London Business School

House Price Prediction and Investment Strategies: A Data Science Approach

Khushi Jaiswal

Abstract **1**
Introduction **2**
Understanding and Cleaning the Data **2**
Cleaning 2
Patterns 3
Interaction Variables 4
Model Training & Analysis **4**
Linear Regression 4
Decision Tree 5
K-Nearest Neighbors (KNN) 5
Random Forest 6
Stacking Ensemble 6
K Fold Cross Validation for Evaluation 6
Challenges Faced 7
Finding the Best Investments **7**
Extensions and Future Ideas **7**
Adding More Data 7
Trying New Models 7
Conclusions **8**

Abstract

In this report, I will explore how to predict house prices and identify the best investments in the housing market. I used historical data, tuned parameters of several prediction models, analysed each model's performance and then chose the one that worked best. Finally, I used the best performing model to figure out which houses might be good investments. Models include linear regression, tree, K nearest neighbours and random forest.

We will end up with a model that will be able to accurately select 200 most profitable future investments from a much larger sample of almost 2000 data points. The model itself will be

trained on a set with nearly 14000 data points and 18 variables. In the end, we will have a list of the top 200 houses that could bring the most profit.

This technical report provides simple, clear steps for anyone who wants to use, understand or create similar models for prediction. But more importantly, the results that are finally obtained will give a good prediction into future investment that can be made.

Introduction

House prices are tricky to predict because they depend on so many things: where the house is, how big it is, transportation facilities near it etc.. This project will aim to take all those factors and use them to estimate how much a house should cost?

I have used data from real houses and built models to predict house prices. These models function as frameworks, processing input data in a structured manner to generate a specific output (price in our case). Here's the plan this project followed:

1. **Explore the data**: Clean up messy parts and look for patterns
2. **Build models**: Test different ways to predict prices.
3. **Check the results**: See how good the predictions are.
4. **Find the best investments**: Use our predictions to identify the best deals in the market.

This report will walk you through everything I did, explaining it in a simple and easy-to-follow way.

Understanding and Cleaning the Data

Cleaning

First, I looked at the data I had - it included but was not limited to:

- **House size**: How big is the house in square feet?
- **Number of bedrooms**: How many rooms does it have?
- **Location**: Which district is the house in?
- **Neighborhood details**: Distance from nearest train lines ?
- **House type** : Flat? detached? Terrace? etc.

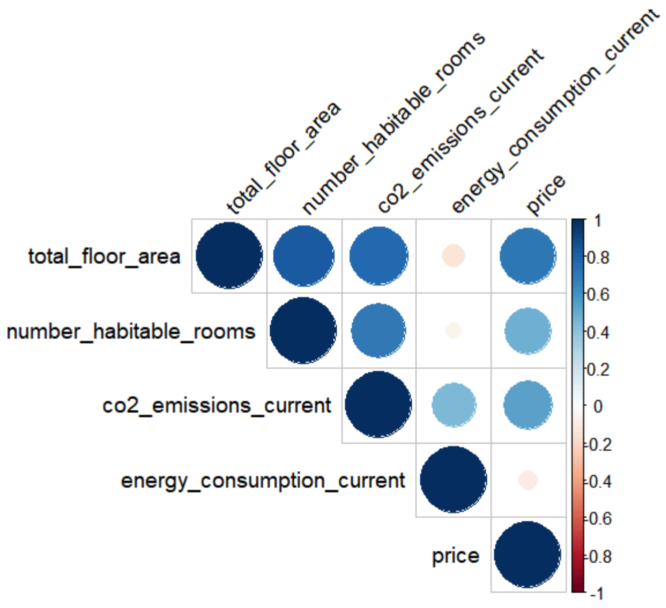
Full house information can be found in the datasets provided with this report

Some of the data was messy. For example, some houses didn't have values for certain details, like the current energy efficiency. I cleaned this by filling in missing values and making sure all the data was ready for analysis.

As is usually done to train ML models, I split the initial dataset into 2 sets; training (75%) and testing (25%). This way we will be able to use training data to train the model and then test the trained model on unseen data using the remaining 25% of the data.

Patterns

It is also important to understand how all the given data links with each other. This will help us identify most significant variables and interaction variables that we can later make use of to train the models.



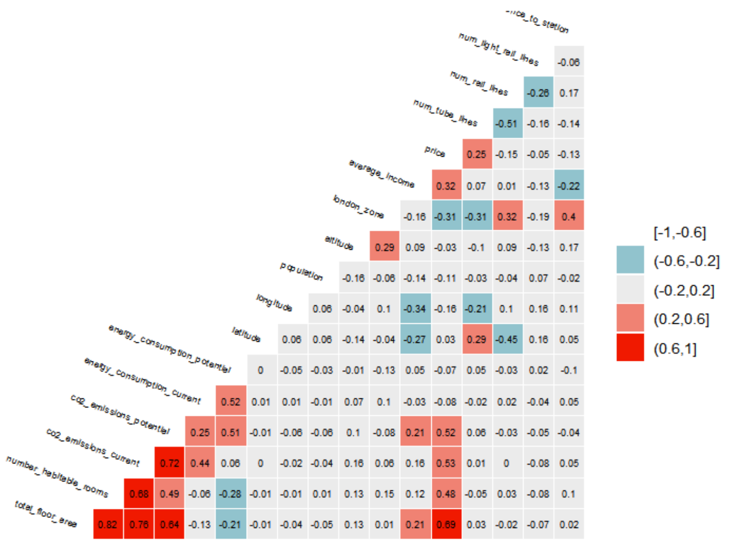
◂ *Correlation heat map shows us that price is more affected by factors like floor area, and number of habitable rooms and less by factors such as current energy consumption.*

▲*Figure 1.1 - Initial correlation Matrix*

Observations: ▸

Variables with the most significant impact on property prices include average income, zone, CO2 emission levels (current and potential), floor area, and the number of rooms. These should be prioritized for predictive modeling.

Variables with weak correlations to price such as potential energy consumption are less impactful and can be excluded from the predictive models.

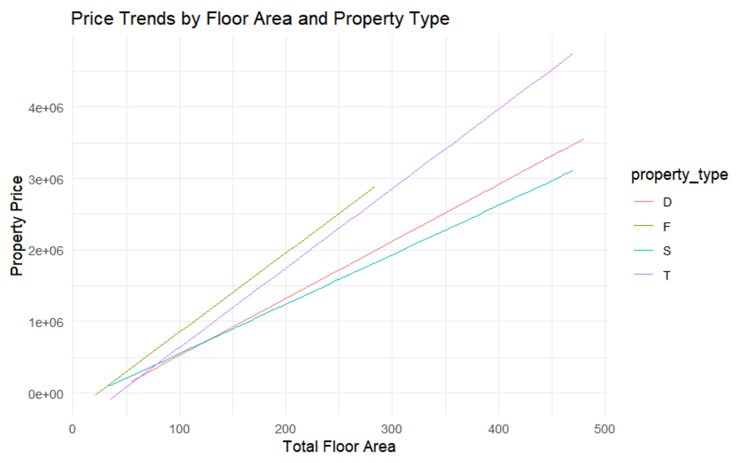


▲*Figure 1.2 - Full Correlation table*

Interaction Variables

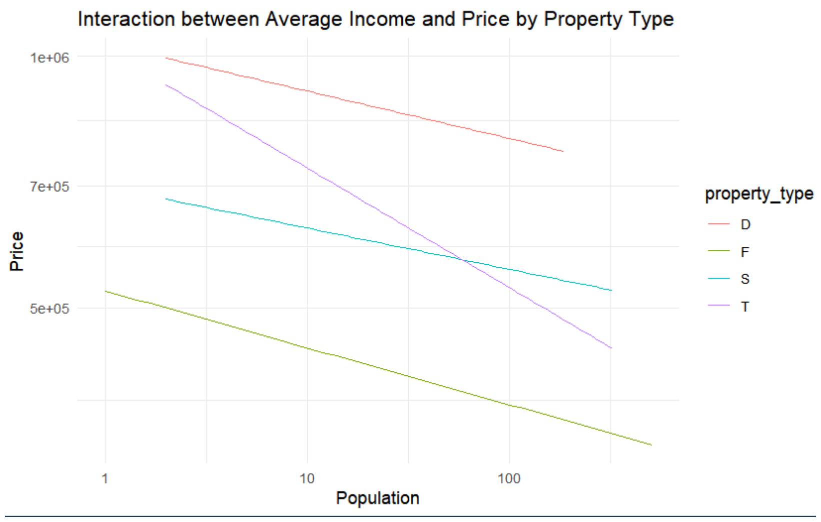
Interaction variables represent the combined effect of two or more independent variables on a dependent variable. In statistical modeling, interactions occur when the relationship between one independent variable and the dependent variable depends on the value of another independent variable.

E.g. The relation between area of a house (independent variable 1) and the price (dependent variable) might be affected by the property type (independent variable 2).



◂ Looking at this line graph, we see the lines are not parallel - they are incepting (though at similar gradients). This indicates that property type and floor area are interaction variables and will help us improve our graph.

▲ Figure 1.3 - Interacting Variables (Total Floor Area & Property Type)



◂ Similarly to Figure 1.3, the intercepting of the line plot indicates that the population and property type are interacting variables.

▲ Figure 1.4 - Interacting Variables (Population & Property Type)

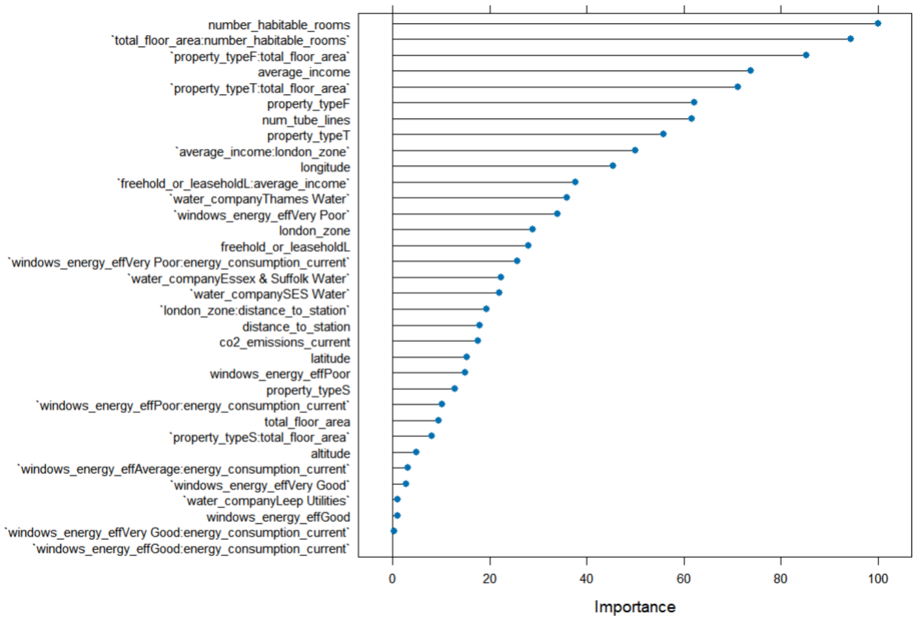
This initial data analysis will greatly help us in the the next part - training models

Model Training & Analysis

Linear Regression

I started with the simplest model: linear regression. This model assumes a straight-line relationship between features like house size and price. Initially, I used a basic version of linear regression, which only considered simple features like the size of the house and the number of bedrooms - this would work well on small dataset but not on one as big as ours.

To improve it, I added interaction terms like combining location and size. Adding these terms allowed us to capture some of the more subtle patterns in the data. Figure 2.1 shows how much of the variation in the graph is captured by a given variable that we add to the model.



◀ The importance of each variable can be seen in this graph.

I kept removing less significant variables and adding more significant ones until I was happy with the models RMSE and R squared returned by the models.

▲*Figure 2.1 - Variable Importance*

* RMSE - Measures the average error between predicted and actual values in a regression model. Lower RMSE indicates better model accuracy (the lower the better)
* R Squared - Represents the proportion of variance in the target variable explained by the model. Values closer to 1 indicate a better fit.

Decision Tree

Next, I trained a decision tree. This is similar to a flowchart where you keep asking questions at each branch. Each question/branch helps narrow down the price range.

In building and optimizing my models, I employed a systematic process for parameter tuning to enhance performance. For models like Decision Trees and Random Forest, I set *tuneLength = 10*, allowing the algorithm to automatically explore a range of hyperparameters. Once the optimal configuration was identified, I hardcoded these values to refine the final model, ensuring faster computation and improved accuracy.

K-Nearest Neighbors (KNN)

Next is the implementation of the K-Nearest Neighbors (KNN) algorithm, which estimates the target value by identifying the k-nearest data points in the feature space and calculating their average. For instance, to predict the price of a house, KNN would consider the prices of the 5 closest houses and compute their mean as the prediction. Once again, I tuned the k parameter to get the most accurate model.

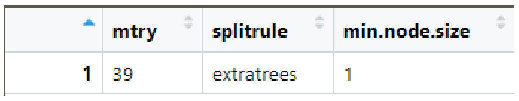
Random Forest

Random Forest is like having a whole group of decision trees that influence the final price. Each tree gives a prediction, and the forest takes the average.

For the decision tree model, using tuneLength = 10, I explored hyperparameters such as:

- Mtry - The number of features randomly selected at each split.
- splitRule - The criterion used to determine splits (e.g., Gini index, variance reduction).
- Min.node.size - The minimum number of samples required to form a terminal node.

These are the optimal hyperparameter ▸ configurations obtained. Giving: RMSE=220802 and R squared=0.82



	mtry	splitrule	min.node.size
1	39	extratrees	1

▲*Figure 2.2 - Optimal Random Forest*

After identifying these values, I hardcoded them to enhance the computational efficiency of the final model.

Stacking Ensemble

Finally, I tried stacking. This combines all the models I trained earlier into one supermodel.

I trained a meta-model (a model of models) to weigh the predictions from linear regression, decision trees, KNN, and random forest. The meta model itself is usually much simpler than the model that it is used to stack. The overall Meta model structure used in this project was a linear regression. The meta model works by training several base models; linear regression, decision tree, random forest and KNN in our case and working out the best way to incorporate all the models to make predictions.

Evaluating the Stacking Ensemble's predictions :

- **RMSE (Root Mean Square Error) = 178,359:**
  This means that, on average, the model's predictions deviate from the actual target values by approximately 178,359 units (e.g., pounds, if predicting house prices). A lower RMSE reflects better predictive accuracy.
- **R^2 (R-Squared) = 0.86:**
  The model explains 86% of the variance in the target variable, showing a strong fit. However, 14% of the variance remains unexplained, potentially due to noise or unmodeled factors.

K Fold Cross Validation for Evaluation

For all of these models, I used k fold cross validation, a technique that ensures our results are reliable and not just specific to one particular data split. Explanation :

- The dataset is divided into **k equal parts**, or "folds."
- Each fold takes turns being the "test set" while the remaining folds are the "training set."
- The model is trained and evaluated k times, once for each fold.
- The final performance score is the average of all k evaluations.

This method helps prevent overfitting and ensures that the model's performance is consistent across different subsets of the data. In this project, I used 10-fold cross-validation, where the data was split into 10 parts, making the results more robust.

Challenges Faced

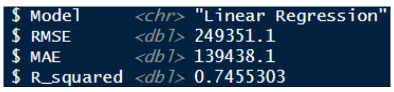
While stacking is a very powerful technique, it has a few extra challenges that you need to be cautious of.

Standardising Models - because all the models are incorporated into a single, we have to make sure that the base models produce and also receive results in the same format. This is one of the main reasons why I have set *preProcess = c("center", "scale"*) for all models. This makes sure that the input features are standardised. More specifically:

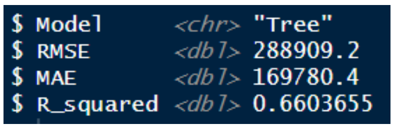
- Centering - Subtracts the mean of each feature from its values, resulting in a mean of 0 for the feature.
- Scaling - Divides each centered value by the standard deviation of the feature, standardizing the range to have a standard deviation of 1.

Finding the Best Investments

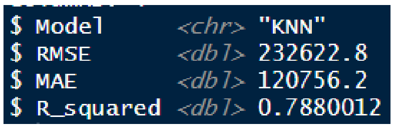
To find the best investments, we will use the best model.



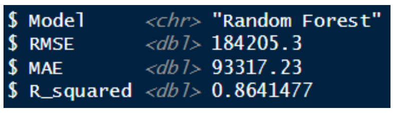
```
$ Model     <chr> "Linear Regression"
$ RMSE      <dbl> 249351.1
$ MAE       <dbl> 139438.1
$ R_squared <dbl> 0.7455303
```



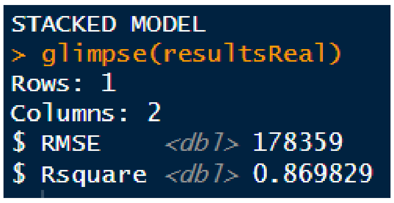
```
$ Model     <chr> "Tree"
$ RMSE      <dbl> 288909.2
$ MAE       <dbl> 169780.4
$ R_squared <dbl> 0.6603655
```



```
$ Model     <chr> "KNN"
$ RMSE      <dbl> 232622.8
$ MAE       <dbl> 120756.2
$ R_squared <dbl> 0.7880012
```



```
$ Model     <chr> "Random Forest"
$ RMSE      <dbl> 184205.3
$ MAE       <dbl> 93317.23
$ R_squared <dbl> 0.8641477
```



```
STACKED MODEL
> glimpse(resultsReal)
Rows: 1
Columns: 2
$ RMSE    <dbl> 178359
$ Rsquare <dbl> 0.869829
```

◂ *Fig 3.1 Model comparisons. We can clearly see that the stacked Ensemble is the best model - with the highest Rsquared and the lowest RMSE*

Once I had my best model (Stacked Ensemble) , I used it to find great investment opportunities. Here's how I did it:

1. Predicted the price for every house.
2. Compared the predicted price to the asking price.
3. Calculate the "investment potential" → Investment Potential=Predicted Price - Asking Price

I ranked all the houses by their investment potential and picked the top 200. These are the houses where the predicted price is much higher than the asking price, meaning they could be profitable buys.

Extensions and Future Ideas

Adding More Data

- Include economic factors like interest rates or inflation to see how they affect house prices.
- Use trends over time to predict how the market might change in the future.

Trying New Models

- Test more advanced models like Gradient Boosting Machines or Neural Networks.
- More in-depth analysis: Use clustering to group similar houses and create tailored models for each group.

Conclusions

Here’s what we achieved:

- Built a pipeline to predict house prices accurately.
- Found the best model that gave the most accurate predictions.
- Developed a clear strategy to identify top investment opportunities.

The results underline the potential of data-driven strategies in optimizing investment decisions. By leveraging these methodologies, investors can gain a clearer understanding of market dynamics and make informed choices.

For future improvements, incorporating additional data points, such as economic indicators, and testing more sophisticated models like gradient boosting machines or neural networks, could further enhance prediction accuracy.