *rooms* is unclear and not provided in the dataset. Since *bedrooms* are a subset of *rooms*, it must be the case that $bedrooms \leq rooms$ in a property. However, Figure 3 shows that there are observations where $bedrooms > rooms$ which is not possible and is likely a data entry error.

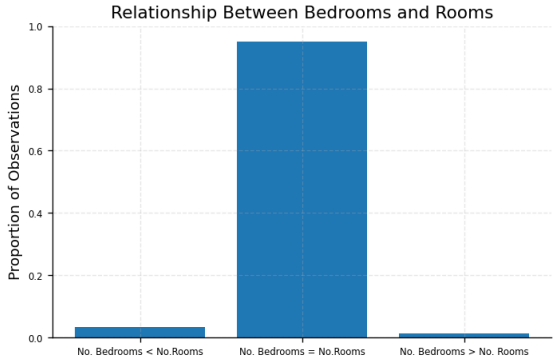
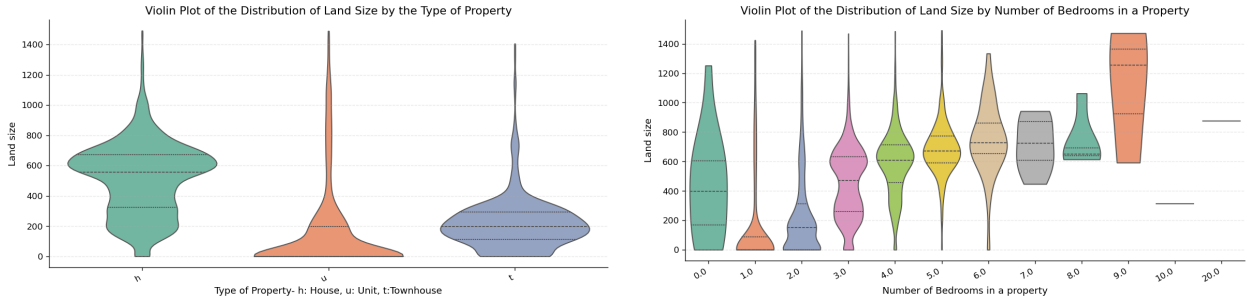


Figure 3

2.4.2 Land size

Landsize was winsorized on the upper tail since there are several outliers. For example, 6 observations have $landsize > 40,000\ m^2$ which is likely to be a data entry error and distorts the distribution significantly since 99% of properties have $landsize \leq 1500\ m^2$. A significant proportion of properties have $landsize = 0$ which under further analysis was concluded not to be missing data, hence *landsize* was not winsorized on the lower tail as it could represent structural differences in $log(prices)$.

$Landsize = 0$ means that property ownership does not include privately owned land. Figure 4a illustrates that units are most likely to exhibit $landsize = 0$. These are apartments where ownership of an apartment does not entail ownership of the land it is built on. Other reasons for no land ownership could be shared property ownership or government restrictions on land ownership. A binary indicator which flags where $landsize = 0$ was be added to represent houses with no land ownership. Figure 4b shows that $landsize > 0$ remains a good measure of house size because median *landsize* rises with the *bedrooms*.



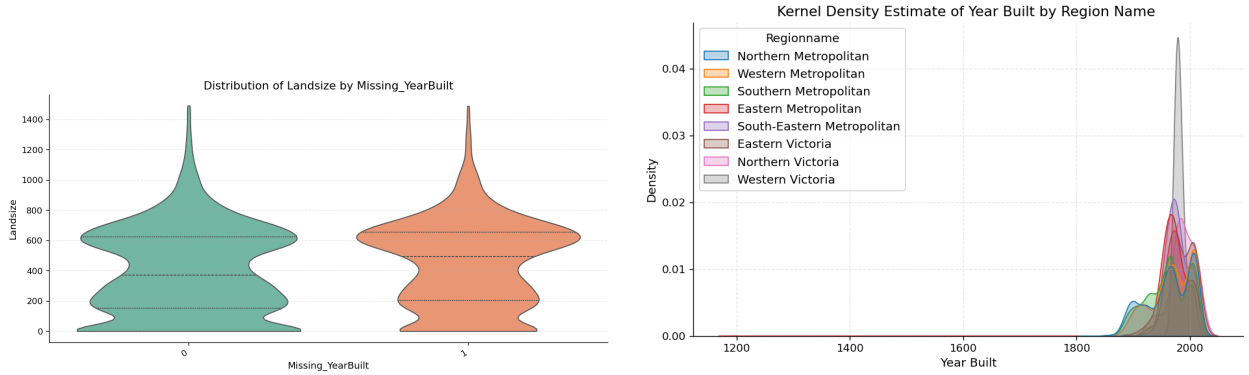
(a) **(b)**

Figure 4: Note, $Landsize \leq 1500$ to visualise plots clearly

2.5 Other Features

Although 39.6% of *yearbuilt* data is missing, it encodes factors such as condition, layout and architecture. Simple median imputation raises two issues. Firstly, leads to 39.6% of observations having the same *yearbuilt*. Secondly, Figure 5a shows that houses with missing data for *yearbuilt* have a systematically different distribution of $log(prices)$ with a higher median $log(prices)$. Intuitively, it could be that houses with missing *yearbuilt* data are older with larger *landsizes* and higher $log(prices)$.

Properties are likely to be constructed in phases with properties in nearby areas built at similar times. Hence, conditional median imputation on *region* was chosen. *region* was chosen as the locational feature since other features are too granular and have missing *yearbuilt* for all observations in a given area. Figure 5b shows that the distribution of *yearbuilt* is indeed different across regions.



(a) **(b)**

Figure 5

3 Feature engineering

The median imputation of *car*, upper tail winsorization of *landsize*, median imputation of *yearbuilt* conditional on *region* were discussed above. Other feature engineering steps are discussed in this section.

3.1 Spline Transformation

It can be seen from the spatial distribution plot of *log*(*prices*) by *longitude* and *latitude* (Figure 6) and scatter plots in Figure 7 that *longitude* and *latitude* observe a non-linear spatial effect on *log*(*prices*). Together, they lead to spatial clusters forming with the highest *log*(*prices*) in Central and South-eastern Melbourne. Hence, a spline transformation was applied on features with 5 knots chosen to avoid overfitting. Knots were chosen at quintiles rather than uniformly since housing density is not evenly distributed across the features, allowing knots to be flexibly placed in areas where there are most observations.

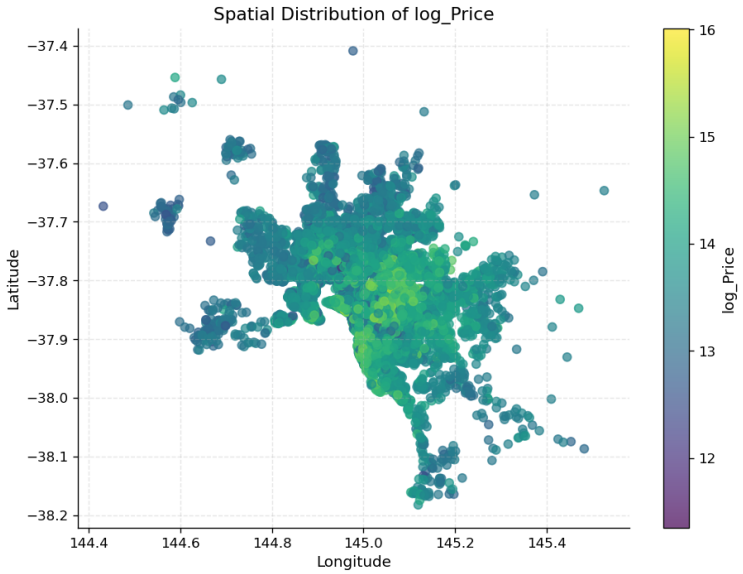
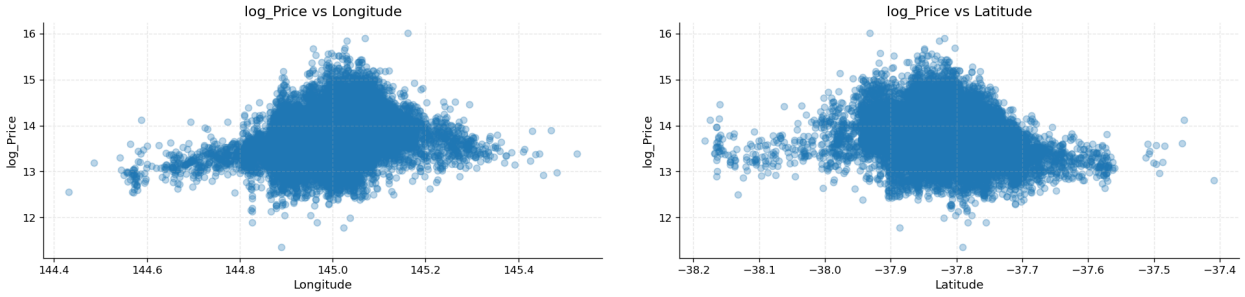


Figure 6



(a) **(b)**

Figure 7

3.2 Target encoding

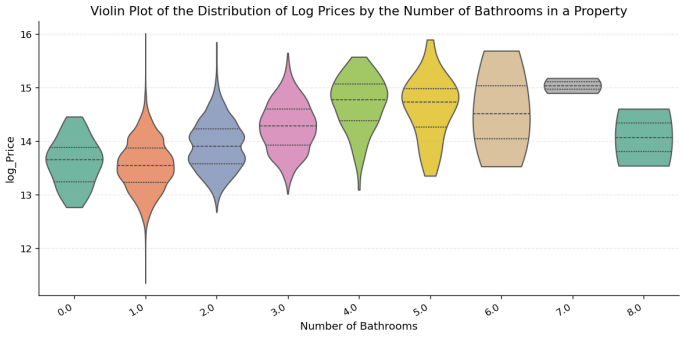
suburb is a highly dimensional categorical feature with 314 unique suburbs in the dataset. Hence, the target encoding transformation was applied rather than one hot encoding to avoid creating 314 new columns in the final dataset, slowing down the model. Target encoding creates a single column for *suburb* which is mean

log(*prices*) conditional on *suburb*, fitted on training data. Target encoding can improve model performance relative to one hot encoding since high dimensional categories get ordered by their association with the target variable while other encodings do not contain this information. Smoothing controls the extent to which conditional means are brought closer to the global mean *log*(*prices*). Smoothing = 10 was chosen since suburbs have an unequal number of observations which increases the variation and noise in conditional means. Hence, smoothing reduces variability to avoid overfitting whilst not excessively smoothing to create bias.

3.3 Simple Imputer- Median imputation

Instances where *bedrooms* = 0 or *bathrooms* = 0 are not possible and are likely data entry errors or missing data. The violin plots (Figure 8) show that the distribution of *log*(*prices*) for *bedrooms* = 0 and *bathrooms* = 0 do not fit the trend where a higher number of rooms leads to higher median *log*(*prices*) and a shift upwards in the violin plot. Hence, I treated the cases where *bedrooms* = 0 or *bathrooms* = 0 as missing data and used median imputation due to the right skew of these features.

(a)



(b)

Figure 8

3.4 Standard Scaler

The Standard Scalar transformation was applied to *landsize*, *longitude*, *latitude* and *yearbuilt* to ensure that features with large absolute variances do not dominate when modelling. Although numeric, *bedrooms*, *bathrooms* and *car* spots were not scaled since they are discrete with low variances and scaling reduces

interpretability of the features.

3.5 One Hot Encoding

One Hot Encoding of all categorical features apart from *suburb* was implemented.

4 Hyperparameter Tuning

I used *RandomizedSearchCV* rather than *GridSearchCV* for hyperparameter tuning. *GridSearchCV* iterates over every combination in the hyperparameter grid, each k times (where k is the number of CV folds), increasing computational time. Instead, *RandomizedSearchCV* tests a fixed number of random combinations and is more efficient. Although this means results on the best pipeline can vary each time it is run, by setting $random\_state = 42$, I controlled for randomness so that results are reproducible.

I used 5 folds in CV which means under each hyperparameter combination tested, the training dataset was split into 5 folds and the given model was trained on 4 folds before being validated on the last fold. The process was repeated 5 times, each on new validation folds and an average model accuracy was recorded. This helps reach a robust conclusion on the best choice of hyperparameters.

4.1 GLM Tuning

Parameters tuned	**Values tuned over**	**Optimal Value**	**Explanation**
α	$0.001, 0.01, 0.1, 1,$ $10, 100, 1000, 10000$	0.001	α controls the overall penalty strength by determining how much coefficients are shrunk by. I obtained a low optimal α which suggests that the optimal model is close to the unregularised GLM. Since I previously dropped features causing multicollinearity, the model may not be not overfitted.
$l1\_ratio$	$0.0, 0.25, 0.5, 0.75, 1.0$	0.0	This determines if a pure *Lasso* ($l1\_ratio = 1$), pure *Ridge* ($l1\_ratio = 0$) or an elastic net ($0 < l1\_ratio < 1$) regularisation is optimal. *Ridge* shrinks coefficients smoothly whereas *Lasso* selects feature coefficients to set to zero to reduce overfitting. Since *Ridge* was selected, it means that setting a particular feature coefficient to zero does not increase accuracy, implying that all selected features are correlated and important in prediction.

Table 2: Regularisation prevents overfitting the model due to multicollinearity

4.2 LGBM Tuning

Parameters tuned	**Values tuned over**	**Optimal value**	**Explanation**
learning_rate	$0.01, 0.02, 0.03,$ $0.04, 0.05, 0.1, 0.2$	0.05	This determines by how much each new tree changes the final prediction and contributes to the model.
n_estimators	5000	5000	I set *n_estimators* high so that I could use *early_stopping* = 25 which allows the model to stop early if validation performance does not improve for 25 trees. *n_estimators* is the maximum number of trees (upper bound in the case of *early_stopping*) an LGBM model can take.
num_leaves	$15, 30, 60, 120$	30	This controls the complexity of each tree by selecting the maximum number of leaves in a tree. Increasing *num_leaves* improves the depth of trees to increase model flexibility but risks overfitting.
min_child_weight	$0.1, 1, 2, 5$	0.1	This controls the minimum number of observations needed for a child leaf to split.

Table 3: Tuned hyperparameters suggest that in the optimal LGBM pipeline, each tree is small with few leaves, but each tree can substantially contribute to the model via a high model.

5 GLM and LGBM Results

Metrics: target variable – $log(prices)$	GLM	LGBM
Relative MAE (MAE / True Mean)	0.0139	0.0100
Relative RMSE (RMSE / True Mean)	0.0188	0.0138
Bias	-0.0003	-0.0002

Table 4: Model performance metrics with $log(prices)$ as the target variable

Metrics: target variable – *prices*	GLM	LGBM
True Mean (AUD)	1,081,438	1,081,438
Mean Prediction (AUD)	1,035,288	1,057,941
Relative MAE (MAE / True Mean)	0.202	0.149
Relative RMSE (RMSE / True Mean)	0.385	0.290
Bias	-0.0427	-0.0217

Table 5: Estimated transformed model performance metrics with *prices* as the target variable

Since models were trained on $log(prices)$, the original $log(prices)$ metrics such as relative MAE and RMSE are multiplicative errors and are difficult to interpret beyond the fact that the LGBM model performs better than the GLM model with lower absolute and relative MAE and RMSE as well as a lower bias. This is expected since the LGBM model is more flexible in accounting for non-linear effects of features on the target.

To interpret results further, I transformed $log(prices)$ to *prices* for predicted and true $log(prices)$. Although this leads to a slight underprediction since the exponential function is non-linear (in line with Jenson's inequality), it allows for evaluation on the original price scale. The GLM model achieves a relative MAE of 20.2% whereas the LGBM model performs better with a 14.9% relative MAE. This means, on average the LGBM predicts price estimates to be 14.9% away from true *prices*.

Although the 14.9% relative MAE means that LGBM model may not be suitable for exact price predictions, it can be used by estate agencies to predict an initial price range based on information available online before viewing the property to evaluate the exact listing price. Estate agencies often use house price range predictions as marketing tools to encourage homeowners to realise their home value and list it to sell.

6 Evaluation

Figure 9 plots predicted and actual $log(prices)$ for the testing dataset, depicting the stronger predictive accuracy of the LGBM model as points are closer to the 45-degree line. It also shows that performance predictions worsen for very expensive homes as there are a larger heterogeneity of features in this range making it more difficult for models to learn. *Prices* for these homes could be driven by factors which are also not captured in this dataset such as architectural design.

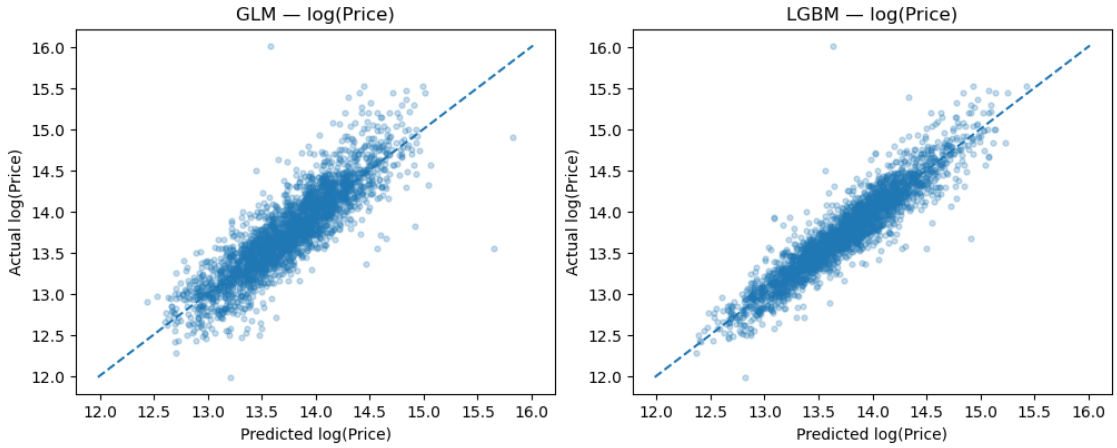


Figure 9

The most relevant features in the LGBM model are shown in Figure 10 and Partial Dependency Plots (PDP) of these features are presented in Figure 11. The PDP of *suburb* is excluded since it is a highly dimensional categorical feature and the PDP provides little interpretability.

The PDP shows the average marginal effect of a feature on prediction whilst holding other features fixed. However, we must consider that the PDP assumes independence between features and since features are correlated, the PDP is biased. In general, it can be observed that the LGBM model is more flexible and considers non-linear effects of a feature explaining its superior performance. Interesting observations from the PDPs are:

- *landsize*: whether $landsize = 0$ or not is a more significant predictor of $log(prices)$ than the absolute value of *landsize*.
- *distance*: as *distance* to the CBD rises, it initially leads to a steep fall in $log(prices)$ but this effect flattens as we reach the commuter zone which is consistent with the concentric zone model in urban economics. The opposite effect is true for *bedrooms*.

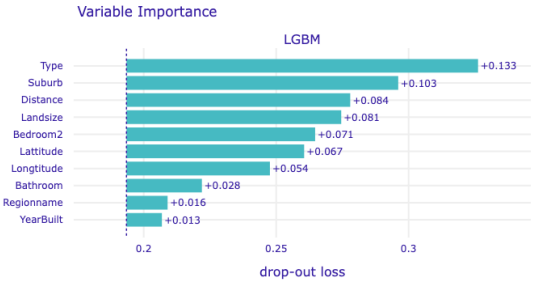
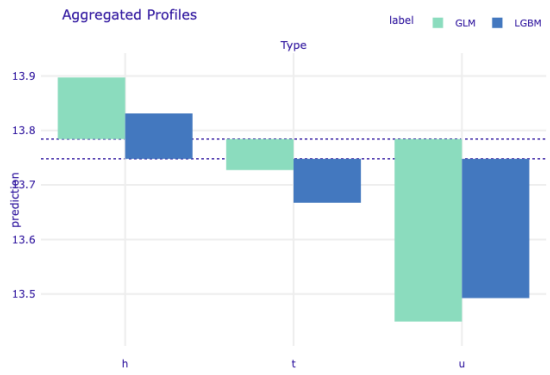
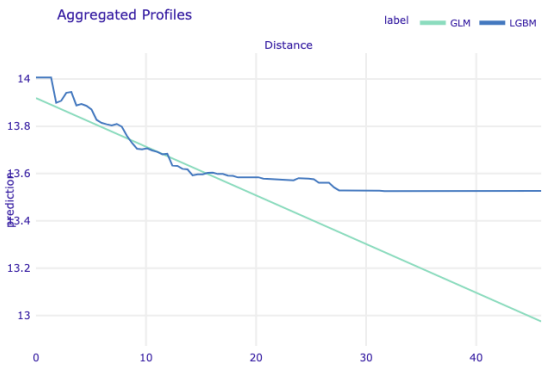


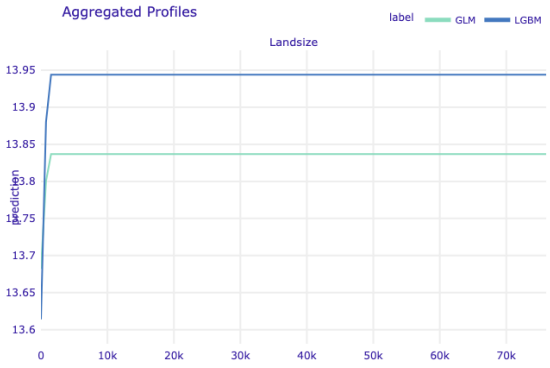
Figure 10



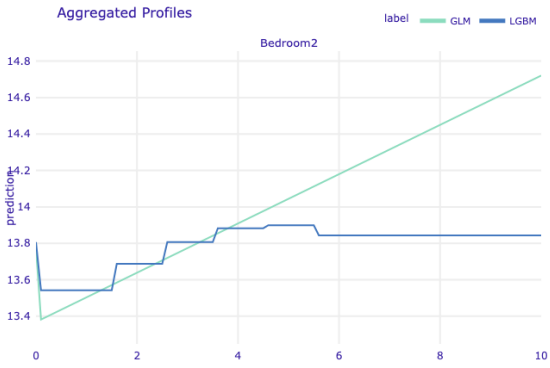
(a) Type of home- h: house, t:townhouse, u: unit



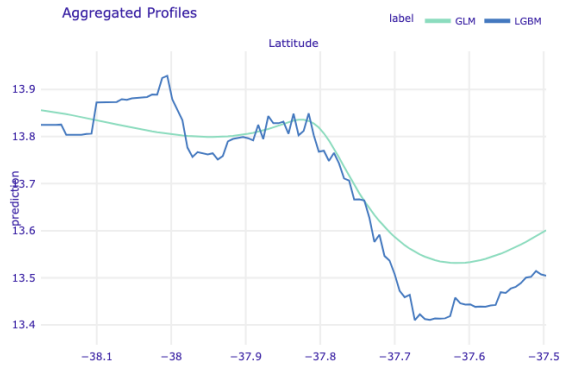
(b) Distance from CBD (km)



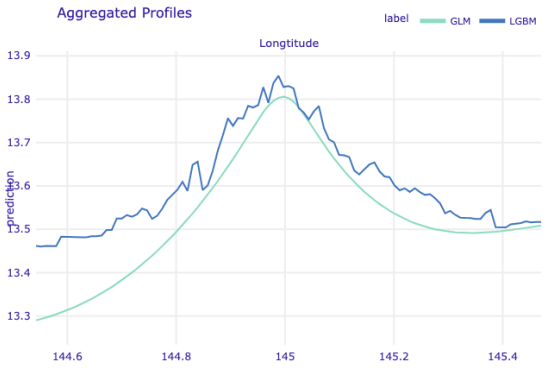
(c) Land size (m^2)



(d) Number of Bedrooms



(e) Latitude



(f) Longitude

Figure 11: Partial Dependency Plots of $log(prices)$ on important features

7 Improvements

Although the LGBM model has 14.9% predictive error on *prices* based on relative MAE, there are improvements which I would make given the opportunity for further analysis:

1. Use *longitude* and *latitude* to extract the proximity to coastline, best schools and public transport facilities. This is possible with the given dataset and these locational factors are historically important predictors of *prices*.
2. Evaluate Accumulated Local Effects plots to overcome the bias in PDP.
3. Web scrape the estate agency listing the property which affects marketing and staging of homes.
4. Use a more robust feature selection method such as forwards stepwise or lasso selection to drop correlated features.

Although *longitude* and *latitude* mean I can add several locational features to the model, the dataset lacks housing specific features such as housing condition and room sizes which are significant price predictors.

References

[1] Dan Becker. Melbourne housing snapshot. https://www.kaggle.com/datasets/dansbecker/melbourne-housing-snapshot/data, 2018. Kaggle dataset.

[2] DALEX developers. Dalex: Python api. https://dalex.drwhy.ai/python/api/, 2023. DALEX documentation.

[3] Post Gourang. A comprehensive guide to loss functions in machine learning and deep learning. https://medium.com/@post.gourang/a-comprehensive-guide-to-loss-functions-in-machine-learning-and-deep-learning-6387db15668f, n.d. Medium article.

[4] scikit-learn developers. sklearn.preprocessing api reference. https://scikit-learn.org/stable/api/sklearn.preprocessing.html, 2024. scikit-learn documentation (stable).

[5] scikit-learn developers. sklearn.preprocessing.binarizer. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Binarizer.html, 2024. scikit-learn documentation (stable).