London house price prediction model using machine learning

Cormac Meade

F136254

Loughborough University

MSc Data Science

Supervisor: Dr Parisa Derakhshan

Submitted on 25th August 2022

Contents

9 Conclusion
8.1 Future work

Abstract

In this project I build models for predicting future London property prices using machine learning techniques. These models were then compared in terms of accuracy and speed. The machine learning methods used are supervised regression techniques using tree-based ensemble methods of boosting, bagging and stacking. I decided which methods I felt would best fit the data by carrying out exploratory analysis of the dataset. Feature engineering was then carried out in order to reduce some of the noise and variation within the dataset. When evaluating the performance of the models, there was found to be considerable variation between the models in terms of efficiency and time taken to fit each model. However, the models were very similar in terms of the accuracy when measured on various error metrics. The chosen model performed considerably worse when predicting future observations than it did on previous observations. Further analysis was then carried out to investigate why this difference in performance between future and previous observations was so significant. This produced some interesting findings. In conclusion, the models built in this project perform well on predicting transaction prices of properties which fall in the mid-range of values, but they fail to take into account the significant variation of transaction prices. I identify a number of reasons why I felt this was the case and what could be done in future work to overcome this.

Acknowledgements

I would like to thank the following people for their support:

- My personal tutor Parisa Derakhshan
- Project coordinator Sara Saravi
- All the lectures which I have met during my time at Loughborough University
- My family and friends

1 Introduction

In this section I will start by discussing the importance of the UK housing market on the wider economy, before moving on to discuss the London housing market and how it differs from the UK housing market as a whole. I will then briefly discuss machine learning. And then discuss the objectives of this project, followed by the roadmap, and finally the potential challenges of this project.

1.2 Background

UK housing market

Household spending accounts for two thirds of Britain's economic activity (Anon., 2020). If you buy a newly built home, it directly contributes to total output (GDP) and this can cause a ripple effect through other sectors of the economy. For example through investment in land and building materials as well as newcomers using local shops and other businesses in the area. Mortgage lending is the main activity for banks and also the main source of debt in the UK, with more than seven out of ten pounds for mortgages (Anon., 2020). It is therefore clear to see that the housing market has a huge effect on the economy. An example of this was the 2008 financial crisis, the most serious financial crisis since the Great Depression (1929) (Anon., n.d.).

London housing market

figure 1 shows how London property prices have varied more and more over the years when compared to the UK as a whole and UK nations. The figure also shows that property prices in London are more volatile than prices in the UK as a whole, with prices not only rising at a sharper rate, but also falling more sharply. Back in 1969, the average London property was just 1.3 times the UK average. In 2016, the gap was 2.3 times the UK average (Cook, 2020). There are a number of key differences between London and the UK housing market as a whole besides price. These include the following, firstly, home ownership in London is 50.9% compared to 66.3% for the UK as a whole (Anon., 2022). Secondly, nearly 80 per cent of the UK's corporately owned residential property (whether foreign-owned or not) are in two London boroughs, Westminster, and Kensington and Chelsea (Anon., 2018).

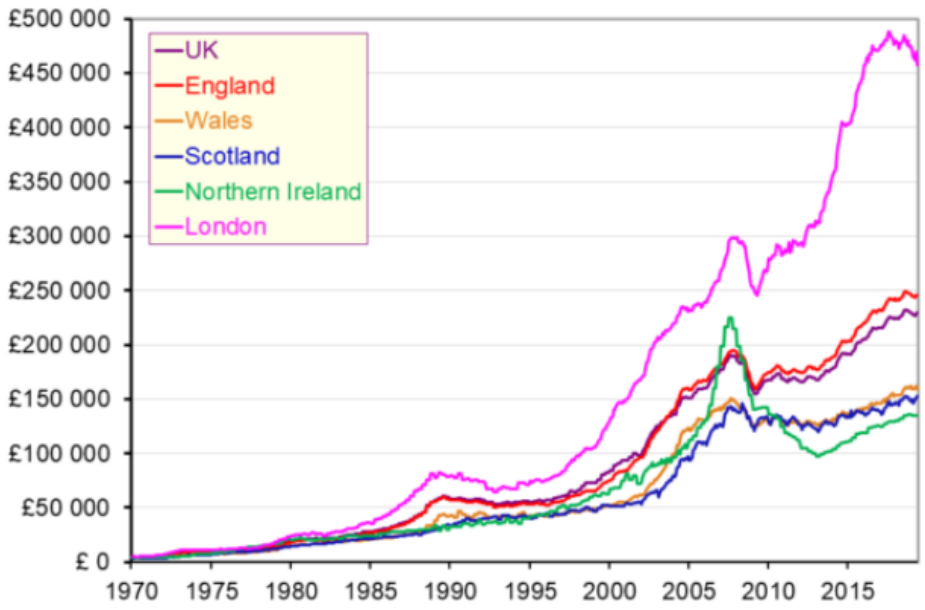


Figure 1: Comparison of average house prices in the nations of the UK as well as in London (Anon., 2019)

It is clear that the London housing market moves very different to that of the UK market in general. It is therefore necessary to develop a model for predicting house prices for London which is separate to that of the UK as a whole to take these differences into account. To do this, I will use machine learning. Machine learning is a subcategory of artificial intelligence which works by exploring data and identifying patterns, and involves minimal human intervention. Almost any task that can be completed with a data-defined pattern or set of rules can be automated with machine learning (Anon., 2020).

1.2 Objectives

The goal of this project is to build a machine learning model which can predict future London property prices. Various machine learning techniques will be explored and tested with past data on previous property transactions, before selecting the best performing model in terms of speed and accuracy which will be used to predict future property prices.

1.3 Roadmap

The roadmap for this project consists of six main steps which are summarised below:

1. Collect relevant data from HM Land Registry which consists of all commercial housing transactions from 1995 to the present

2. Using domain knowledge to select the features most relevant to the value of a property

3. Carry out in-depth exploratory analysis to gain a clearer understanding of the dataset

4. Transform the dataset into a form which is suitable to be put through machine learning models

5. Fit the data to the machine learning models and evaluate the performance using various error metrics

6. Determine the best performing model in terms of accuracy and efficiency and use this model to make predictions on future housing transactions

7. Attempt further improvements such as filtering the dataset

1.4 Challenges

This project has a number of challenges, the main ones are listed below:

- **The size of the dataset:** Since the size of the dataset is relatively large at just over two million inputs, training a model and making predictions on it will require more computational power. To overcome this, I increased the amount of RAM on my computer which increased the system speed and also the speed of the machine learning models.
- **Extreme response values:** London property prices vary significantly. This variability has increased in recent years including a number of extreme values which are much larger than 1.5 multiplied by the interquartile range. The response is also highly positively skew.
- **Limitations of the dataset:** despite the dataset being relatively large, it does not contain any information regarding size of property.
- **How to best represent certain features:** certain information such as location and date of transaction can be represented numerically in a number of way.
- **Data imbalances:** When observing the labels of certain variables, certain labels have significantly more observations than others. This can result in certain class labels having a much higher level of accuracy of predictions due to those labels getting trained on more data.
- **Variability of prices for the London housing market:** The London housing market is extremely varied in terms of prices paid. Building a model which can take into account this huge amount of variation will be very challenging.

2 Literature review

2.1 Introduction

Section 1 emphasised the importance of the housing market on the wider economy and how the London housing market differs from the UK housing market as a whole. In this day and age, there is an increasingly enormous volume and variety of data. The access and affordability of computational power, and the availability of high-speed Internet means that it is necessary to have a method which can analyse bigger, more complex data and deliver faster, more accurate results. This is where machine learning comes in. Machine learning gives us a way of performing data analysis with automated analytical model building (Anon., n.d.).

There are many ways to build a model for house price prediction. Some of these are:

1. Machine learning
2. Hedonic Price Model
3. Rule-Based Forecasting (RBF)
4. Data Mining
5. Fuzzy Logistic System

Due to this significance of the housing market on individuals and on the wider economy. Along with the popularity of machine learning in recent years. It is no surprise that lots of research has been done on the topic of 'house price prediction using machine learning methods'. As a result of this, eligible statistical methods for this study were limited to machine learning-based approaches for house price prediction. Due to advances seen in machine learning in recent years, I have also only included papers which were published no later than 2015. Eligibility for this literature review is summarized in figure 1.

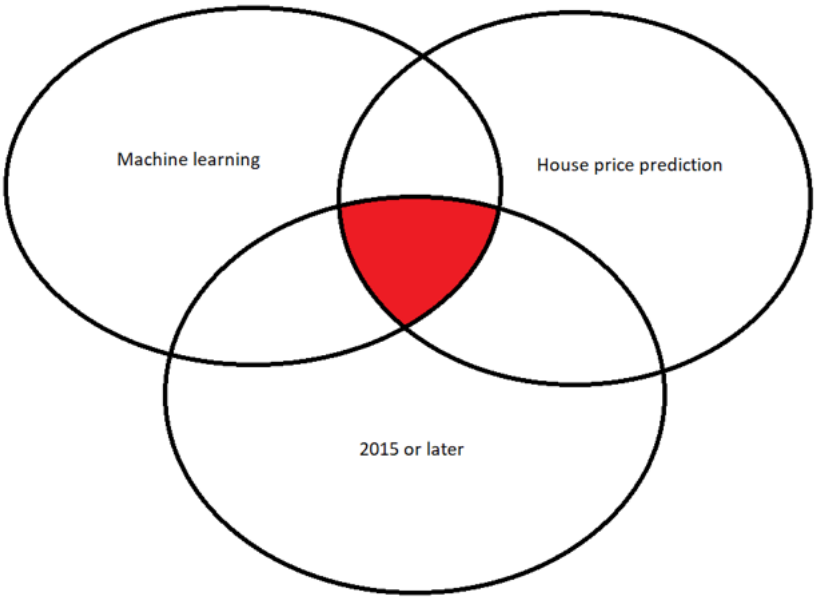


Figure 1: Criteria for this literature review

In this literature review I will aim to:

1. Justify the methods I am using
2. Justify my choice of data
3. Identify gaps in the area

2.2 Methods

The search strategy identified a total of 18 publications which met the search criteria. Of these, 17 were from Universities, 10 of which were student thesis (NG, 2015) (Vlahovljak, 2022), while 7 were research papers, and the final paper was from a property investment and financial journal. The most commonly used algorithms used were linear regression, random forest and SVM (39%), followed by decision tree, neural network, and stochastic gradient descent (28%). A total of 19 algorithms were used. The best performing algorithms were tree-based algorithms, coming out on top 60% of the time when multiple algorithms were experimented with. Ten papers used ensemble methods. In terms of the number of models used for each paper, the highest number was seven, while four papers just focused on one model.

2.3 Datasets

The datasets used varied quite significantly. In terms of the number of property transactions, four of the datasets had 3,000 or less, while three had more than a million. In terms of features which were used, all but three contained information on size of property, age of property was included in all but five. However, just seven datasets had information on the date of transaction, this will mean that many of the models won't be able to accurately predict future house prices due to not taking house price inflation into account. Another aspect I found interesting about the datasets was, only two dataset had information relating to the wider economy, such as GDP or inflation, and just one dataset included information on changes in population or the number of new dwellings.
In terms of similarities to my dataset, two papers are also using data from HM Land Registry 'Prices paid data'. One for London, and the other for the UK as a whole. Both of these papers chose to represent location in terms of longitude and latitude, and to represent the date of transaction in terms of months since Jan 1995.

2.4 Results

It should be noted that, due to the varying metrics of measuring performance used, and some datasets containing information on properties in areas with considerably cheaper prices than other areas which can make some models appear better than others. This makes it harder to compare like for like. Nevertheless, many of the papers achieved impressively high scores. From x achieving an R2 score of 0.90 for their SVM model, x achieving an R2 score of 0.92 for their XGBoost model, X achieving a MAE of 12,556 also using XGBoost, and Beng achieving an accuracy of within 10% of the actual price 39% of the time and within 20% of the actual price 62% of the time with their GP model with linear kernel based on London house prices using HM Land Registry 'Price paid data'. However, many of the better performing models were not predicting prices based on future house prices and were simply using a proportion of the previous data to test accuracy.

2.5 Summary

The existing research has shown that machine learning methods are capable of accurately predicting future house prices. We have also seen that despite the HM Land Registry 'Price paid data' not containing information on size of property, when effectively preprocessed, can still yield a high level of accuracy. The research has also shown that tree-based models can work well for house price data, coming out as the best performing model the majority of the time. The research has also allowed us to identify several gaps in the area, these are the following. Firstly, only one paper was seeking to predict prices exclusively for London which was in 2015. Given the advancements in machine learning over recent years, it is worth attempting to improve on the methods used in this paper. Finally, out of the eight papers which contain information on the year of transaction, just two use data which includes information relating to the wider economy such as GDP or inflation, and just one of these papers includes information on changes in population or number of new dwellings.

Figure gives a breakdown of the of the papers considered for this literature review in term so the datasets used, methods and results obtained.

Instances	Number of features	Location	Age of property	Property size	Transaction date	Region/Country	UK	Period (in years)	No. of models	Features relating to economy
2200000	5	Yes	Yes	No	Yes	Country	Yes	26	2	No
40000	5	No	Yes	Yes	No	Region	No	NA	3	No
2000000	5	Yes	Yes	No	Yes	Region	Yes	20	4	No
3000	80	No	Yes	Yes	No	Region	No	1	2	No
NA	11	No	No	Yes	No	Region	No	NA	1	No
3000	80	No	No	Yes	No	Region	No	10	7	No
NA	18	Yes	Yes	Yes	No	Region	No	NA	4	No
21000	17	No	Yes	Yes	Yes	Country	No	5	4	No
NA	79	No	Yes	Yes	No	Region	No	3	3	No
1200000	8	Yes	No	Yes	No	Region	No	NA	2	No
94000	26	Yes	Yes	Yes	Yes	Region	No	18	4	Yes
21000	26	yes	Yes	Yes	Yes	Region	No		1	No
32000	23	No	Yes	Yes	No	Region	No	NA	4	No
9000	79	No	No	Yes	No	Region	No	5	5	No
3000	10	No	Yes	Yes	Yes	Region	No	NA	4	No
NA	7	No	No	No	Yes	region	No	10	1	Yes
2000	6	No	Yes	Yes	No	Region	No	NA	3	No
27000	6	Yes	Yes	Yes	No	Region	No	NA	1	No

Changes in population/new dwellings	Year published	Ensemble methods	Type of paper	Best performing algorithm
No	2022	No	University thesis	Decision tree
No	2020	Yes	Research paper	Inconclusive
No	2015	No	University thesis	GP
No	2019	Yes	University thesis	Random forest
No	2021	No	University thesis	Inconclusive
No	2018	Yes	University paper	XGBoost
No	2021	Yes	University paper	Inconclusive
No	2017	No	University thesis	Inconclusive
No	2020	Yes	University thesis	XGBoost
No	2018	Yes	Research paper	Inconclusive
No	NA	Yes	Research paper	XGBoost
No	2017	No	Research paper	NA
No	2020	No	Research paper	SVM
No	2020	Yes	University thesis	Lasso regression
No	2018	No	University thesis	Inconclusive
Yes	2021	No	Research paper	NA
No	2021	Yes	Journal	Neural network
No	2018	Yes	Research paper	Random forest

Figure: Table with a breakdown of the papers considered for this literature review

2 Data

In this section I will start by explaining the dataset which will be used for this project including its limitations. I will then explore data relating to the wider economy and examine any relationship there might be with the target attribute 'Price purchased' to investigate whether any data relating to the wider economy can be added to our dataset.

2.1 HM Land Registry 'Price paid data'

All property sales since 1990 must be registered at the Land Registry (Telfor, 2019). Land Registry Price Paid Data (PPD) includes information on all property sales in England and Wales which were sold for value since 1995 up until the most current monthly data. This dataset can be downloaded in text or CSV format and the use is permitted for commercial and non-commercial purposes. The dataset includes housing attributes regarding location of property, date of transaction, property classification and the transaction price. A full list of attributes is as follows:

- Address (postcode, district, primary address, street, locality)
- Date of transaction (date when sale was completed)
- Unique ID (reference number to record each published sale)
- Property type (detached, semi-detached, terrace, flat, other)
- Old/New (indicating the age of the property)
- Tenure (freehold or leasehold)
- Transaction price

From this list, features have been identified which I believe affect the value of a property. I have chosen to represent the location of a property by district. The specific attributes are as follows:

1. District
2. Date of transaction
3. Property type
4. Age of property
5. Tenure

2.2 Limitations of the dataset

The main limitation of the Land Registry PPD is that it does not contain any characteristics relating to property size. Other limitations include a lack of information relating to the interior such as the number of bedrooms, number of bathrooms, availability of garages, and energy-efficient heating systems. Nevertheless, the available characteristics along with the size of the dataset should give us the potential to build a model which can predict the values of a property with a reasonable level of accuracy.

2.3 Data relating to the wider economy

The basis of all property value is what buyers are willing to pay (Team, n.d.). As well as characteristics of a property, it is well known that factors relating to the wider economy also affect what someone is willing to pay for a property.

Areas of the wider economy which are thought to affect house prices include the following. Firstly, GDP. House prices tend to rise if people expect to be richer in the future. Normally that happens when the economy is doing well as more people are in work and wages are higher (Anon., 2020). Secondly, supply and demand. Demand for housing may rise if the population is increasing or there are more single-person households. Prices will also tend to be higher if fewer houses are built, reducing the supply of housing (Anon., 2020).

Figure 1 shows changes in London house prices using Land Registry PPD vs changes in UK GDP and vs changes in the total population of London.

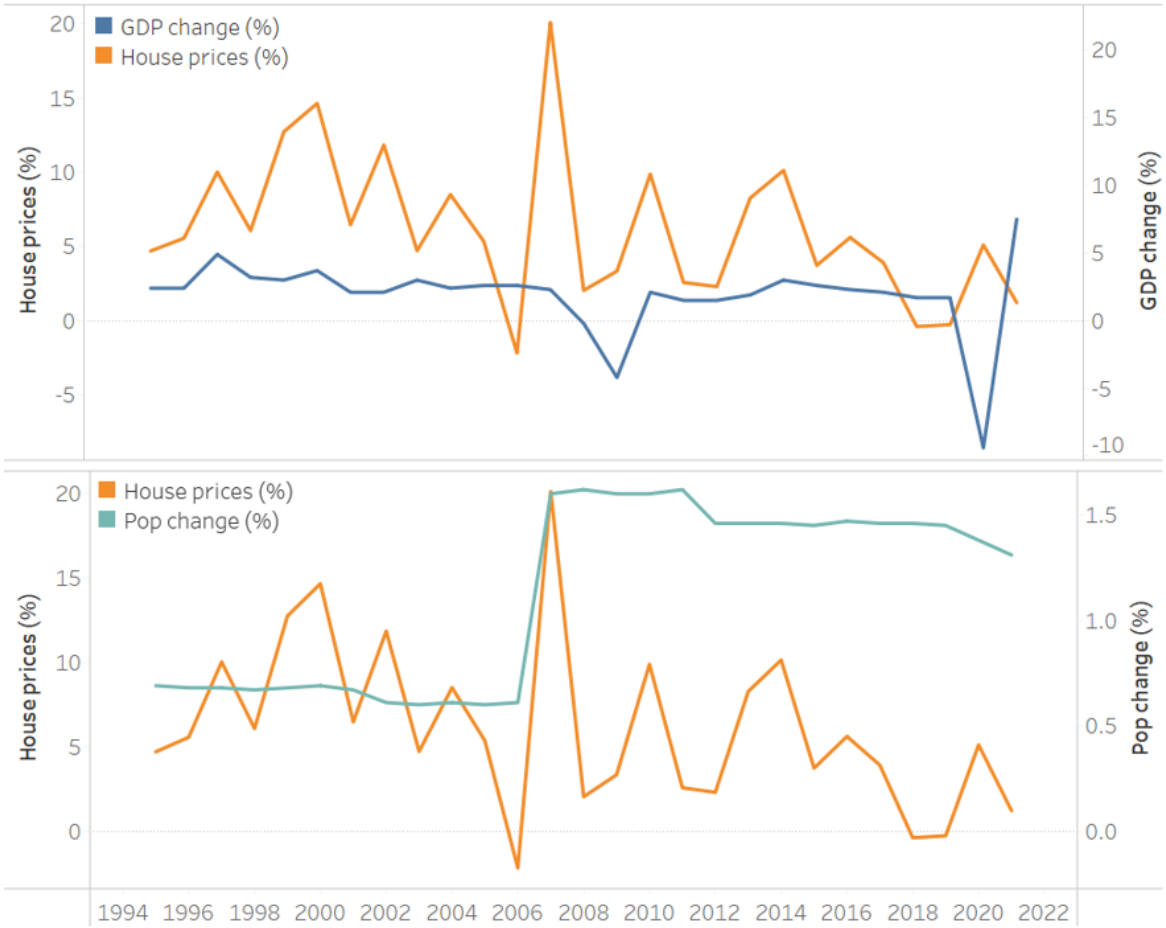


Figure 1: Relationship London property price has with UK GDP and with London population 1995-2021 (Anon., 2022), (Anon., 2022), (Anon., 2022)

The figure shows that based on the Land Registry PPD, London house prices do not have a particularly strong relationship with UK GDP and the overall population of London. Reasons for this could be. The UK economy as a whole not moving in line with the London economy, not considering the amount of net additional dwellings in London each year which would give us a clearer picture of housing supply and demand in London. Due to figure 1 and the lack of publicly available data

regarding the London economy stretching back as far as 1995, I will not be including characteristics relating to the wider economy in my analysis.

3 Preliminary analysis

In this section I will start by exploring the relationships the input variables have with the target variable and also the relationships between input variables. I will also explore the distributions of each variable. Following this, I will clean the data where required by removing any incorrect data and extreme values. Finally, based on the findings from 3.1, I will transform various aspects of the dataset so that the models will be able to pick up patterns more easily.

3.1 Exploratory analysis

Figure 4 shows the linear relationship each input variable has with the target attribute, price purchased, and also the linear relationship the input variables have with each other in the form of correlation coefficients. The range of the coefficients is [-1, 1], the further the absolute value is away from zero, the stronger the relationship between two variables. We can see that time, given as months since January 1995, has by far the strongest linear relationship with transaction price with a relatively strong correlation of 0.69. The binary variables detached, semi-detached, terrace, new build and freehold all have an absolute value of less than 0.1. the location variable, area, also has a correlation of less than 0.1, although this is to be expected since the relationship between price purchased and area is not linear.

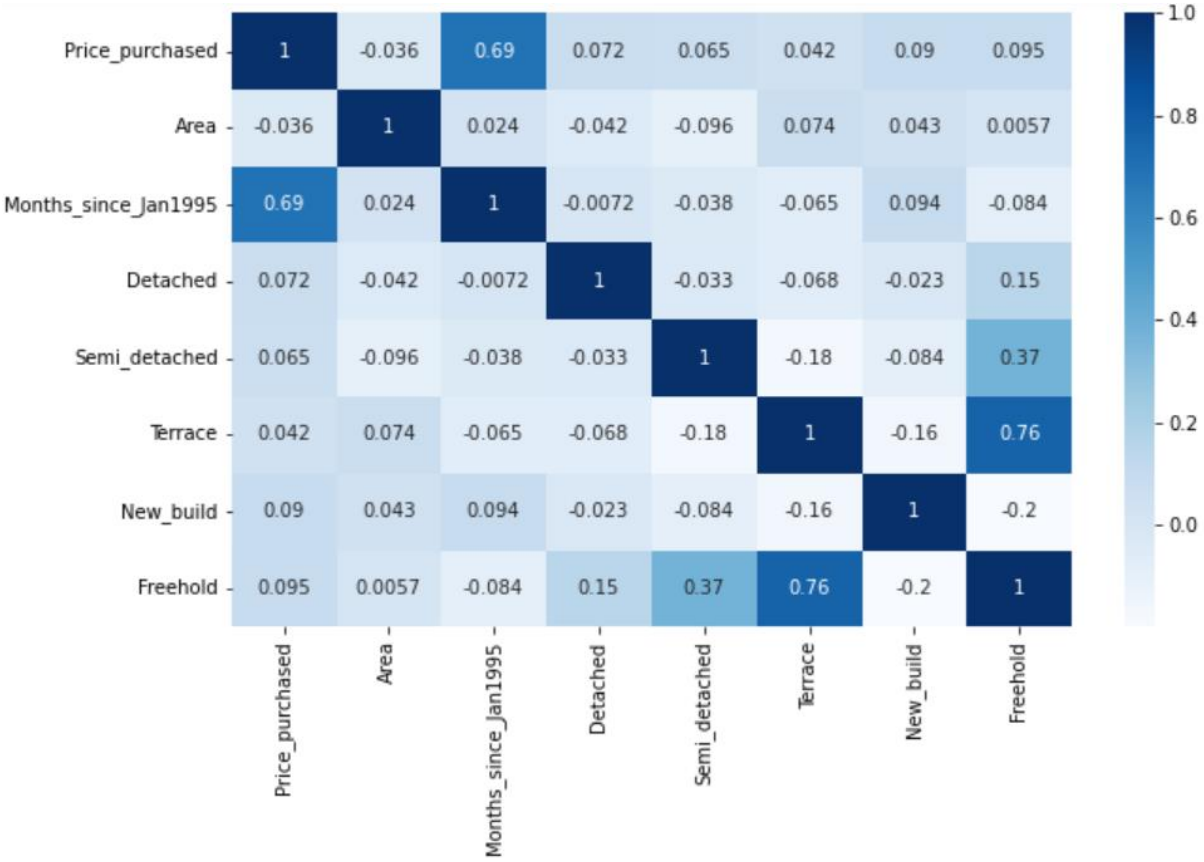


Figure 4: Correlation coefficients between the input variables and between the input variables and the target attributes

Figure 3 shows the total number of transactions in each London district over the period 1995-2021. As can clearly be seen, the number of transactions differs hugely between districts, with four districts having just a single transaction over the entire period, while Wandsworth has almost 180,000 transactions. Two classes in this variable are not London districts (Malvern Hills and Rhondda Cynon Taff).

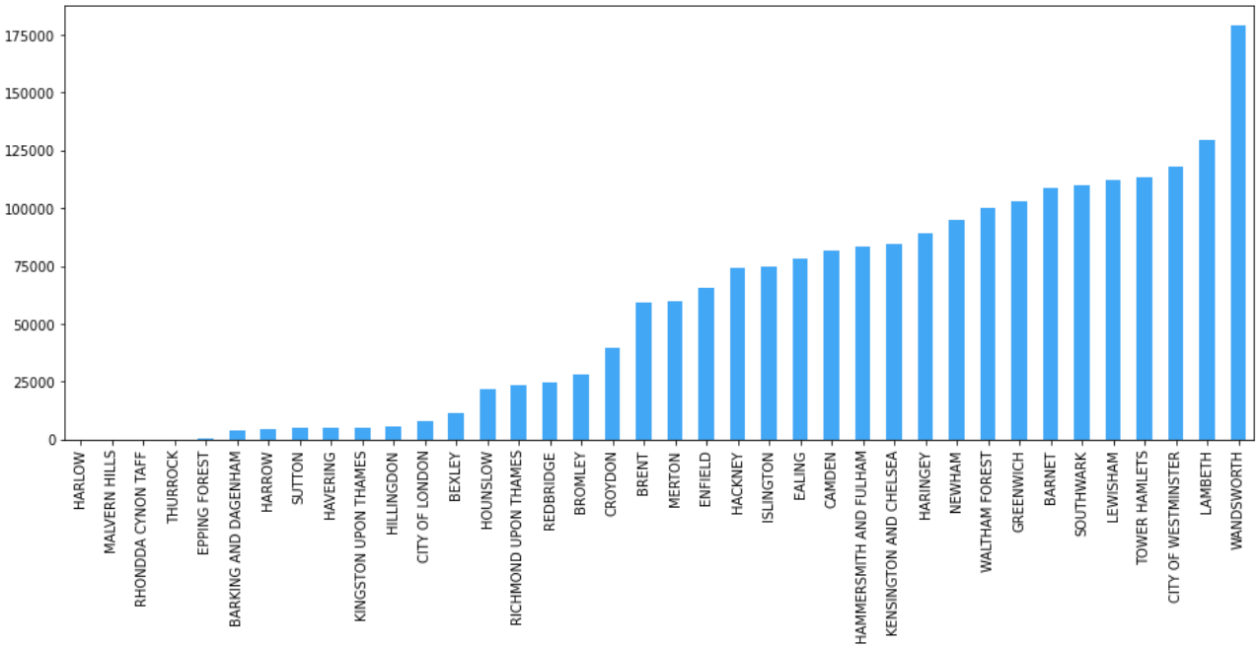


Figure 2: Total number of transactions in each London district over the period 1995-2021

Figure 3 shows the distribution of prices paid (in thousands) for each year of the dataset in the form of boxplots. As expected, the distributions contain many values which differ hugely from the range of the majority of the data. So much so that the only information which can be gathered from inspecting the individual boxplots is the values of the most extreme instances. The highest of which reaching a whopping 160 million. The number of extreme values also appears to increase with time. It is likely that these values are not outliers and simply represent transactions which are very different to the rest of the data.

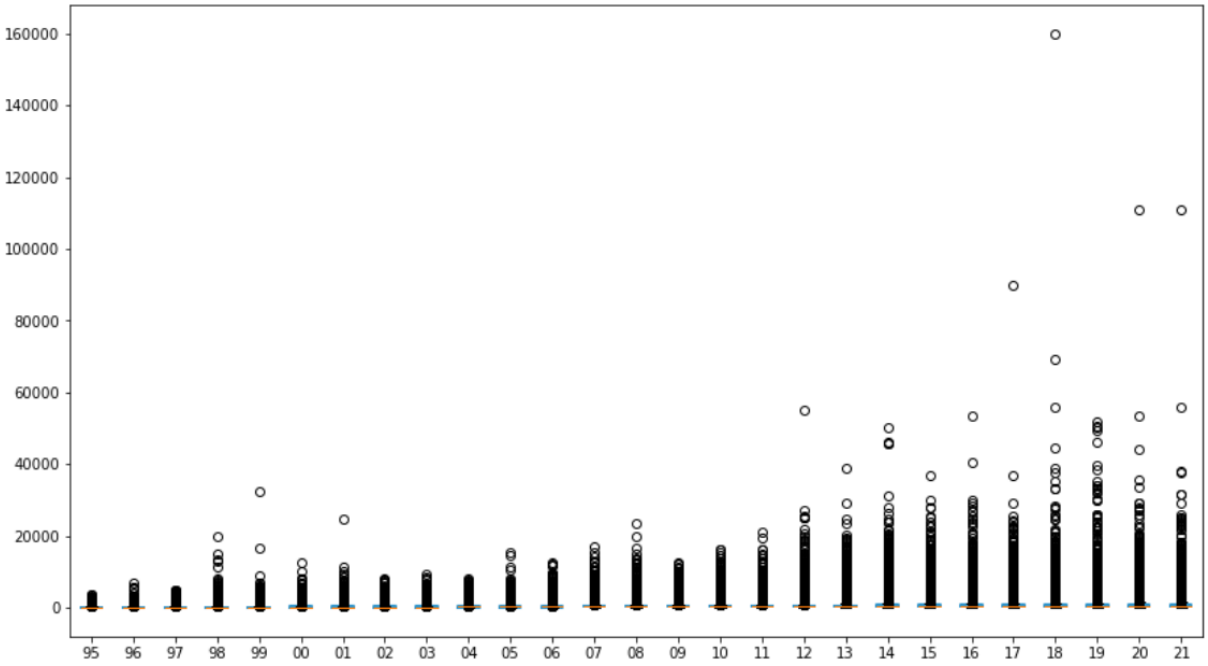


Figure 3: Boxplots of distributions of prices paid in thousands each year 1995-2021

Figure 4 shows the number of property transactions sorted property type, estate type and whether a property is a new build or not. We can see that, over half the transactions are for flats, with a proportionally tiny number of transactions for detached houses. Around two-thirds of transactions took out a leasehold rather than a freehold, and the vast majority of properties were not considered to be new builds.

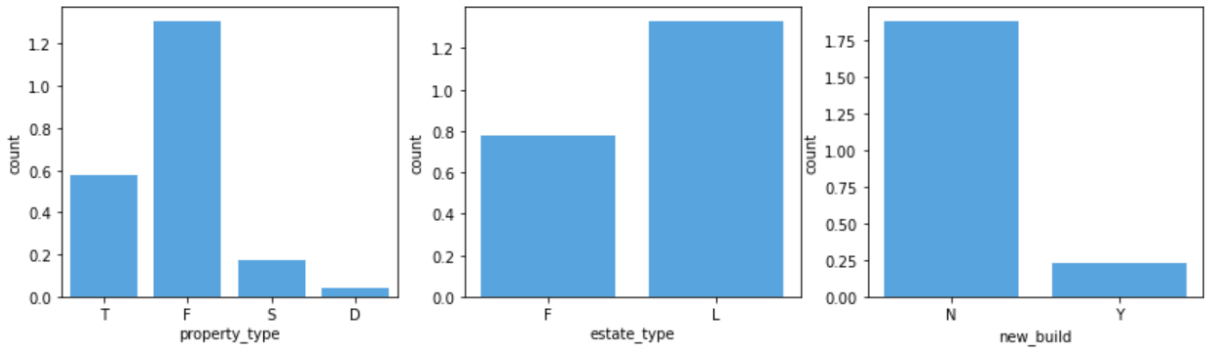


Figure 4: Number of property transactions sorted by property type, estate type and new build

From figure 5, we can see that there were more transactions in the period up until 2007 than there were in the years prior. With 2006 having significantly more transactions than any other year. This could be due to the housing bubble bursting towards the end of 2007.

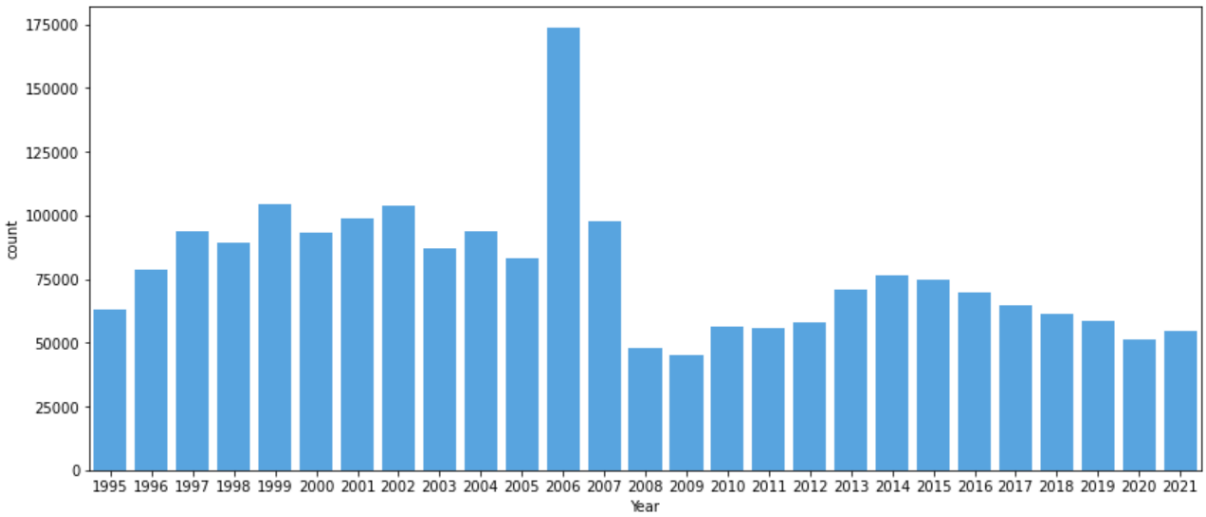


Figure 5: Number of property transactions by year

3.2 Data cleaning

The first step was to remove incorrect records from the dataset. While the dataset is relatively clean, the following errors were detected and removed:

1. Instances which have property type given by O.
2. Instances which have estate type given as ‘no_value’.
3. Instances which have district given by RHONDDA CYNON TAFF or MALVERN HILLS as these are not districts in the greater London area

This has reduced the size of the dataset by 33,386 rows.

3.3 Data transforming

Since machine learning models require all features to be numeric, we must encode any categorical features into an appropriate form before fitting the data to the models. The following changes were made:

1. Categorical variables property type, estate type and new build are each transformed using one-hot encoding. What this does is creates new binary columns indicating the presence or otherwise of each possible value from the original categorical variables represented by 1 or 0.
2. Label encoding is applied to the district variable. In label encoding we replace the categorical value with a numeric value between 0 and the number of classes minus 1.

3. The date of transaction was converted into the number on months since January 1995 to give a simplified way to represent time.
4. The price purchased variable was divided by 1000 so that this variable does not dominate too much when it comes to predicting values.

Due to the huge imbalance in the number of transactions in each district over the period 1995-2021 as seen in figure 2. I decided to combine districts with less than 5,000 transactions with neighbouring districts. The districts combined were as follows:

1. Kingston and Richmond
2. Harrow and brent
3. Sutton and Croydon
4. Harlow, Epping Forest and Redbridge
5. Barking and Dagenham, Havering and Thurrock

Machine learning models are sensitive to the range and distribution of attribute values (Tripathi, 2020). I will therefore remove instances for each year of the dataset which fall above equation 2.1. Where Q3 is the value under which 75% of data points are found and IQR is the spread of the middle half of the distribution. Since the mean overall house price in the dataset is nearly £150,000 larger than the median value, I will not remove any extreme values at the lower end of the distribution as this will increase the spread between the mean and the median. Getting the mean as close to the median as possible will reduce the likelihood of the machine learning models over valuating the target attribute since a higher mean relative to the median tells us that the majority of observations are below this value.

$$Q3 + 1.5(IQR) \quad (2.1)$$

The distribution of values with extreme values removed is shown in figure 4. While the boxplots still show a certain amount of positive skew, the distributions are relatively normally distributed. For all years, the mediun is close to the middle of the box and the whiskers are a similar length on the top and bottom of the boxes. This will now increase the machine learning models ability to describe typical cases as it won't have to deal with rare one-off cases. The distance between the mean and the median has also reduced to 58,000 from 149,000.

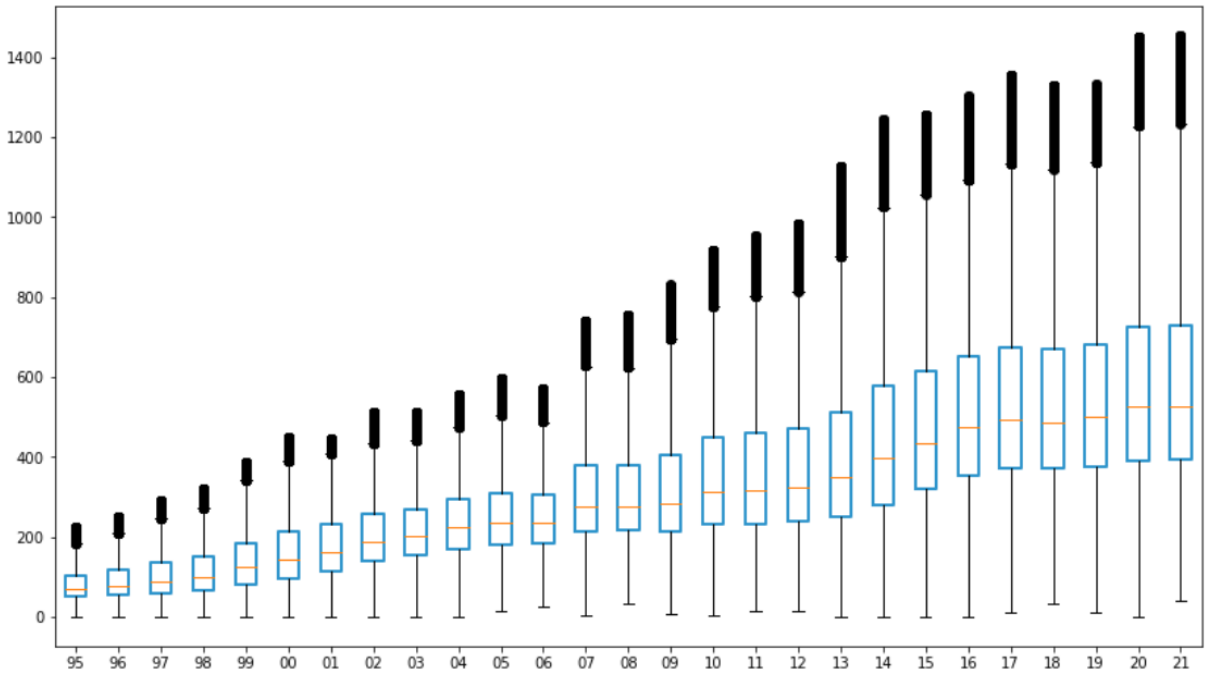


Figure 4: Boxplots of distributions of prices paid in thousands with extreme values removed each year 1995-2021

From figure 5, we can see that most values are still clustered around the left tail of the distribution while the right hand tail is longer. Figure 6 shows that more recent years contain more extreme values than earlier years. For this reason, no further attempt will be made to normalize the distribution as more recent years will reflect future house prices a lot more than older transactions. We therefore do not wish to remove too many transactions from recent years.

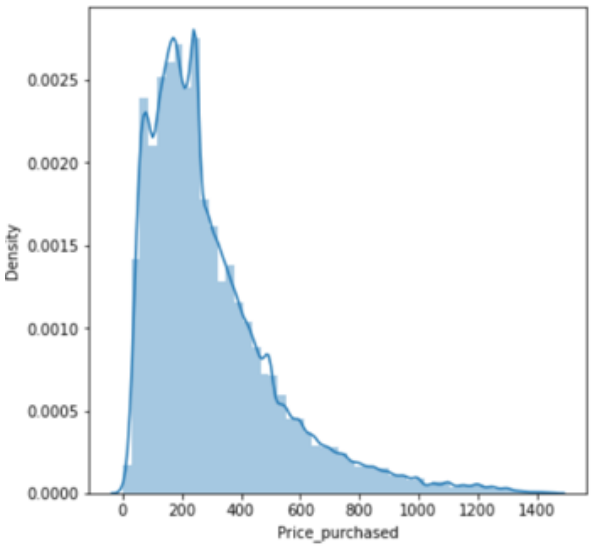


Figure 5: Distribution of price purchased

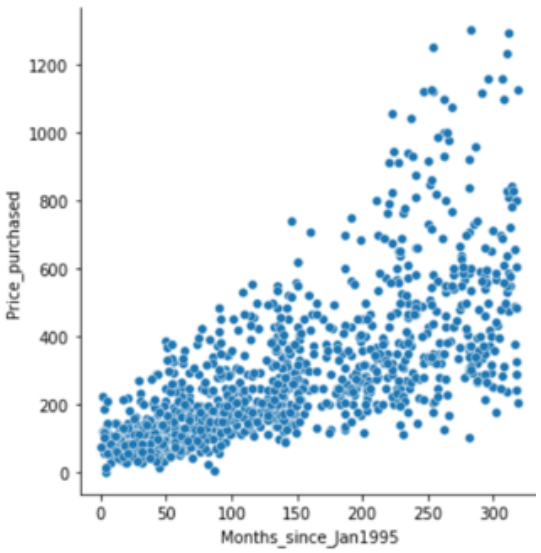


Figure 6: Scatterplot of price purchased over time

4 Methods (check plagorism of this section)

In this section I start by giving a more indepth overview of the machine learning including its real world applications, why the demand for machine learning has grown in recent years, the process of how it works in a nutshell, and the main techniques of machine learning. I then discuss my approach such as the algorithms I will be experimenting with and giving an overview of how each of them work.

4.1 Machine learning Overview

In this day and age there is a huge amount of data available. From pictures, words, spreadsheets, music and videos which are generated on computers, mobiles and other devices. Anytime a person browses the internet or makes a transaction, this generates data (Marr, 2018). Previously, computers could only do what humans programmed them to do. As the volume of data surpasses the ability for humans to make sense of it, we will turn increasingly to automated systems that can learn from the data (G, 2017). With machine learning, computers can behave almost like humans do and gain knowledge based on their past experiences. Many of today's leading companies, such as Facebook, Google, and Uber, make machine learning a central part of their operations (Mahajan, 2021).

How it works

The process of a machine learning algorithm can be broken down into three main steps:

1. **A Decision Process**: In general, machine learning algorithms are used to make a prediction or classification. It takes some input data, this can be either labelled or unlabelled, the algorithm will then produce an estimate regarding a pattern in the data
2. **An Error Function**: Typically in machine learning we aim to reduce the difference between the predicted values and the actual values. An error function evaluates this difference, there are a number of error functions available for both regression and classification problems.
3. **A Model Optimization Process**: The overall aim of machine learning is to create a model which gives accurate predictions. if the model can fit better to the data points in the training set, weights can be adjusted to reduce the disparity between the predicted and actual values. This process of evaluate and optimise will be repeated until a threshold is achieved.

4.2 ML methods

Machine learning has three main techniques with are summarised below:

1 **Supervised learning:** Supervised learning is an approach which uses labelled data to feed to the model in order to train the machine. The model is trained until it can detect the underlying patterns and relationships between the exploratory variables and the target variable. The enables us to obtain accurate results on unseen observations. There are two main types of supervised learning, classification and regression. These two methods differ in the type of output in which they predict. In classification, the response is a discrete categorical variable. In regression, the target variable is continuous.

2 **Unsupervised learning:** Unsupervised learning is an approach which uses unlabelled data as a training set, therefore, we do not know the output. Unlike supervised learning, there are no correct output values. The model attempts to determine the patterns within the dataset, rather than relating it to some external measurement.

3 **Reinforcement learning:** Reinforcement learning is a type of learning which is based on reward and punishment which works on the principal of feedback. The model uses trial and error in the form of positive or negative feedback based on whether it correctly predicted an output.

4.3 Applications of ML

Machine learning is all around us (Anon., n.d.). Many of us use machine learning in some way or another every day without even realising it (Anon., n.d.). Some examples of machine learning in our everyday lives are the following:

4 **Virtual personal assistants:** Alexa, Siri and Google Now are some of the most popular assistants. They assist in finding information when asked over voice (Anon., n.d.).

5 **Video surveillance:** Involves observing a scene or scenes and looking for specific behaviours.

6 **Email spam:** Machine learning can successfully detect and filter spam emails by using a number of spam filtering techniques.

7 **Social media features:** Social media platforms use machine learning approaches in an attempt to improve the overall customer experience on the platform by monitoring a person's activity.

8 **Product recommendations:** Product recommendation is one of the most popular machine learning techniques and is used by most e-commerce websites today (Anon., 2022). This allows them to provide a personalised experience to customers on their platform by recommending products based on a user's search history.

9 **Image recognition:** In the context of machine learning is the ability of software to identify objects, people, writing and actions in images (Anon., n.d.). It's many uses include facial recognition and other security systems, as well as to develop driverless cars

4.4 My approach

Since the goal is to predict continious values, this is a supervised learning regression problem. My approach to this problem will be to use tree based machine learning methods. Starting with the basic decision tree as the base model to measure the performance of other models against, I will then build tree-based ensemble models in an attempt to improve the performance. Tree-based methods use one or more decision trees to generate predictions and can handle both numerical and categorical data for regression or classification problems. Tree-based algorithms are generally considered to be a good choice when there is a level of non-linearity and complexity between the target variable and some of the input variables (Anon., 2016). They also perform well with large datasets since large amounts of data can be analysed using standard computing resources in reasonable time (Anon., n.d.).

Tree based regression

Tree-based regression is a family of supervised Machine Learning which performs regression tasks by building a tree-like structure by learning simple decision rules for estimating the value of the target variable according to the features. If feature values are not categorical then they are discretized prior to building the model.

It has three types of nodes:

- **The root node**: Is the initial node and represents the entire data sample. In a normal decision tree the root node is chosen to be the input variable which best splits the data [].
- **Interior nodes**: These represent the remaining features of the dataset and split the data further. Each feature is split based on a condition which best splits the data
- **Leaf nodes:** These are the final nodes of the tree where the predictions of each data point are made. Without defining the maximum depth of the tree, these will be pure leaf nodes where all of its data belongs to a single class.

The decision rules are given by the branches. Figure 7 shows the basic structure of a decision tree.

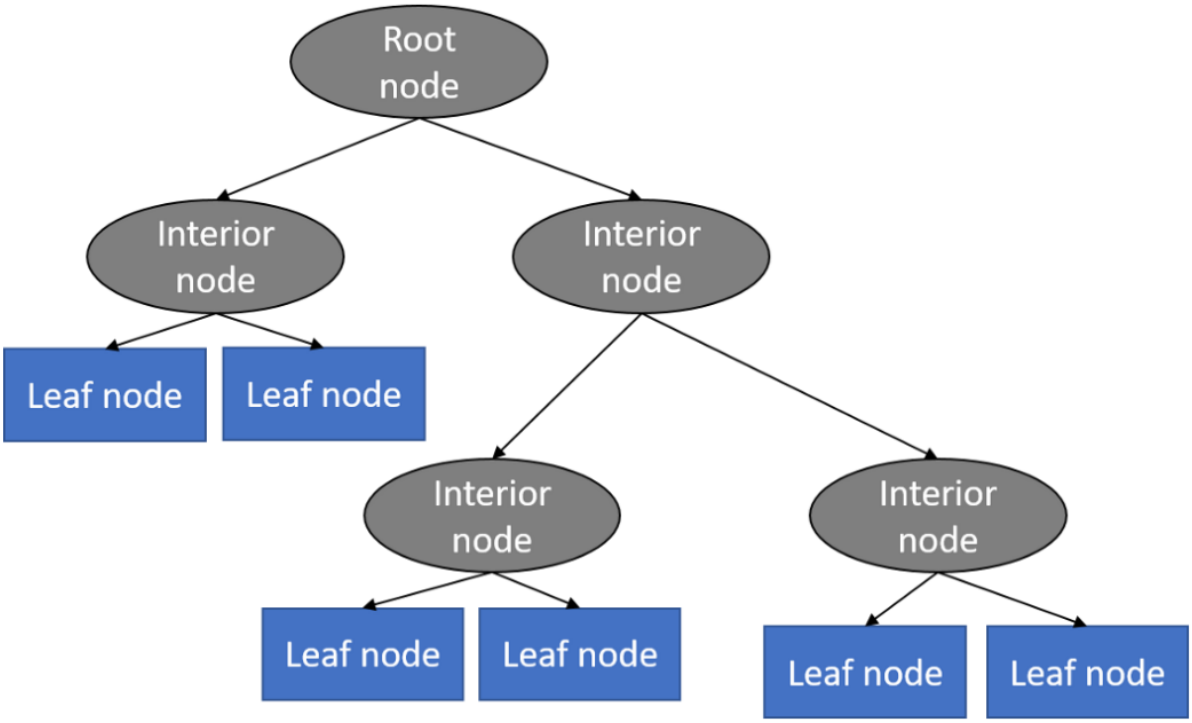


Figure 7: Diagram showing the basic structure of a decision tree (Chatterjee, 2019)

Algorithm

There are various decision tree algorithms. Sklearn uses an optimised version of the Classification and Regression Trees (CART) algorithm (Anon., 2022). In this algorithm, each data point is run through the entire tree by answering binary questions until it reaches the leaf node. For each variable, the model experiments with splitting the data at many different values and calculating the impurity function, that is, the value which best splits the target variable into higher and lower values. Methods available in the sklearn package for calculating the impurity include mean squared error, Poisson deviance and mean absolute error. If a feature does not split the dataset well in terms of the values of the target variable, this feature will not be included in the final model. Once a particular datapoint has reached a leaf node, its value is predicted by calculating the average value of the target variable in the particular leaf node the point lies in.

Advantages of decision tree

Some of the advantages of using a decision tree over other models are given below:

1. Decision Trees are easy to explain as they result in a set of rules.

2. No feature scaling required such as standardisation discretisation due to its rule-based approach instead of distance calculation (Kumar, 2019)
3. Interpretation of a complex Decision Tree model can be simplified by its visualizations.
4. Non-linearity does not affect the models performance
5. Less resources are required for computation, hence it consumes less time than more complex tree algorithms
6. Performs well even if its assumptions are somewhat violated by the true model from which the data were generated (Anon., n.d.)

Drawbacks of decision tree

Some of the drawbacks of decision tree model are as follows:

1. There is a high probability of overfitting in Decision Tree. Since if a number of input values change, we will get a completely different tree.
2. Generally, it gives low prediction accuracy for a dataset as compared to other machine learning algorithms.
3. Calculations can become complex when there are many class labels.
4. A small change in the data can cause a large change in the structure of the decision tree causing instability (K, 2019)

Ensemble learning

Ensemble learning techniques combine multiple models together in order to find a model with better accuracy and robustness than an individual model could. The ideology behind ensemble learning is that certain models do well in modelling one aspect of the data, while others do well in modelling another (Massaoudi, 2020). Instead of learning a certain complex model, learn several simple models and combine their output to produce the final decision. The combined strength of the models offsets individual model variances and biases (Singh, 2018). This provides a composite prediction where the final accuracy is better than the accuracy of individual models.

This improved accuracy does however come at a cost. Unsurprisingly, evaluating the prediction of an ensemble typically requires more computation than evaluating the prediction of a single model. In a sense, ensemble learning may be thought of as a way to compensate for poor learning algorithms by performing a lot of extra computation. On the other hand, the alternative is to do a lot more learning on one non-ensemble system. An ensemble system may be more efficient at improving overall accuracy for the same increase in compute, storage, or communication resources by using that increase on two or more methods, than would have been improved by increasing resource use for a single method (Anon., n.d.).

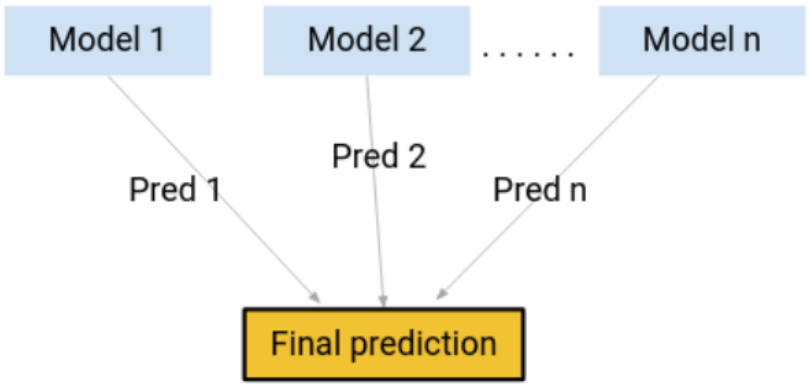
Ensemble learning methods can be split into two groups. Sequential methods and parallel methods. In sequential methods, base learners are generated consecutively. The basic motivation of sequential methods is to use the dependence between the base learners by weighing previous values which were badly predicted with higher weight (LENDAVE, 2021). The basic motivation of parallel methods on the other hand is to use independence between base learners. They are applied wherever the base learners are generated in parallel. Since the errors are often significantly reduced by averaging the output of the base learners.

While there are an unlimited number of ensembles that can be developed for predictive modelling. There are three main classes of ensemble learning, these are:

1. **Bagging**: which is used to decrease variance
2. **Boosting:** which is used to decrease bias
3. **Stacking:** which is used to improve predictions

Boosting

The idea behind boosting is to combine multiple rules of thumb in order to make an accurate prediction. Boosting is an ensemble technique which uses a set of machine learning algorithms to combine weak learners in order to make a strong learner and thus increase the accuracy of the model. Boosting is a sequential ensemble learning technique, the base models are therefore dependant on each other.



A weak learner is one which has weak correlation with the target attribute. A strong learner has strong correlation with the target. These weak learners are generated by applying decision tree algorithms on different distributions of the dataset.

How boosting algorithm works

The entire dataset is put through the first base model, this model will then make some predictions. The dataset will then get put through a second base model. However, each observation will be assigned a weightage relative to how well base model 1 predicted its value with observations which gave worse predictions being assigned higher weights. This whole process will continue until all observations give predictions which meet a certain threshold. Predictions are made on unseen values by putting them through all of the base models. The average of all these outputs will then be calculated to give the predicted value.

Bagging

The Idea behind bagging is to combine the results from multiple decision trees to get a generalised result from a single model. By training a model on a single dataset there is always a risk of the model overfitting to the training data (Brownlee, 2018). A way of reducing this likelihood is to take many samples from the original dataset with replacement and train the model on each of these sample datasets. This process of taking samples from the dataset with replacement is called row sampling with replacement. When making predictions we can then take the average of all of these outputs. Since the individual decision trees are weak learners, it is very unlikely that they will overfit to the training data.

There are 3 steps of performing bagging which are summarised below:

1. Create randomly sampled datasets of the original training data (bootstrapping)
2. Build and fit several unpruned decision trees on each sample dataset
3. Take an average of all the predictions to make a final overall prediction

Bagging is a parallel learning process. Therefore, the individual models are built independently of each other and hence the models are independent of each other.

In summary, the contribution of bagging is in the varying of the training data used to fit each ensemble member, which, in turn, results in skilful but different models.

Stacking

Stacking is a process which involves training a model to combine the predictions of several other models. It addresses the question Given multiple machine learning models that are skilful on a problem, but in different ways, how do you choose which model to use (Brownlee, 2020)? The approach to this question is to use another machine learning model that learns when to use or trust each model in the ensemble. The benefit of stacking is that it can harness the capabilities of a range of well-performing models on a regression task and makes predictions that have better performance than any single model in the ensemble (Brownlee, 2020).

- Unlike bagging, the models are typically different (I.e. not all decision trees) and fit on the same dataset.
- Unlike boosting, a single model is used to learn how to best combine the predictions from the contributing models.

The architecture of a stacking ensemble is as follows:

1. Training data is put through several base models
2. The predictions from these models are then taken and combined to form a matrix of size m x M where M is the number of base models used
3. This dataset then becomes the second level model
4. This second level model then makes the final predictions

Figure 9: Basic architecture of a stacking ensemble (Anon., n.d.)

The training data for the second level model is created as follows:

1. Split the training data into K-folds just like K-fold cross validation
2. A base model is fitted on the K-1 parts and predictions are made for the Kth part
3. We do this for each part of the training data

4. The base model is then fitted on the entire training dataset to calculate its performance on the test set
5. Stages 2-4 are repeated for each of the base models

Random forest

Random forest is an ensemble learning method which uses bagging in order to build models and uses decision trees to build each base model. It is therefore a parallel learning method. It is considered an extension to bagging since it applies feature selection as well as row selection during bootstrapping to select the random samples for each of the base models.

The algorithm establishes the outcome on the predictions of the decision trees. The term random is due to using two random processes: bootstrapping and random selection. There are no formal distributional assumptions, random forest are non-parametric and thus can handle skewed or multi-modal data as well as categorical data (Richmond, 2016).

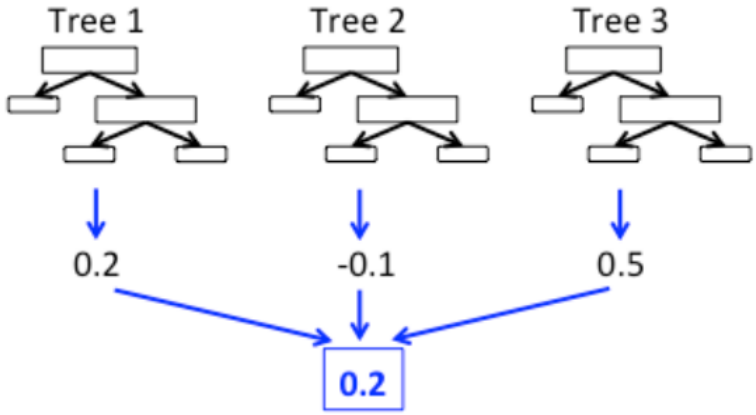


Figure 8: Example of a random forest regression (Anon., n.d.)

Construction of a random forest

It builds new datasets from the original dataset by randomly selecting rows. All these datasets will contain the same number of rows as the original dataset. This means that rows can be selected more than once. This process of creating new data is called bootstrapping. A decision tree will now be trained on each on each of the bootstrapped datasets independently. Not all the variables will be used to train the trees, instead variables will be randomly selected to form a subset. Once the trees have been trained predictions are made on new datapoints by passing them through each decision tree one by one. All the predictions are then combined and divided by the number of trees to obtain the average value. This value is the prediction made for the particular datapoint. This process of combining results from multiple models is called aggregation. The process of performing bootstrapping followed by aggregation is called bagging.

Overcomes short comings of decision tree

Random forest helps us overcome a number of limitations associated with a basic decision tree:

1. It is less prone to overfitting since random forest is a collection of many random decision trees.
2. Bootstrapping ensures that we're not using the same data for each tree, hence it helps our model become less sensitive to the original training data (Anon., 2021).
3. Random feature selection helps to reduce the correlation between trees.

XGBoost

XGBoost stands for extreme gradient boosting trees and uses the ensemble method of boosting. It is implemented on top of gradient boost and is a scalable, distributed gradient-boosted decision tree (GBDT) machine learning library which provides parallel tree boosting (Anon., n.d.).

XGBoost is considered one of the fastest and most accurate models used in machine learning (Brownlee, 2016). Reasons why XGBoost is able to work very fast are as follows:

1. Parallelisation processing
   Parallelisation processing is a method of running two or more processors (CPUs) to handle separate parts of an overall task. Breaking up different parts of a task among multiple processors will help to reduce the amount of time taken to run a program (Anon., n.d.).
2. Cache optimisation
   Cache is a special storage space for temporary files that makes a device, browser, or app run faster and more efficiently. Optimisation of cache performance ensures that it is utilised in a very efficient manner to its full potential (Anon., n.d.)
3. Out of memory computation
   If we have data that is larger than the size of our memory
   XGBoost has a way of optimising the memory in such a way that it can work on data that is larger than the size of the ram (Anon., 2019)

Reasons XGBoost is able to give a high level of accuracy are as follows:

1. Regularisation
   XGBoost has a built-in regularisation parameter which prevents the likelihood of model from overfitting and hence variance of the model is controlled (Anon., 2019).
2. Auto-pruning of the trees
   XGBoost will not allow the trees to grow beyond a certain level (Anon., 2019). This will also reduce the likelihood of overfitting

The steps involved in any gradient boosting model are as follows:

Steps

1. Initially builds the first model and calculates the error for each observation in the dataset
2. Once we have these residuals, we then build a new model to predict those residuals (errors) in the first model
3. We then add this model to the ensemble models,

We repeat this process of calculating residuals, building a model to predict the residuals, and adding this to the ensemble over and over again until the model becomes extremely powerful.

LightGBM

LightGBM is short for light gradient boosting machine. LightGBM shares many similarities of other gradient boosting models. A major problem of gradient boosting is that it is slow to train models on large datasets consisting of millions of rows (Saini, 2021). LightGBM utilises a number of techniques which make it quicker and more efficient.

There are two main techniques which make LightGBM faster than other gradient boosting methods:

1. Histogram binning
   Training the trees that are added to the ensemble can be dramatically accelerated by discretizing (binning) the input variables to a few unique values. This way the algorithm doesn't need to evaluate every single value of the features to compute the split. LightGBM implements this approach by creating a histogram of feature values, where the values of each feature are put into bins of equal size built using a subset of the dataset.

2. Gradient based one-side sampling
   Once the base model has been trained on the dataset, all observations will be ordered based on its error (gradient). The dataset will then be split into two subsets, subset A, which will consist of the values with the highest errors (E.G., highest 20%). Subset B will consist of the remaining data. A sample will then be made consisting of subset A and a percentage of subset B (E.G., 10%). The idea behind this approach is that less observations need to be considered when the individual trees are split.

Adaboost

Adaboost short for adaptive boosting is a sequential learning process. Therefore, each model depends on the output of the model before it and hence the models are all dependant of each other. Since Adaboost is a boosting ensemble, models do not have equal say in the final model and are weighted based upon their relative importance, with weaker performing models being given more weight. In Adaboost, decision trees are not fully grown. They consist of one root node and two leaf nodes known as stumps. These stumps can only use one variable from the dataset to make a decision as opposed to full decision trees which can use all variables in a dataset. This makes individual decision trees in Adaboost extremely weak learners.

Ideology

The idea behind using stumps as opposed to fully grown trees is that since trees are fitted sequentially in boosting, once a boosting ensemble starts over fitting, it will keep over fitting and cannot go back on itself. Using stumps significantly reduces the likelihood of overfitting since just one variable is used per model. Also, using just one variable per tree means that every line is boosted as it is made.

Process

1. First we assign an equal weightage to each observation in the dataset
2. We then create M decision stumps for M number of features
3. Each of these decision stumps will be trained on the entire dataset
4. New decision stumps will then be built with weights assigned to each observation relative to how well it predicted the outcome, with the observations which gave the worst predictions being assigned larger weightage.
5. This process will continue over and over again

5 Design

In this section I talk about how the project has been set out and the tools used. I start by discussing the software packages. Then move onto how the models were trained and how model performance was optimised. I will then discuss how model performance will be evaluated and finally the data which the best performing model will be tested on.

5.1 Software

In order to carry out the machine learning techniques I will be using Scikit-learn. Scikit-learn is a free software machine learning library for the python programming language. It features various classification, regression and clustering algorithms and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

5.1 Training

Train-test split function

The dataset will be split using the train-test split. Train-test split is a procedure which allows you to simulate how a model would perform on unseen data. It involves splitting the dataset into two subsets. The first subset is used to fit the model and is referred to as the training dataset. For the second subset referred to as the testing subset, the input element is provided to the model, then predictions are made and compared to the actual values. 20% of the data will be assigned to the testing subset with the remaining 80% for the training subset. Given the size of the dataset, a smaller test size could likely be chosen while maintaining a representative test dataset, and allowing the models to train on a larger amount of data. However, due to time taken to train models, the test size will not be reduced further.

Hyperparameters and parameters

Hyperparameters are configurations which are external to the model and whose values cannot be estimated by the data. That is, there values are manually specified and are used to derive the models parameters. The model parameters on the other hand are defined by the model, that is, something that is learnt by the data. The process of training a machine learning model therefore involves choosing the optimal hyperparameters which the algorithm will then use to learn the optimal parameters that correctly input features to the target variable (Nyuytiymbiy, 2020).

Examples of parameters include:

- The mean
- Standard deviation/variance

- Weights associated with the input variables

Although hyperparameters are generally specified manually. Scikit-learn has a set of default hyperparameters, these are selected if the user does not specify the value of hyperparameters. Generally, models are trained with these default hyperparameter values, before attempting to improve the model performance by tuning the hyperparameters. The default hyperparameters for the decision tree regressor are shown below:

```
{'ccp_alpha': 0.0,
 'criterion': 'squared_error',
 'max_depth': None,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'random_state': None,
 'splitter': 'best'}
```

Default hyperparameters for decision tree regressor

Hyperparameter tuning

We generally don't initially know the optimal values of hyperparameters. Therefore, we would like to explore different values in order to find the values which give the best possible model performance. This process is called hyperparameter tuning and can be broken down into the following steps:

1. Define a range of possible values for the hyperparameters
2. Define a method for sampling hyperparameter values
3. Define an evaluative criteria to judge the model

There are many different hyperparameter tuning techniques. This paper explores two such techniques, randomized search and grid search. Both of these techniques allow the user to test the model using a defined range of values for hyperparameters specified. Both techniques also use cross validation.

Cross validation

Generally, we split our data into a training and testing split. If we want to test the model on a larger amount of data, we will need to decrease the size of the training set. However, with less training data, the parameter estimates have greater variance and with less test data, the performance statistics will have greater variance (Anon., n.d.). A way of overcoming this is by using cross-validation. The process of performing cross-validation is given below:

- Split the data into n parts of equal size
- For each iteration the model will be fit on k-1 parts and tested on the final part
- This process will be repeated n times, that is, until each part has been treated as the test set once. For instance, if k=5, the data will be split into five parts and five iterations will be carried out on the data treating each part as the test set once.
- Summarise the model performance using the test scores such as taking the average of the scores

In order to reduce the computational power required and time taken to run the models. A low value of k=3 will be used.

Grid search

Grid search allows us to select the hyperparameters that we are interested in tuning, and selecting a range of possible values for these hyperparameters. Grid search will then test every possible combination. Figure 12 gives an example of two hyperparameters with specified values to test on. If more hyperparameters are chosen, then the grid will be multidimensional. When our dataset is large and the number of hyperparameters we wish to test gets higher, this process of testing every possible combination can get very computationally expensive (Brownlee, 2020). A way to increase the speed of this is to use randomized search.

Hyperparameter 2

Hyperparameter 1	4	5	6	7	8
100					
200					
500					
750					
100					

Figure 12: Example of grid search with two hyperparameters

Randomized search

Randomized search works in a similar way to that of grid search. The main difference is that randomized search does not run for all possible combinations. Rather, it will randomly select combinations from the hyperparameters. The number of combinations is determined by the number of iterations selected, which is simply the number of times we wish to run the model. For instance, if we select the number of iterations to be 10, it will select 10 randomly selected combinations. The drawback of randomized search is that it yields high variance during computing (MALADKAR, 2018).

When deciding which of these two methods to use, there is a trade off to be made between the guarantee of identifying the best possible combination of hyperparameter values and the computation time (Fabien, n.d.). Due to the size of the dataset, the method which will be used for this project will be randomized search.

Despite the randomized search function taking less time to run than the grid search function, it can still take a significant amount more time than the train-test split function. Because of this, I initially carried out randomized search on a sample of the dataset. This will allow me to get an idea of how long randomized search would take if it was to be carried out on the entire dataset of just under two million rows.

To run this test, I will set the number of iterations to just two and cross validation to just three folds to further reduce the time taken to fit the models. Due to the amount of time taken to run the stacking models with the train-test function, it will be excluded from the randomized search.

The results are shown in figure 17:

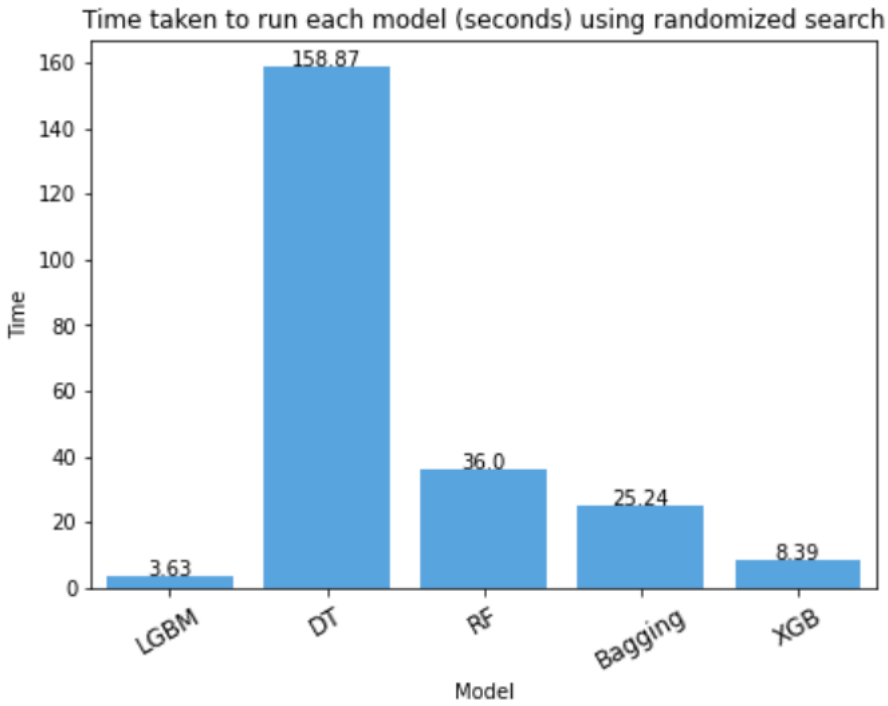


Figure 17: comparison of time taken to run each model using randomized search on 100k rows

Interestingly, the decision tree model has gone from being the fastest most efficient when using train-test split, to being significantly the slowest. The LightGBM model is the fastest at just three seconds.

Since the amount of time taken to fit models appears to increase exponentially as the size of the dataset increases, I will only carry out randomized search using the entire dataset on the LightGBM and XGBoost models. The number of iterations will be five and the number of folds will be kept at three for cross validation in order to increase efficiency and reduce the computational power required. Although reducing the value of K in cross validation generally increases the bias and reduces the variance (yousefnami, 2021), due to the size of the dataset, this shouldn't be too much of an issue and any significance in the amount of bias or variance will likely not be due to the number of folds used in cross validation.

Evaluation metrics

In order to evaluate the predictions made by the models, error metrics will be used. There are a number of error metrics available for evaluating predictions made on regression problems. The four metrics which I will be using are given below:

R^2: Tells us how much of the difference in outcome is explained by the model. E.g., an r-squared score of 0.7 tells us that 70% of differences in the target attribute are explained by the model, while the remaining 30% are due to factor we don't know.

MAE : Is the average absolute difference between the predicted values and the actual values

MSE: Is just like the MAE, but squares the difference before summing them all instead of using the absolute value.

RMSE: The root mean square error is simply the square root of the MSE

Filtering the dataset

It is likely that the relationship between input features and the target attribute property price will have changed over the past 20+ years. Because of this, I will filter the dataset to only include transactions from the year 2010 onwards. Figure 5 shows that the number of transactions dropped in 2008 and 2009, likely due to the financial crisis, before levelling off in 2010. This period from 2010 onwards includes nearly 700,000 transactions which should comfortably be enough for the models to take into account relationships between features and to be able to identify patterns. It is likely that filtering the dataset in this way will increase the error metrics on the training data, but reduce the error metrics when making future predictions, since the model is only fitted on more recent data and therefore won't be taking into account transactions from 20+ years ago.

Fitting model to 2022 data

The best performing model will be fitted to data consisting of future housing transactions. These future transactions will be made up of all observations from January 2022 up until May 2022. This amounts to 8,825 transactions.

6 Implementation

In this section, the plan will be put into effect. Model performance will be evaluated and compared in terms of speed and accuracy for the train-test split function as well as the randomised search function.

Train-test split

Figure 13 shows the mean absolute error (in 1000's) and the R2 score for each of the models using the train-test split function without any hyperparameter tuning being performed. As can clearly be seen, six of the seven models performed very similar on both metrics. Excluding the Adaboost model, the difference between the best and worst performing model is a 1.58 which is a negligible £1,580. The R2 scores are just as similar, once again ignoring the Adaboost model, all have a score of either 0.69 or 0.7.

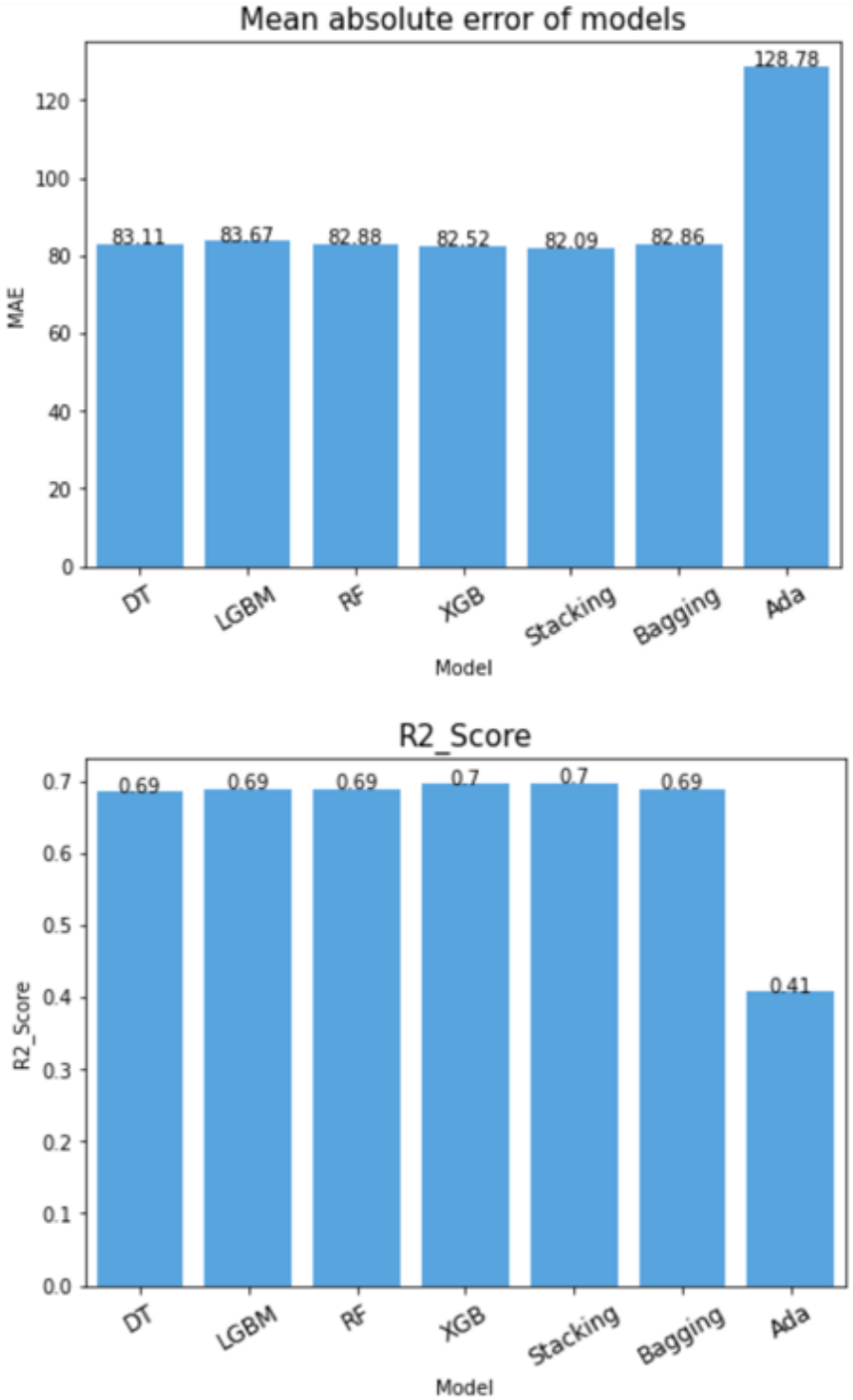


Figure 13: Comparison of mean absolute error and R2 score for each model

Figure 14 gives a comparison of the speed taken to fit each model. As the figure clearly shows, there is a relatively large amount of variation between all of the models. Despite this variation, all but two models took less than a minute to run. The stacking model took a very significant of time longer to run than any other model, at 1,283 seconds or just over 23 minutes. The fastest model, decision tree, took just over two seconds to run with is almost 614 times faster than the stacking model.

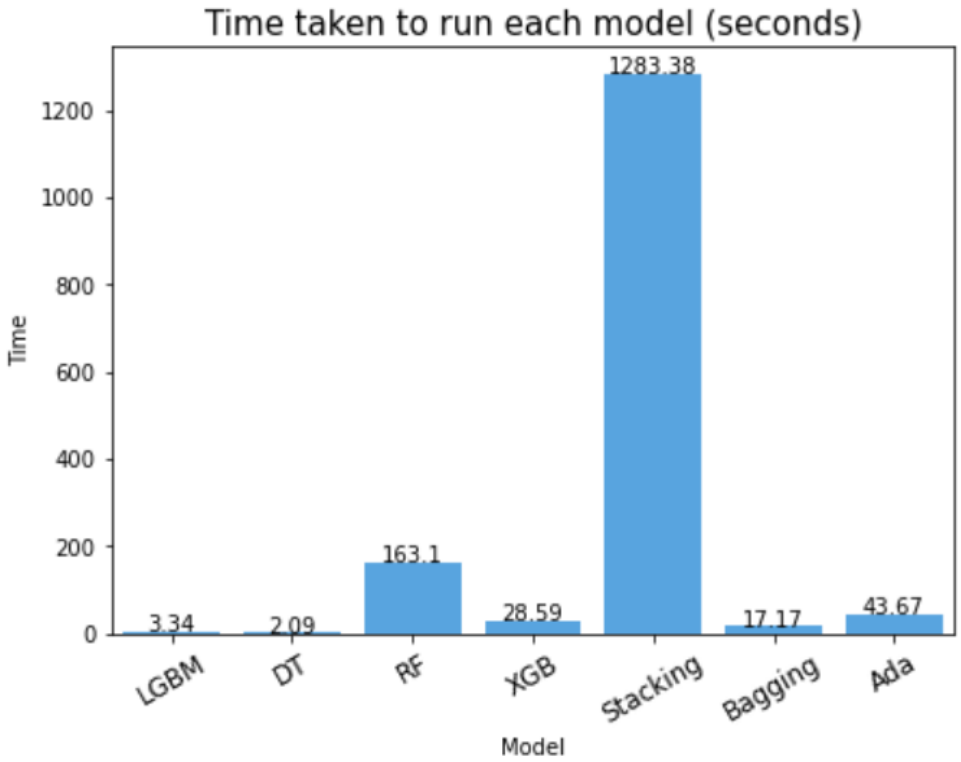


Figure 14: Comparison of the speed taken to run each model using train-test split function

Randomized search

Figure 15 shows the mean absolute error and the R2 score for the LightGBM and XGBoost models using the randomized search function. When comparing the results with the results of the train-test split models, interestingly, LightGBM has performed almost identical, with a mean absolute error of £83,000 and R2 score of 0.69. The performance of XGBoost on the other hand has reduced significantly with a mean absolute error of £132,000 and R2 score of 0.49. When using cross validation, the R2 score was consistently under 0.5, achieving a mean score of 0.47 with K=5.

```
In [374]: score_xgb = cross_val_score(xgb_tuned_full, X,y, cv=5)
          score_xgb

Out[374]: array([0.47701409, 0.47582994, 0.47487147, 0.47177252, 0.47188303])
```

```
In [375]: print("Mean cross-validataion score: %.2f" % score_xgb.mean())

          Mean cross-validataion score: 0.47
```

Filtering dataset from 2010 onwards

As expected, once filtering the dataset to only consist of transactions from 2010 onwards has reduced the fit of the model. The mean absolute error has increased to £135,000 while the R2 score has reduced to 0.47.

```
In [426]: mae_lgbm_rs_2010 = mean_absolute_error(y_test, pred_lgbm_2010)
          mse_lgbm_rs_2010 = mean_squared_error(y_test, pred_lgbm_2010)
          rmse_lgbm_rs_2010 = np.sqrt(mse_lgbm_rs_2010)
          r2_lgbm_rs_2010 = r2_score(y_test, pred_lgbm_2010)
          print('Mean absolute error:     {}'.format(mae_lgbm_rs_2010))
          print('Mean squared error:      {}'.format(mse_lgbm_rs_2010))
          print('Root mean squared error: {}'.format(rmse_lgbm_rs_2010))
          print('R2 score:                {}'.format(r2_lgbm_rs_2010))

          Mean absolute error:     135.09812945878846
          Mean squared error:      32972.79402251164
          Root mean squared error: 181.58412381734158
          R2 score:                0.46756606687846936
```

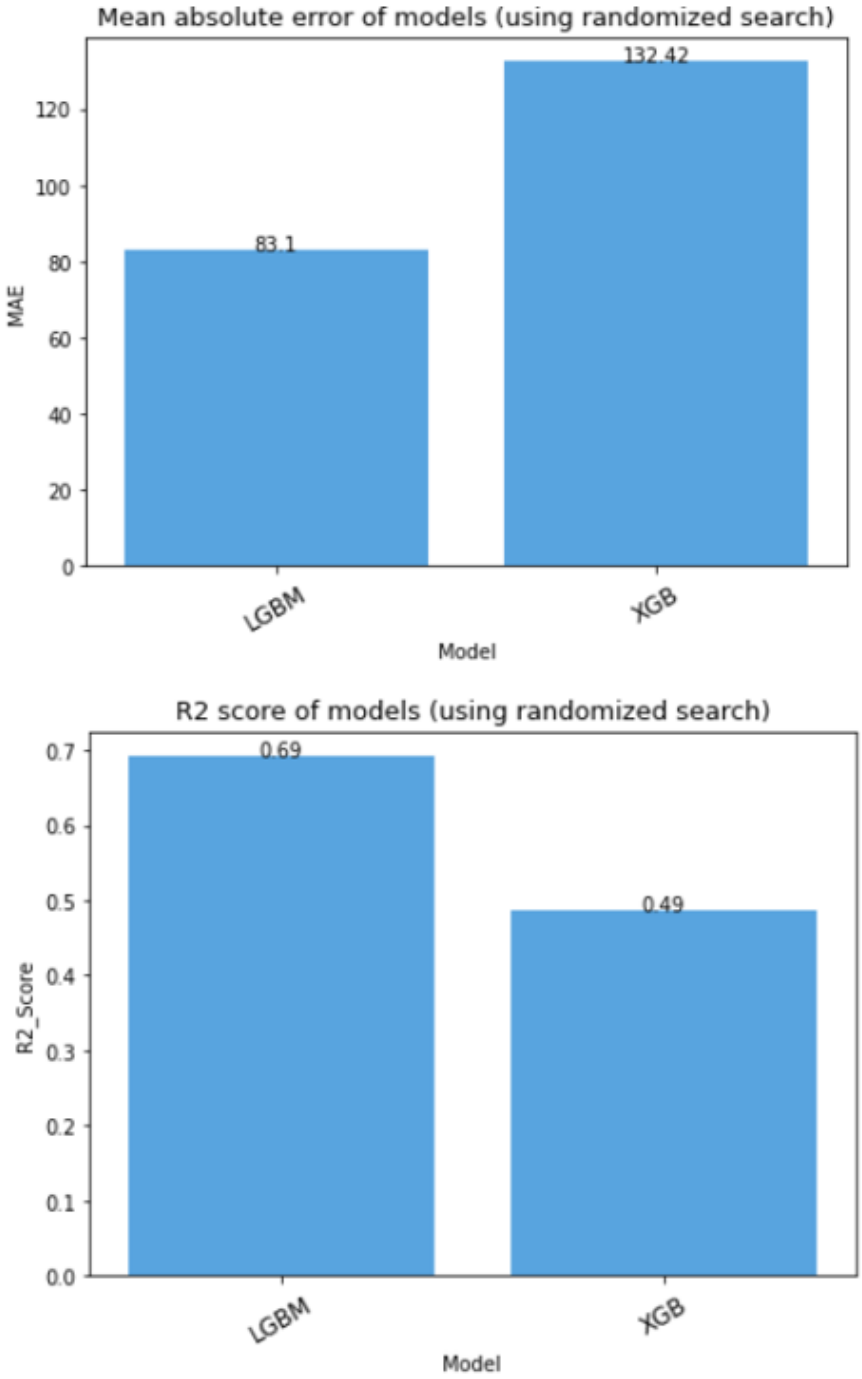


Figure 15: Comparison of the speed taken to run each model using randomized search

7 Results

This section presents the results of the best performing model on predicting future transactions trained on data from 1995-2021 as well as trained on data from 2010 – 2021. Of the model which performed better, the performance will be analysed further in terms of the number of predictions within 10% and 20% of the actual value, the number of overpredictions and underpredictions. The section will finish by observing the relation between predicted values and actual values, before giving reasons why I believe the model didn't perform as well on future predictions than was hoped.

Using model trained on transactions from 1995-2021

For the best fitting model containing all transactions from 1995, the mean absolute error has shot up to £265,000. This is more than three times the value which the model obtained when testing its performance on previous transactions. while the error was almost certainly going to increase when predicting future transactions, one would of hoped that this increase would be less substantial.

```
In [663]: mae_actual = mean_absolute_error(y_2022, pred_2022)
          print('Mean absolute error:     {}'.format(mae_actual))

          Mean absolute error:     264.9886102204842
```

Using model trained on transactions from 2010-2021

The performance is only marginally better when using the model fitted on transactions from 2010 to 2021 rather than from 1995 to 2021. The mean absolute error is now £246,000, which is around twice the value obtained when testing the models performance on previous transactions of £135,000.

Amount within 10% and 20%

Figure 17 shows the first five rows of a table which gives a breakdown of the difference between the predicted price and the actual price for all 8,825 observations. It also gives the squared difference, absolute difference and percentage difference. This will allow us to gain a deeper insight into predictions made by the model. The first thing which sticks out is that the first three predictions made by the model are identical at 682.837416, interestingly, the actual values for each of these three observations differ considerably. Secondly, we can see that four out of the first five predictions, the model has overestimated the transaction value. This can be seen by observing the 'Difference' column, where four of the values have a negative sign in front of them.

	Predicted	Actual	Difference	Squared_difference	Absolute_difference	Percentage_difference
0	682.837416	182.5	-500.337416	250337.529847	500.337416	2.741575
1	682.837416	595.0	-87.837416	7715.411649	87.837416	0.147626
2	682.837416	320.0	-362.837416	131650.990448	362.837416	1.133867
3	669.994890	785.0	115.005110	13226.175311	115.005110	0.146503
4	753.515804	420.0	-333.515804	111232.791603	333.515804	0.794085

Figure 17: The first five rows of a table showing differences between predicted values and actual values for each observation

In total, 6,302 predictions were overestimated, while 2,523 were underestimated. The model is therefore around 2.5 times more likely to overestimate the selling price of a transaction than underestimate it.

```
In [697]: over_estimating = dataframe.loc[dataframe['Difference'] < 0]
          under_estimating = dataframe.loc[dataframe['Difference'] > 0]
          print('Overestimating the value:', over_estimating.shape[0])
          print('Underestimating the value', under_estimating.shape[0])

          Overestimating the value: 6302
          Underestimating the value 2523
```

On average, the model overestimated the selling price by £107,000.

```
In [682]: dataframe.Difference.mean()

Out[682]: -107.21677078558615
```

The number of predictions within 10% of the actual price is 1,305, while the number within 20% is 2,483. This equates to 14.8% and 28.1%.

```
In [707]: within_10 = dataframe.loc[dataframe['Percentage_difference'] <= 0.1]
          within_20 = dataframe.loc[dataframe['Percentage_difference'] <= 0.2]
          print('Number of predictions within 10% of the actual selling price:', within_10.shape[0])
          print('Number of predictions within 20% of the actual selling price:', within_20.shape[0])

          Number of predictions within 10% of the actual selling price: 1305
          Number of predictions within 20% of the actual selling price: 2483
```

The average error made by the model was 56%.

```
In [714]: dataframe.Percentage_difference.mean()

Out[714]: 0.5617389011492961
```

figure 18 identifies a number of insights about the relationship between predicted values and actual values. Firstly, a strange pattern of predicted prices on the vertical axis can be seen. It appears that the model has classified the majority of observations into one of a small number of values, as can be seen by the horizontal lines on the scatter. Secondly, we can see that the model has hugely overestimated the predictions of lower transactions, and massively undervalued predictions of higher transactions, while average selling prices of £550,000 to £650,000 have been predicted relavively well. This tells us that the model hasn't taken into account the very large amount of variation in the target attribute.

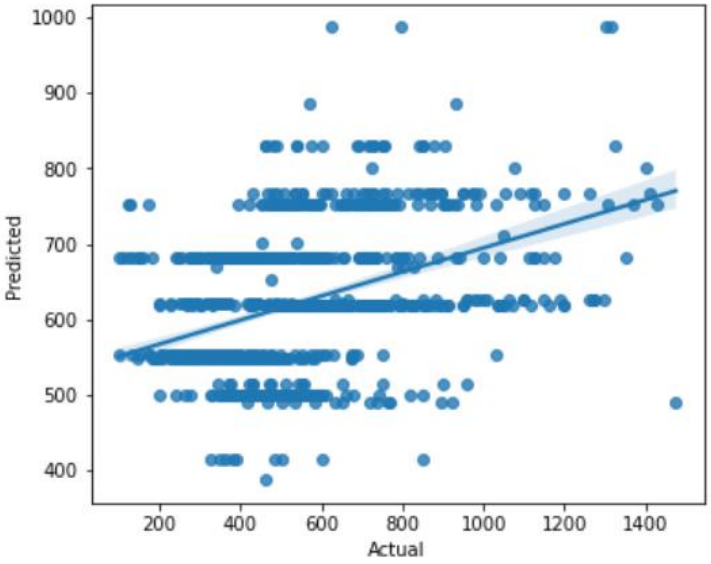


Figure 18: Scatterplot of random sample of 1,000 observations comparing the predicted price with the actual price

Figure 19 gives the spread of prices paid from 2010 until 2022 for a random sample of 1000 observations. Values on the horizontal axis from 324 onwards represent transactions from 2022. We can see that the spread has once again increased, with transactions ranging from around £180,000 up to around £1.5m for a small sample size of 1,000 transactions.

Figure 19: Random sample of 1,000 observations showing the distribution over time from 2010 until 2022

7.1 Summary

In summary, I believe the main reason for the increase in error on future observations than on previous observations is due to the positive skew of the distribution of the target attribute (even after removing some of the extreme values) which has caused the model to overestimate three times as many observations as it underestimated. Also, the variability of the target attribute increasing in 2022 compared to all previous years.

8 Conclusion

We have built a machine learning model which can predict future London property prices with a relative level of accuracy for transactions which are in the mid-range of sale prices.

The size of the dataset did provide some limitations. These included the following. Firstly, representing the location of property by longitude and latitude was too computationally expensive with around two million inputs. Secondly, when carrying out hyperparameter tuning, the number of folds for cross validation and the number of iterations needed to be kept low in order to keep the run time to a manageable amount of time. The model struggled to take into account the amount of variability of the target attribute, the main reason for this is likely due to the dataset not containing any information regarding the size of property. In order to reduce this variability, the most extreme values, those which exceeded equation (2.1), were removed for each year. The dataset was filtered to only include transactions which from 2010 onwards so that only transactions which were more relevant to future transactions were considered. The accuracy of the models used were relatively similar, with the exception of the Ada model. However, the models varied significantly in terms of speed. Hyperparameter tuning had minimal effect on accuracy, a likely reason for this to keeping the number of iterations very low, yet had a major effect on speed. The skewness of the distribution meant that the model was overestimating the value of around two thirds of properties. As shown figures 5 and 18.

In order for this model to be able to predict the majority of values with a high level of accuracy, information relating to size of property would need to be added.

8.1 Future work

I have identified a number of things which could be done which would potentially improve the accuracy of predictions:

- **Gathering more information:** While the HM Land Registry prices paid dataset contain a large number of instances. The number of features was rather limited. As mentioned, gathering information on size of property would almost certainly improve predictions. Other information worth considering would include availability of off-street parking, distance from public transport such as the London underground, and information relating to the wider economy such as the population increase relative to the number of net additional dwellings.
- **Giving predictions in the from of intervals rather than point estimates**: This would likely take into account more of the variability if it was not possible to obtain information about the size of property.
- **Consider other ways to represent the features:** different ways to encode the categorical features could be considered. Finding a way to represent location by longitude and latitude which isn't too computationally expensive could help to reduce the variability, since prices in a district can differ significantly.
- **Consider other types of models:** models such as neural networks or nearest neighbours could be considered in addition to tree-based models.

Bibliography

Anon., 2016. [Online]
Available at: https://www.analyticsvidhya.com/blog/2016/04/tree-based-algorithms-complete-tutorial-scratch-in-python/
[Accessed 11 06 2022].

Anon., 2018. *Nearly 100,000 properties in England and Wales owned by foreign entities.* [Online]
Available at: https://www.step.org/industry-news/nearly-100000-properties-england-and-wales-owned-foreign-entities
[Accessed 17 05 2022].

Anon., 2019. *London at the heart of real UK house price falls.* [Online]
Available at: https://pearsonblog.campaignserver.co.uk/tag/real-and-nominal-house-prices/
[Accessed 02 07 2022].

Anon., 2019. *Why XGBoost ? and Why is it so Powerful in Machine Learning.* [Online]
Available at: https://abzooba.com/resources/blogs/why-xgboost-and-why-is-it-so-powerful-in-machine-learning/#:~:text=XGBOOST%20%E2%80%93%20Why%20is%20it%20so,computation%20on%20a%20single%20machine.
[Accessed 05 06 2022].

Anon., 2020. *How does the housing market affect the economy?.* [Online]
Available at: https://www.bankofengland.co.uk/knowledgebank/how-does-the-housing-market-affect-the-economy#:~:text=Normally%20that%20happens%20when%20the,house%20and%20prices%20will%20rise.
[Accessed 08 07 2022].

Anon., 2020. *How machine learning works.* [Online]
Available at: https://www.datarobot.com/blog/how-machine-learning-works/#:~:text=How%20does%20machine%20learning%20work,and%20involves%20minimal%20human%20intervention.
[Accessed 19 04 2022].

Anon., 2020. *What is the housing market?,* London: The Bank Of England.

Anon., 2021. *Bagging.* [Online]
Available at: https://www.ibm.com/cloud/learn/bagging
[Accessed 27 07 2022].

Anon., 2022. *An In-depth Guide to SkLearn Decision Trees.* [Online]
Available at: https://www.simplilearn.com/tutorials/scikit-learn-tutorial/sklearn-decision-trees
[Accessed 18 05 2022].

Anon., 2022. *Gross domestic product of the United Kingdom from 1948 to 2021.* [Online]
Available at: https://www.statista.com/statistics/281744/gdp-of-the-united-kingdom/
[Accessed 22 04 2022].

Anon., 2022. *Home Ownership Statistics 2022.* [Online]
Available at: https://www.birdandco.co.uk/site/blog/conveyancing-blog/midlands-has-the-highest-home-ownership-rate-in-england
[Accessed 19 06 2022].

Anon., 2022. *London, UK Metro Area Population 1950-2022.* [Online]
Available at: https://www.macrotrends.net/cities/22860/london/population#:~:text=The%20current%20metro%20area%20population,a%201.38%25%20increase%20from%202019.
[Accessed 22 04 2022].

Anon., 2022. *Top 10 Machine Learning Applications and Examples in 2022.* [Online]
Available at: https://www.simplilearn.com/tutorials/machine-learning-tutorial/machine-learning-applications
[Accessed 09 05 2022].

Anon., 2022. *UK House Price Index.* [Online]
Available at: https://www.ons.gov.uk/economy/inflationandpriceindices/bulletins/housepriceindex/january2022#:~:text=Despite%20being%20the%20region%20with,house%20price%20at%20%C2%A3151%2C000
[Accessed 19 03 22].

Anon., n.d. [Online]
Available at: https://www.nosimpler.me/cgi-sys/suspendedpage.cgi

Anon., n.d. *1.10. Decision Trees.* [Online]
Available at: https://scikit-learn.org/stable/modules/tree.html
[Accessed 14 06 2022].

Anon., n.d. *Cache (computing).* [Online]
Available at: https://en.wikipedia.org/wiki/Cache_(computing)
[Accessed 18 06 2022].

Anon., n.d. *Decision tree learning.* [Online]
Available at: https://en.wikipedia.org/wiki/Decision_tree_learning#Advantages
[Accessed 02 07 2022].

Anon., n.d. *Ensemble learning.* [Online]
Available at: https://en.wikipedia.org/wiki/Ensemble_learning#Ensemble_theory
[Accessed 08 07 2022].

Anon., n.d. *Financial crisis of 2007–2008.* [Online]
Available at: https://en.wikipedia.org/wiki/Financial_crisis_of_2007%E2%80%932008
[Accessed 04 06 2022].

Anon., n.d. *image recognition.* [Online]
Available at: https://www.techtarget.com/searchenterpriseai/definition/image-recognition
[Accessed 06 21 2022].

Anon., n.d. *Is there a rule-of-thumb for how to divide a dataset into training and validation sets?.* [Online]
Available at: https://stackoverflow.com/questions/13610074/is-there-a-rule-of-thumb-for-how-to-

divide-a-dataset-into-training-and-validatio
[Accessed 18 07 2022].

Anon., n.d. *Machine Learning.* [Online]
Available at: https://www.sas.com/en_gb/insights/analytics/machine-learning.html
[Accessed 29 05 2022].

Anon., n.d. *Machine learning is used in a range of everyday situations. But do you know where?.* [Online]
Available at: https://royalsociety.org/topics-policy/projects/machine-learning/machine-learning-in-the-world-around-you-infographic/#:~:text=Machine%20learning%20is%20all%20around,needed%20to%20solve%20a%20problem.
[Accessed 07 08 2022].

Anon., n.d. *parallel processing.* [Online]
Available at: https://www.techtarget.com/searchdatacenter/definition/parallel-processing#:~:text=Parallel%20processing%20is%20a%20method,time%20to%20run%20a%20program.
[Accessed 13 06 2022].

Anon., n.d. *Stacking in Machine Learning.* [Online]
Available at: https://www.javatpoint.com/stacking-in-machine-learning
[Accessed 23 07 2022].

Anon., n.d. *Virtual Assistant.* [Online]
Available at: https://en.wikipedia.org/wiki/Virtual_assistant
[Accessed 15 05 2022].

Anon., n.d. *XGBoost.* [Online]
Available at: https://www.nvidia.com/en-us/glossary/data-science/xgboost/
[Accessed 17 06 2022].

Brownlee, J., 2016. *A Gentle Introduction to XGBoost for Applied Machine Learning.* [Online]
Available at: https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/
[Accessed 11 06 2022].

Brownlee, J., 2018. *How to Avoid Overfitting in Deep Learning Neural Networks.* [Online]
Available at: https://machinelearningmastery.com/introduction-to-regularization-to-reduce-overfitting-and-improve-generalization-error/
[Accessed 05 07 2022].

Brownlee, J., 2020. *Hyperparameter Optimization With Random Search and Grid Search.* [Online]
Available at: https://machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search/
[Accessed 10 08 2022].

Brownlee, J., 2020. *Stacking Ensemble Machine Learning With Python.* [Online]
Available at: https://machinelearningmastery.com/stacking-ensemble-machine-learning-with-python/
[Accessed 18 07 2022].

Chatterjee, S., 2019. *A Deep Dive into Decision Trees.* [Online]
Available at: https://medium.com/analytics-vidhya/a-deep-dive-to-decision-trees-6575e016d656
[Accessed 08 06 2022].

Cook, L., 2020. *The house price gap: tracking 50 years of London v the rest of the UK.* [Online]
Available at: https://www.savills.co.uk/blog/article/298109/residential-property/the-house-price-gap--tracking-50-years-of-london-v-the-rest-of-the-uk.aspx
[Accessed 07 06 2022].

Fabien, M., n.d. *Grid Search vs. Randomized Search.* [Online]
Available at: https://maelfabien.github.io/machinelearning/GridRand/#randomized-search
[Accessed 07 08 2022].

G, Y., 2017. *What is Machine Learning?.* [Online]
Available at: https://towardsdatascience.com/what-is-machine-learning-8c6871016736
[Accessed 07 05 2022].

K, D., 2019. *Top 5 advantages and disadvantages of Decision Tree Algorithm.* [Online]
Available at: https://dhirajkumarblog.medium.com/top-5-advantages-and-disadvantages-of-decision-tree-algorithm-428ebd199d9a
[Accessed 06 06 2022].

Kumar, N., 2019. *Advantages and Disadvantages of Decision Trees in Machine Learning.* [Online]
Available at: http://theprofessionalspoint.blogspot.com/2019/02/advantages-and-disadvantages-of.html
[Accessed 12 05 2022].

LENDAVE, V., 2021. *A Tutorial on Sequential Machine Learning.* [Online]
Available at: https://analyticsindiamag.com/a-tutorial-on-sequential-machine-learning/
[Accessed 17 08 2022].

Mahajan, C., 2021. *Machine learning.* [Online]
Available at: https://mechomotive.com/machine-learning-3/
[Accessed 17 06 2022].

MALADKAR, K., 2018. *Why Is Random Search Better Than Grid Search For Machine Learning.* [Online]
Available at: https://analyticsindiamag.com/why-is-random-search-better-than-grid-search-for-machine-learning/
[Accessed 15 08 2022].

Marr, B., 2018. *How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read.* [Online]
Available at: https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/
[Accessed 12 04 2022].

Massaoudi, T., 2020. *What You Should Know About Ensemble Learning.* [Online]
Available at: https://towardsdatascience.com/what-you-should-know-about-ensemble-learning-e92d4b3c3608#:~:text=Summary,model%20and%20can%20generalise%20better.
[Accessed 04 06 2022].

NG, A., 2015. *Machine Learning for a London.* s.l.:Aaron NG.

Nyuytiymbiy, K., 2020. *Parameters and Hyperparameters in Machine Learning and Deep Learning.* [Online]
Available at: https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac
[Accessed 11 06 2022].

Perry, S., 2020. *My London/News/Property.* [Online]
Available at: https://www.mylondon.news/news/property/london-house-prices-new-graph-19187261
[Accessed 17 04 2022].

Richmond, S., 2016. *Algorithms Exposed: Random Forest.* [Online]
Available at: https://bccvl.org.au/algorithms-exposed-random-forest/#:~:text=ASSUMPTIONS,are%20ordinal%20or%20non%2Dordinal.
[Accessed 21 07 2022].

Saini, A., 2021. *Gradient Boosting Algorithm: A Complete Guide for Beginners.* [Online]
Available at: https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/
[Accessed 09 07 2022].

Singh, A., 2018. *A Comprehensive Guide to Ensemble Learning (with Python codes).* [Online]
Available at: https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/
[Accessed 26 05 2022].

Team, G. B., n.d. *8 Key Factors Influencing Property Price In The UK.* [Online]
Available at: https://www.gsbrown-construction.co.uk/blogs/8-key-factors-influencing-property-price-in-the-uk/
[Accessed 28 04 2022].

Telfor, M., 2019. *Registering land for more than 150 years.* [Online]
Available at: https://hmlandregistry.blog.gov.uk/2019/05/30/registering-land-for-more-than-150-years/
[Accessed 05 04 2022].

Tripathi, M., 2020. *Knowing all about Outliers in Machine Learning.* [Online]
Available at: https://datascience.foundation/sciencewhitepaper/knowing-all-about-outliers-in-machine-learning
[Accessed 09 05 2022].

Vlahovljak, A., 2022. *House price prediction in the UK.* s.l.:s.n.

yousefnami, 2021. *Why Does Increasing k Decrease Variance in kNN?.* [Online]
Available at: https://towardsdatascience.com/why-does-increasing-k-decrease-variance-in-knn-9ed6de2f5061
[Accessed 05 08 2022].