UPPSALA
UNIVERSITET

Improving House Price Prediction Models: Exploring the Impact of Macroeconomic Features

Max Hansson and Martin Holmqvist

Bachelor's thesis in Statistics

Advisor
Patrik Andersson

2023

Abstract

This thesis investigates if house price prediction models perform better when adding macroeconomic features to a data set with only house-specific features. Previous research has shown that tree-based models perform well when predicting house prices, especially the algorithms random forest and XGBoost. It is common to rely entirely on house-specific features when training these models. However, studies show that macroeconomic variables such as interest rate, inflation, and GDP affect house prices. Therefore it makes sense to include them in these models and study if they outperform the more traditional models with only house-specific features. The thesis also investigates which algorithm, out of random forest and XGBoost is better at predicting house prices. The results show that the mean absolute error is lower for the XGBoost and random forest models trained on data with macroeconomic features. Furthermore, XGBoost outperformed random forest regardless of the set of features. In Conclusion, the suggestion is to include macroeconomic features and use the XGBoost algorithm when predicting house prices.

Contents

1 Introduction

Accurate house price predictions have gained much interest in the data science community due to the growing popularity of machine learning algorithms and their predictive power (Irizarry, 2019). Tree-based algorithms, like random forest and XGBoost, regularly outperform traditional prediction methods on complex data (Gareth et al., 2013). Accurate house price predictions are crucial for several relevant groups, including investors, homeowners, and academics. For example, the prediction model could help investors find profitable apartments or homeowners get accurate and automated valuations. Additionally, the model could aid in detecting macroeconomic trends such as price bubbles or recessions.

Earlier research has been done in the field of trying to predict house prices. Ekberg and Johansson (2022) compare different machine learning techniques regarding their ability to predict house prices in Stockholm, where they only include house-specific features such as location and living area. They find that random forest outperforms the models K-nearest neighbors and neural networks. W.Revend (2020) instead found that XGBoost performs better than random forest when predicting house prices using house-specific features. However, training such models leads to a major flaw; house-specific features cannot consider changes in the macroeconomic environment over time.

Recently house prices have increased drastically. For example, in Stockholm, the value has risen more than 300% in the last 20 years (Öljemark, 2023). Getahun (2011) found that 92% of the house price index variation can be explained by changes in interest rates, net household income, inflation rate, and supply. Apergis (2003) also writes in his article that macroeconomic variables, such as interest rate, inflation, and unemployment rates, impact the housing market. Bohlin and Kellgren (2021) conclude that inflation has a long-term positive predictive relationship with housing prices. Xu (2017) writes that GDP, housing prices, interest rate, and income have a complex interdependent relationship. Thus, the research implies that macroeconomic features have a strong predictive relationship with house prices. Hence, not including them in house price prediction models could be sub-optimal.

Decision tree based algorithms such as random forest and XGBoost have outperformed other algorithms when predicting house prices which is why they will be studied and compared further in this thesis. Additionally, the most intuitive method for analyzing the results of these models is looking at the mean absolute error, as this gives a clear and easily interpreted error metric (Willmott and Matsuura, 2005). To be able to compare the performance of different algorithms a statistical test such as a paired t-test is suitable.

The purpose of this thesis is to investigate if a set of macroeconomic features significantly improves tree-based prediction models for house prices. It looks to broaden and build on previous research. Further, another objective will be to compare the chosen methods by evaluating their performance using house-specific and macroeconomic features.

Therefore, the research questions that will be answered are the following:

- *Will the addition of macroeconomic features significantly improve the performance of house price prediction models?*
- *Is random forest or XGBoost better at predicting house prices?*

The prediction area will be limited to include apartments and houses in the Stockholm municipality. The model will be optimized using feature selection, feature engineering, and tuning methods. We will compare the models' performance with and without macroeconomic features using various metrics, including mean absolute error and mean absolute percentage error. This thesis has limited the scope of the research to the macroeconomic variables interest rate, GDP, house price index, CPI, and unemployment rate. These variables are stated to have explanatory power to house prices. Further, these variables are publicly accessible through trusted government sources. Due to computational limitations, the thesis only tests the difference in performance between random forest and XGBoost.

The thesis is organized as follows. In Section 2 we provide an overview of the methods used to fulfill the purpose and address the research questions. In Section 3 we present the data, features and training process of the prediction models. Section 4 presents and explain the meaning of the obtained results and Section 5 provides a discussion of these. Finally, Section 6 provides a conclusion of the thesis, summarizing the key findings.

2 Method

This section presents the methods used to answer the research questions.

2.1 Decision trees

All models that will be constructed in this thesis are based on decision trees. They are a class of non-linear machine learning methods. They can be used for both classification and regression tasks. A decision tree is built from a root node that splits the data based on some rule. Take Figure 1, for example; if $z < 4$, the observation will go to the left leaf; if not, it will go to the right. This way, the decision tree keeps splitting the observations into new leaves through multiple rules. A tree can split up the data until there is only one observation in each leaf, leading to zero in-sample error, which means the prediction is the same as the real class or value. This will lead to an overfit model and increase the out-of-sample error since the model is adjusted to classify the in-sample data perfectly. Hence, a more generalized model is often preferable to minimize the out-of-sample error (Gareth et al., 2013). For regression decision trees, the aim to find the best variable j with the best splitting point t that reduces the error metric the most in both leaves R_1 and R_2 as in (1).

$$j^*, t^* = \text{argmin}_{j,t} \left[\frac{1}{n} \sum_{\mathbf{x}_i \in R_1(j,t)} (y_i - p_{iR_1})^2 + \frac{1}{n} \sum_{\mathbf{x}_i \in R_2(j,t)} (y_i - p_{iR_2})^2 \right] \tag{1}$$

.

The variable that reduces the error metric the most at the current node is then used since it separates the data in the best way. This is how regression decision trees are built (Gareth et al., 2013).

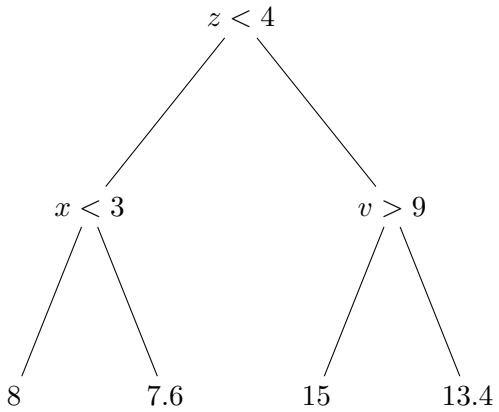

Figure 1: example of a decision tree.

2.2 Ensemble methods

Ensemble methods are machine learning techniques that aim to produce several learners, also known as weak learners, that are then combined to create a united prediction. This can drastically increase the predictive performance and is different from traditional models where one learner is optimized, as opposed to many weak learners being merged. Various methods, such as neural networks or decision trees, can produce weak learners. Ensemble learning is more effective at decreasing the generalization error than weak learners since, for example, one decision tree will often suffer from overfitting. Most ensemble methods fall into one of two types of ensemble learning methods, sequential and parallel. In sequential ensemble methods, the models are created individually and learn from the previous model in some manner. In parallel ensemble methods, the weak learners are made not in a sequence but parallel to each other, meaning decision tree n does not learn anything from tree $n-1$ (Zhou, 2012).

2.3 Feature selection

When constructing tree-based models, finding good features is one of the most meaningful improvements to the models' prediction power. This is known as feature selection, where a feature has the same meaning as a covariate in traditional models. Before training a model, it is important to include features that will improve its performance and help predict the dependent variable (D´Agostino, 2022). In the early years of machine learning, the computational power was not as strong as it is now, meaning it was necessary only to use the essential features for the computer to be able to train the models. In later years, the increase in computational power has partially solved the problem of having too many features in a data set. However, using unimportant features that do not improve the prediction accuracy will cause random noise that ultimately will lead to a flawed and misleading model that could be prone to overfitting (Guyon and Elisseeff, 2003).

2.4 Boosting

Boosting is a set of algorithm that employ sequential ensemble strategies to combine several weak models into a stronger and more reliable prediction model. The idea of boosting is to create a sequence of weak learners that, for each iteration, learns from the mistakes of previous weak learners. Weak learners are slightly better at predicting the dependent variable than random guessing; however, in an ensemble, they become strong learners with a drastically increased predictive performance (Zhou, 2012).

2.5 Gradient descent

Gradient descent is a numerical optimization algorithm used for minimizing a loss function. The method is especially useful when the analytical solution is unavailable or computationally expensive. The algorithm works by iteratively adjusting model parameters in the direction of the steepest descent. This way, the algorithm will find a local minimum in the loss function, thus realizing the optimal parameter values. A key hyperparameter in the algorithm is the learning rate, which controls how much the algorithm learns from each gradient update. A small learning rate will cause the algorithm to find the minimum point slowly, while a large learning rate may cause it to overshoot the minimum point of the function (Ruder, 2016).

2.6 Gradient boosting

Gradient boosting aims to minimize the loss function for some sample data $(x_i, y_i)_{i=1}^n$. The algorithm uses *gradient descent* to minimize the loss. The algorithm minimizes the loss function, as in (2).

$$\hat{F}(x_i) = \arg\min_{F(x_i)} \sum_{i=1}^{n} \mathcal{L}(y_i, F(x_i)). \tag{2}$$

Here, y_i is the observed value, and $\hat{F}(x_i)$ is the model made by combining the weak learners. A loss function measures the accuracy of a model's predictions for the dependent variable. Thus, the algorithm aims to minimize some given error metric between predicted and observed values. In this thesis, the error metric that is being minimized is the mean squared error:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - p_i)^2, \tag{3}$$

where y_i is the observed values and p_i is the predicted values.

The algorithm calculates the residuals, which are defined as

$$\epsilon_{i,m} = -\frac{\partial L(y_i, F(\boldsymbol{x}_i))}{\partial F(\boldsymbol{x}i)} \Bigg|_{F(\boldsymbol{x}_i)=F_{m-1}(\boldsymbol{x}_i).} \tag{4}$$

Here, $\epsilon_{i.m}$ is the residual of sample number i and weak learner m. Thus, $\epsilon_{1,1}$ would be the residual for the first weak learner's first prediction.

After calculating the residuals, the algorithm fits a new weak learner to the residuals using the same loss function. It creates terminal regions $R_{j,m}$ for $j = 1...J_m$ where the algorithm once more minimizes the same loss function and finds the optimal output value for all leaves

with more than one observation. This is done as in (5):

$$O_{j,m} = \arg\min_{O} \sum_{x_i \in R_{jm}} \mathcal{L}(y_i, F_{m-1}(x_i) + O), \tag{5}$$

where $O_{j,m}$ is the output value for terminal region j and tree m.

Then, using the updated tree, the new model, $F_m(x)$, is

$$F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} O_{J,m} I(x \in R_{j,m}). \tag{6}$$

Here ν is the shrinkage rate, determining how much the model will learn from previous weak learners (Chen and Guestrin, 2016). $I(x \in R_{j,m})$ is an indicator function that returns 1 if x is in the j:th leaf node and 0 if it is not. Thus, the Macro model is the last model + the learning rate $\times$ the new output values.

2.7 XGBoost algorithm

This is a summary of some of the ways the XGBoost algorithm differs from the gradient boost algorithm. In the original paper about XGBoost, all the steps and calculations have been shown in detail (Chen and Guestrin, 2016). The Extreme Gradient Boosting algorithm, or XGBoost, provides an optimized model known to perform well on various tasks and data sets (Fernández-Delgado et al., 2014). It is based on the gradient boosting algorithm, and its greatest advantages include great scalability and speed compared to other gradient boosting techniques. For example, when the algorithm was introduced in 2016, it was at least ten times faster than any other popular existing algorithm (Chen and Guestrin, 2016).

Like gradient boosting, the XGBoost algorithm starts with an initial prediction and then fits a regression tree to the residuals of that prediction. The loss function in the XGBoost model is

$$\sum_{i=1}^{n} \mathcal{L}(y_i, F(x_i)) + \gamma T + \frac{1}{2}\lambda O^2. \tag{7}$$

Here, γT represents that complex trees with many terminal nodes are penalized by the parameter γ by pruning all splits with less gain than γ. This is explained further in (9). O^2 is the square output value, which is the value assigned to each leaf node in a decision tree. The output value is calculated as in (10). λ is another regularization parameter that will prevent overfitting by shrinking the residuals and thereby lowering their influence in the model. The large output values will be penalized more than the smaller ones, reducing the generalization

error.
How the regression trees are built differs from earlier gradient-boosting techniques. Unlike gradient boosting, the algorithm calculates a similarity score for the residuals in each split of the regression tree. The similarity score is defined by

$$Score = \frac{(\sum_{i=1}^{n}(y_i - p_i))^2}{n + \lambda} \tag{8}$$

Where n is the number of residuals and λ is a regularization parameter set manually. In (8), because the sum of the residuals is taken before the square, residuals with differing signs will cancel each other out. This means that if the numerator consists of positive and negative values of similar magnitude, it will lead to a low similarity score. However, if the residuals are all positive or negative, the similarity score will be large, and thus the split will be considered good. The algorithm will greedily use the split that minimizes the loss function at every step in the tree, then evaluate the split using the similarity score (Chen and Guestrin, 2016). After calculating the similarity score, the *gain* will be calculated as in

$$Gain = Score_{Left} + Score_{Right} - Score_{Root} \tag{9}$$

If the similarity score is an evaluation of the split, the gain can be seen as a way of measuring the influence each feature has on the accuracy of the model (Géron, 2022). The model will remove branches with less gain than γ. This regularization parameter controls the pruning of the regression tree. This way, the algorithm only uses the most valuable features and avoids doing splits that do not help the model predict the dependent variable. The output value is then calculated as

$$O = \frac{\sum_{i=1}^{n} y_i - p_i}{n + \lambda}. \tag{10}$$

Finally, the prediction of the decision tree is calculated as

$$F_m(x) = F_{m-1}(x) + \eta \sum_{j=1}^{J_m} O_{J,m} I(x \in R_{j,m}) \tag{11}$$

which is the same as in the original gradient boosting algorithm.

2.8 Random forest

Random forest is a machine learning method that falls into the category of parallel ensembling methods, which will be explained now. It builds upon a concept called bagging, a way to train parallel trees. In bagging, the trees are built with data generated by bootstrap aggregating, which draws multiple random samples with replacements from the original data set. Each subsample B is used to train each tree $\hat{f}$. Hence, each tree learns from multiple, slightly

different subsamples. (Breiman, 1996)

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x). \tag{12}$$

In regression trees, the final prediction is the average of all the predictions as in (12) (Breiman, 1996). Using bagging, creating multiple decision trees on the same data is possible, leading to lower variance and better generalization error. However, bagging leads to a high correlation among the regression trees since the same variables are used in all the trees. This will, in turn, increase the variance and worsen the generalization error. In random forest, this problem is dealt with by decorrelating the trees. When only considering m number of variables at each split, the trees will differ not only in the sample they are built on but also in the features they are built with. At each split, a number of features j are randomly generated, and the feature with the splitting point t that minimizes (1) is then selected. Since each tree is unique with randomized features, this means that the most important variable is not always at the first split, giving the less important variables a chance to impact the result. Consequently, the model will perform better on out-of-sample data. The random forest algorithm also uses some hyperparameters to decrease overfitting to training data. These control the number of trees the model uses, the maximum depth of each tree, and how many features should be considered at each split (Gareth et al., 2013).

2.9 Model evaluation

Here we will introduce the evaluation metrics used to compare the performances of models.

2.9.1 Mean Absolute Error (MAE)

Mean absolute error (MAE) is a measure used to evaluate the performance of regression models. Specifically, it is the mean absolute difference between a given data set's actual and predicted values. It is a popular measure as it is easier to interpret than similar measurements like RMSE (Willmott and Matsuura, 2005).

Let the observed value be denoted by y_i and the predicted value by p_i. The residual of the model is then calculated as

$$\epsilon = y_i - p_i. \tag{13}$$

Then, the MAE is calculated as

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\epsilon_i|. \tag{14}$$

2.9.2 Mean absolute percentage error (MAPE)

Mean absolute percentage error (MAPE) is a measure used to calculate the performance of a regression model. The measurement aims to evaluate the model's performance by calculating the relative difference between the predicted and actual values. Therefore, MAPE is suitable when comparing models where the dependent variable has different units. It is also useful when briefing someone unfamiliar with the units being used on a model's performance, and it gives a universal idea of how good a model is. (1) shows the calculation of MAPE, where ϵ_i is the residuals shown in (13)(Swamidass, 2000).

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\epsilon_i}{y_i}\right| \times 100 \tag{15}$$

2.9.3 Variable importance

When creating ensembles of trees it can be difficult to interpret the results. Therefore, it is common to look at a variable importance chart or index to evaluate what features contributed the most to the results of a model. The importance of a feature is measured based on the influence that feature had on the overall prediction performance of the model. These charts are not only useful to evaluate the final model, but are frequently used for the selection of features when creating a model.

2.9.4 K-fold cross validation

When training and evaluating a model, it is common to use an error metric such as MAE that estimates how well the model performs on unseen test data. However, when minimizing the in-sample error, the model risks overfitting the to training data set. To avoid this, k-fold cross-validation can be used. K-fold cross-validation estimates the out-of-sample MAE by splitting the training data set into k different groups, using one of them as a validation set, similar to a test data set. The model is trained on the first $k-1$ parts of the data and then evaluated on the validation set to give the validation MAE. This is repeated until all k groups have been used as validation sets. The k-fold cross-validation MAE is then calculated as

$$MAE_k = \frac{1}{k}\sum_{i=1}^{k} MAE_i. \tag{16}$$

This gives a good estimation of the out-of-sample error (Hastie et al., 2009). Deciding how many groups to use is a trade-off between bias and variance, and the literature does not suggest any single number of groups. However, typical values for k are 5 or 10 depending on the data set and available computational power (Wong and Yeh, 2019).

2.9.5 Test between the models

Paired t-tests will be used to test for differences in MAE. This test was chosen as it effectively measures the effect of a treatment, which in this thesis is the addition of macroeconomic features to a prediction model. For more details about the performed t-tests, see Appendix.

3 Data

In this section we will present the data, features and training process of the prediction models.

The models are created with a data set provided by Booli Search Technologies AB (Wickert, 2023). The data contains 121 608 house sales from 2018 to 2021, with 69 variables describing the characteristics and location of each sale. However, this data set does not include the added macroeconomic variables. The variables *House price index* (Öljemark, 2023), *unemployment rate* (Ekonomifakta, 2023), *CPI* (SCB, 2023b), *Interest rate* (SCB, 2023c) and *GDP* (SCB, 2023a) has been gathered online and merged to the original data set.

3.1 Data cleaning and features

The data set from Booli contains several features that are not useful for our models, such as customer and Booli ID. To make the models realistic, the variables inflation, house price index, and unemployment rate need to be lagged by one month since they are updated on a monthly basis. This means that, in any given month, the models will rely on the data published in the month prior. Gross domestic product is lagged for the same reason, but a quarter instead, since its information is updated quarterly. Because of computational limitations, only house sales within the Stockholm municipality are included, which are 68 535 sales. However, it would only hurt the model to include observations outside of Stockholm, and building separate models for rural and urban areas is more effective. All house sales above 25 000 000 SEK and their respective rows in the data set have been omitted as the data only contains 108 such sales, and the variation between them is considerable. All observations with any missing values for the chosen features have been omitted. After cleaning the data, 55 636 observations were left. The distribution of the dependent variable sold price can be seen in Figure 2, where the majority of the observations sold prices are under five million SEK.

The correlation plot in Figure 3 shows the correlation among the chosen features. The features rent, living area, and rooms are highly correlated. All the macroeconomic features are also highly correlated. Further, the macroeconomic features and house-specific features have close to no correlation at all. Figure 4 shows a time series for all the macroeconomic variables within our time window. All variables have increased during the investigated time span, and this will affect the house prices according to theory. This change is something that the house-specific features cannot measure and will hopefully increase the accuracy of the predictions. To measure the importance of each feature in our models, the "caret" and "mlr" packages in R will be used to see a variable importance score for each feature. This is a way of determining the relative importance of the predictor variables in the models since it is otherwise hard to interpret what each feature contributes. These packages are powerful tools when evaluating the models.

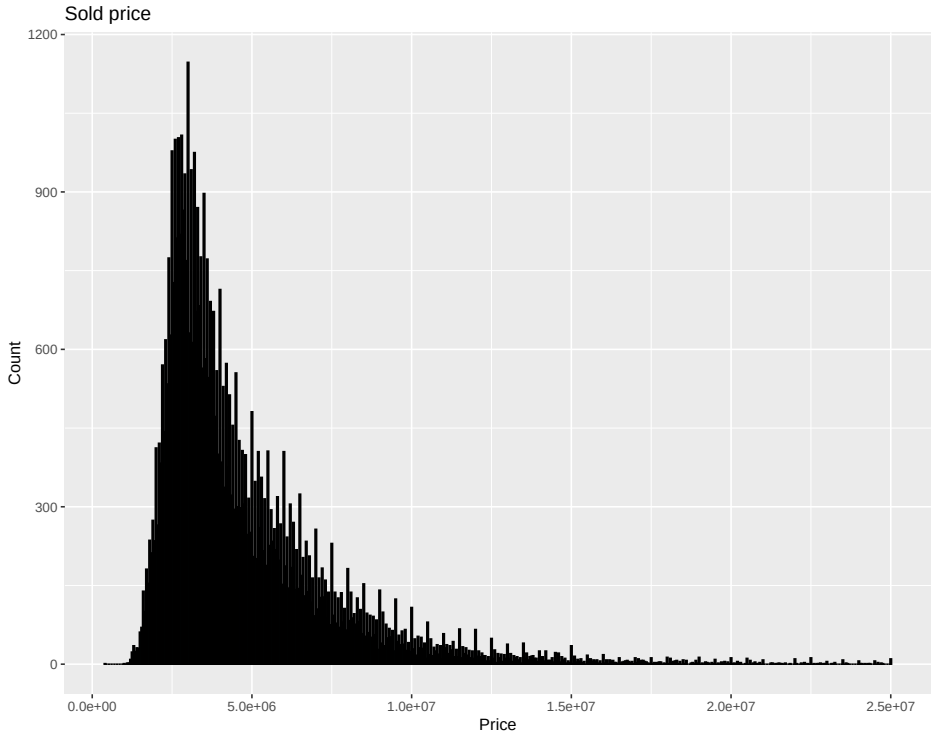

Figure 2: Histogram of sold price.

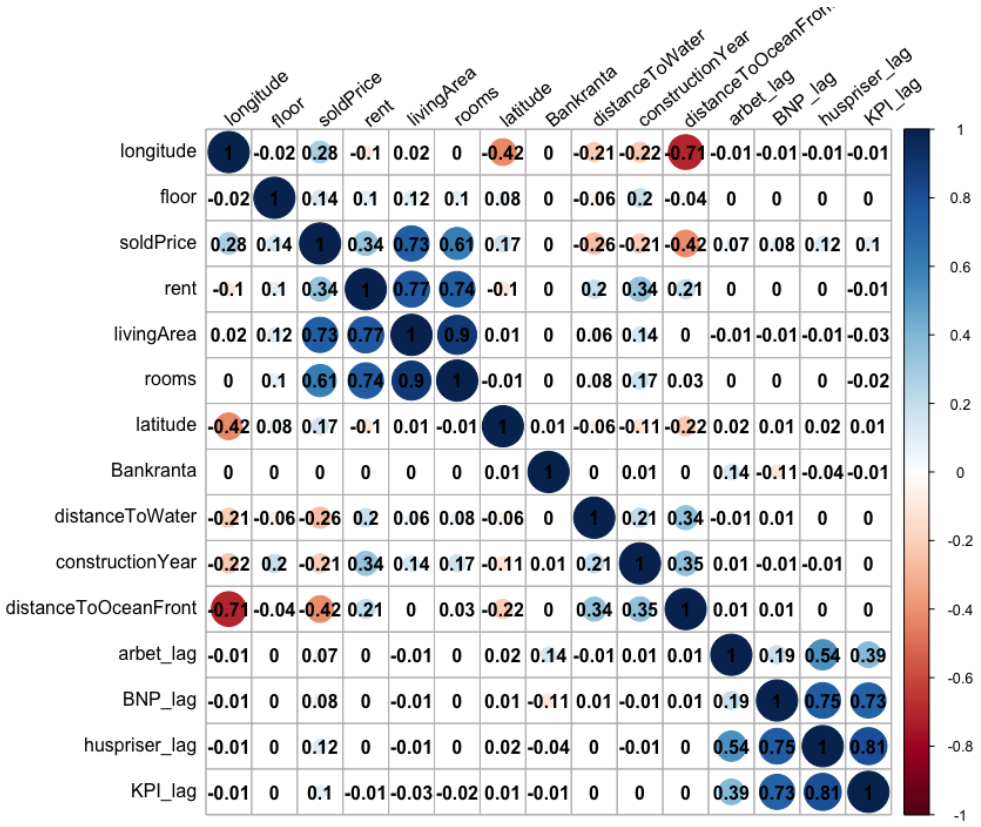

Figure 3: Correlation plot of all used features.

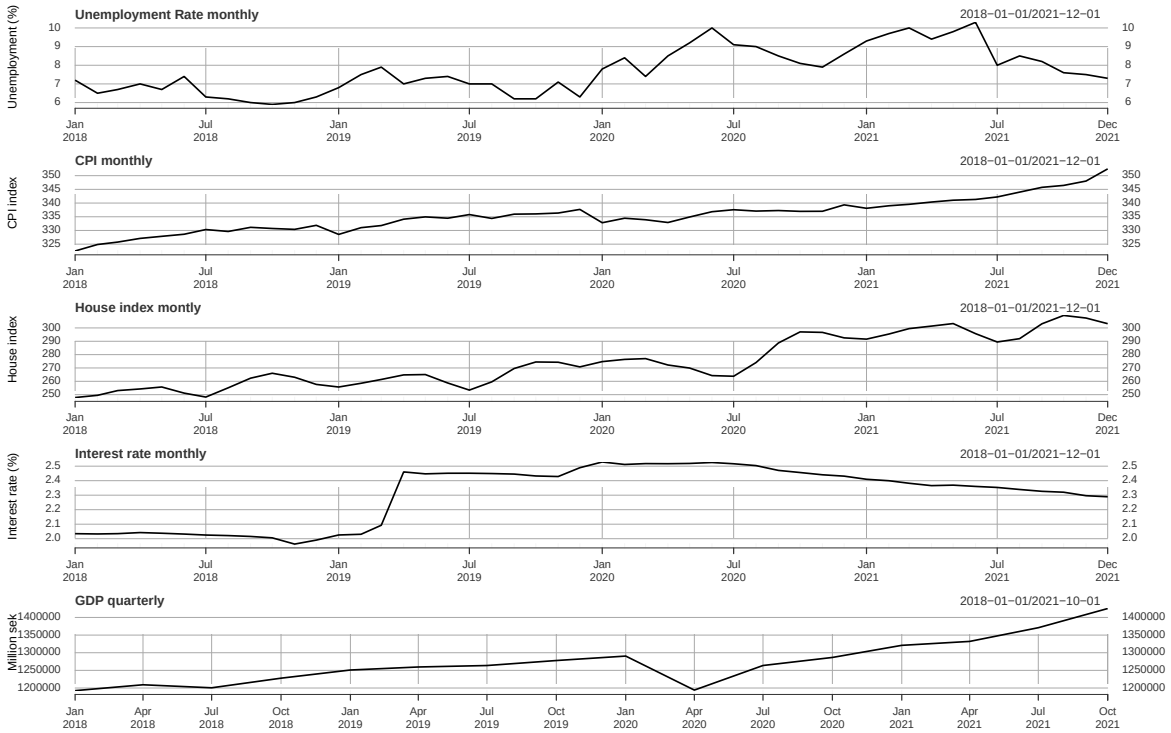

Figure 4: Time series for all macroeconomic features.

3.2 The benchmark and macro features

The models have been created using two different sets of features, with and without the macroeconomic variables but with the same observations. The data set with only house-specific features will be referred to as the "Benchmark model," and the set with macroeconomic variables added will be referred to as the "Macro model". These are presented in Table 1.

3.3 Model training

Each model is trained on 70% of the observations and tested on the remaining 30%. All models use the same training and test data. This split is a standard ratio for partitioning the data (Brownlee, 2018). Each model's hyperparameters have been tuned to the corresponding training data set using 5-fold cross-validation in order to optimize their performance. These grids of values were constructed by using random search and then tuning around the best results.

3.3.1 Random forest tuning

Table 2 shows the considered parameter values in the random forest models. The number of trees controls how many trees will be built during training and features per split are the

Table 1: Features included in the models

Benchmark model	**Macro model**
Latitude	Latitude
Longitude	Longitude
Monthly rent	Monthly rent
Floor number	Floor number
Number of square meters	number of square meters
Number of rooms	Number of rooms
Construction year	Construction year
Distance to ocean in meters	Distance to ocean in meters
Distance to water in meters	Distance to water in meters
Sold Price	Sold Price
	Consumer price index
	House price index
	Interest rate
	Gross domestic product
	Unemployment rate

number of features considered at each split. The random forest models used in this report are made using the "randomForest" package, and tuning and optimization of the hyperparameters are done using the "mlr" package.

Table 2: Hyperparameters for Macro model trained with random forest.

Hyperparameter	Macro	Benchmark
Number of trees	450 - 900 with 50 intervals	450 - 900 with 50 intervals
Features per split	3 - 11	2 - 7

3.3.2 XGBoost tuning

Table 3 shows the range of values considered for the hyperparameters when training the XGBoost models. The number of rounds controls how many trees are built during training, ETA is the learning rate of each round, and max depth controls the maximum number of splits in each tree. Gamma and min child weight is regularization parameters that will control the complexity of the model, leading to less overfitting and better performance on unseen data.

Table 3: Hyperparameters for Macro model trained with XGBoost.

Hyperparameter	Macro	Benchmark
Rounds	14 000, 15 000, 16 000	5000, 7000, 9000
Gamma (γ)	4, 6, 8	2, 4, 6
ETA (ν)	0.01, 0.05	0.01, 0.05
Max depth	4, 6, 8	4, 6, 8
Min child weight	12, 15	9, 12

4 Results

This section contains the results from our study, with the results from the random forest models in the first part of the section, XGBoost in the second part, and a comparison between the models in the third part.

4.1 Random forest model comparison

Two models have been trained using random forest. The first is the Macro model, tuned with the values shown in Table 2. The tuned Macro model had 500 trees and considered seven features per split. The second model is the Benchmark model, and it was tuned with the hyperparameter values presented in the same table. The tuned Benchmark model had 800 trees and considered four features per split. It is reasonable that the Macro model performed optimally with more features per split since that model has more features, but it is unclear why the Benchmark model has more trees than the Macro model. Table 4 presents some essential statistics used to evaluate and compare the models. The Macro model outperforms the Benchmark model in all evaluation metrics, MAE, MAPE, and R^2. A paired t-test is performed (see Appendix) to see if the difference in MAE is significant, with the null hypothesis that the population MAE is the same and the alternative hypothesis that the population MAE of the Macro model is lower than the Benchmark model. The observed p-value is 10^{-16}. This means that we accept the alternative hypothesis that the population MAE is lower for the Macro model compared to the Benchmark model. The observed R^2 values also show that the Macro model explains more of the variation in the data, which aligns with the hypothesis test results. Not only is the MAE significantly lower for the Macro model, but also the residual standard deviation, meaning that the residuals are more closely distributed, and the model will hence be more consistent when predicting.

Figure 5 shows the residuals for the two models; the spread is smaller for the Macro model as the dots are closer to the red line. This is in line with the difference in standard deviation presented in Table 4. The figure shows that the spread of the residuals seems to be heteroscedastic, meaning that the spread of the residuals is not the same for all prediction ranges. The variance of the residuals is lower for predictions made on less expensive houses and gets bigger when predicting more expensive houses. It is in Figure 5 visually difficult to see how the predictions are distributed across the price ranges, but looking at Figure 2, it

Table 4: Statistics for the random forest models.

Statistic	**Macro model**	**Benchmark model**
MAE	361 083 ± (4149.57)	450 575 ± (4853.38)
MAPE	7.05%	8.61%
R^2	0.9479787	0.926127
SD residuals	646324	770193

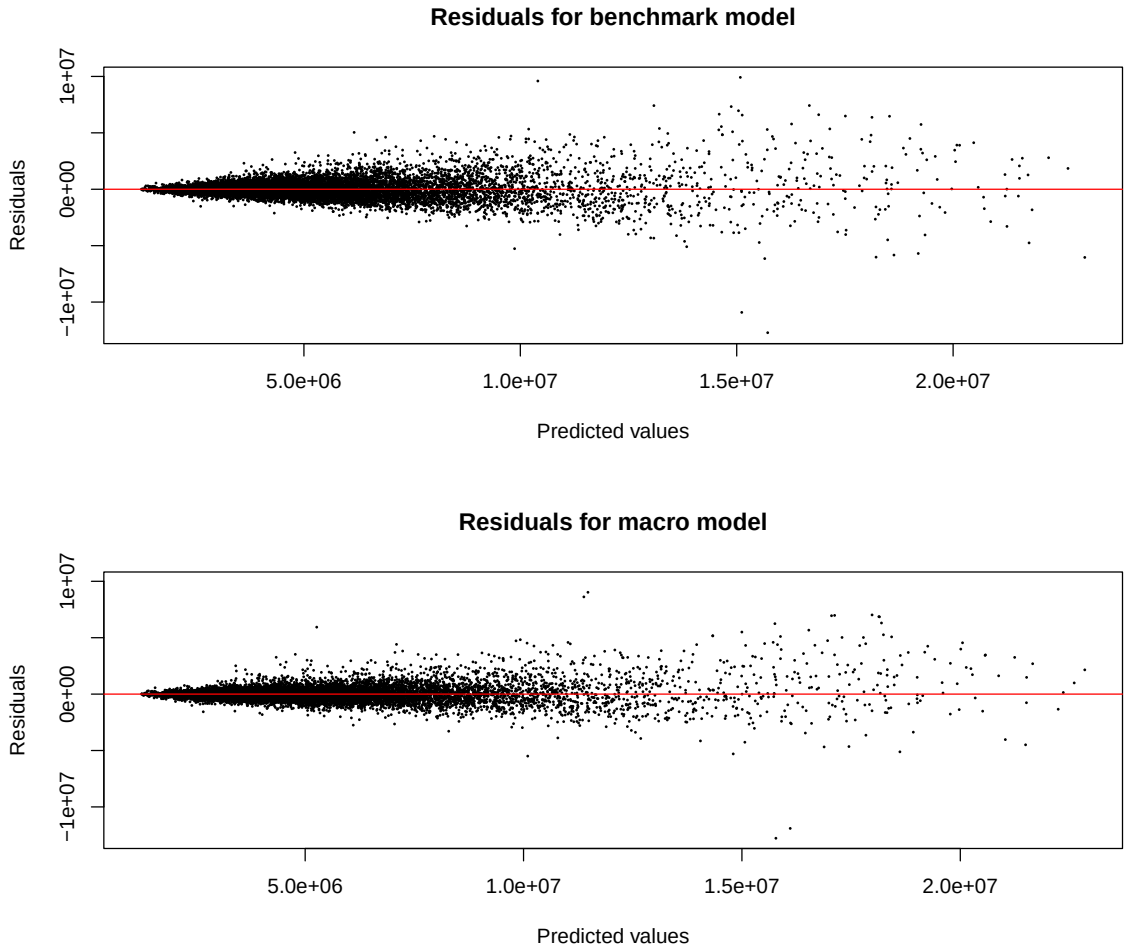

Figure 5: Residuals for predictions of both random forest models

can be seen that the majority of the sold prices are below 5 million SEK.

The Macro model has a significantly better prediction performance compared to the Benchmark model. However, Figure 6 shows that all macroeconomic features score low in the variable importance plot. The house-specific features are, in general, of more importance, according to the figure. The feature *livingArea* is the feature with the highest importance in both models. The observed R^2 value is about 2.2 percentage points better for the Macro model meaning that the increase in explained variation is rather low compared to the already high R^2.

4.2 XGBoost model comparison

As in the previous section, two different models have been trained but with XGBoost. The Macro model has been tuned using the values in Table 3. The best hyperparameters found for this model were the number of rounds to be set to 15 000, Gamma to be set to 8, ETA to

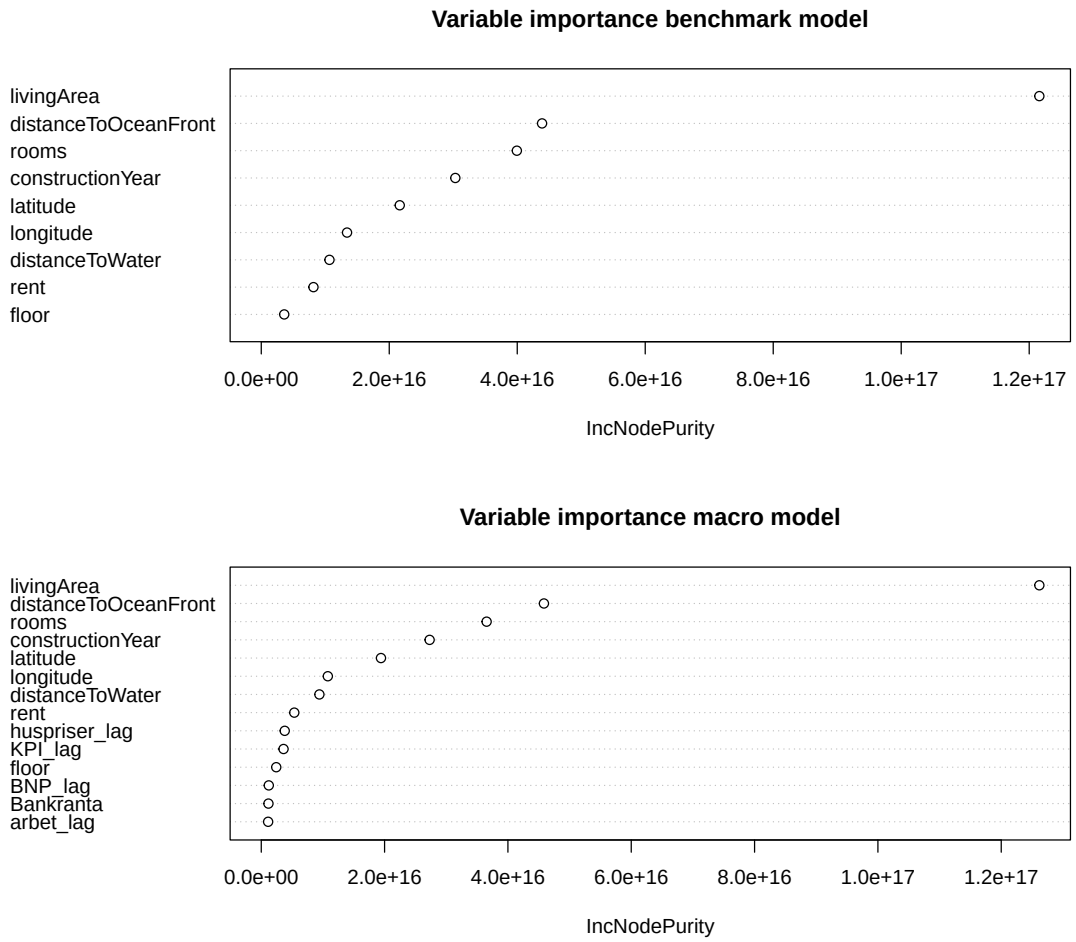

Figure 6: Variable importance score for random forest models.

be set to 0.01, Max depth to be 8, and min child weight to be 15. The best hyperparameters for the Benchmark model are the number of rounds to be set to 7000, Gamma to be set to 6, ETA to be set to 0.01, max depth to be set to 8, and min child weight to be set to 12. These values seem reasonable as the Macro model needs more rounds to optimally learn from the data as it has more features than the Benchmark model. Further, when the number of rounds is higher, the regularization parameters need to control the complexity of the model to prevent overfitting. Table 5 shows that the MAE, MAPE, and R^2 are better for the Macro model. A paired t-test was performed to test the difference in MAE (see Appendix), with the null hypothesis that there is no difference in population MAE and the alternative hypothesis that the Macro model has a lower population MAE. The observed p-value from the test is 10^{-16}. This means that we accept the alternative hypothesis that the Macro model has a lower population MAE compared to the Benchmark model. The standard deviation is also lower for the Macro model, meaning that the residuals are more closely distributed, and the model will hence be more consistent when predicting house prices.

Table 5: Statistics for the XGBoost models.

Statistic	**Macro model**	**Benchmark model**
MAE	318 816 ±(3674)	440 463 ±(4716.21)
MAPE	6.29%	8.42%
R^2	0.9592898	0.9296146
SD residuals	571755	751797

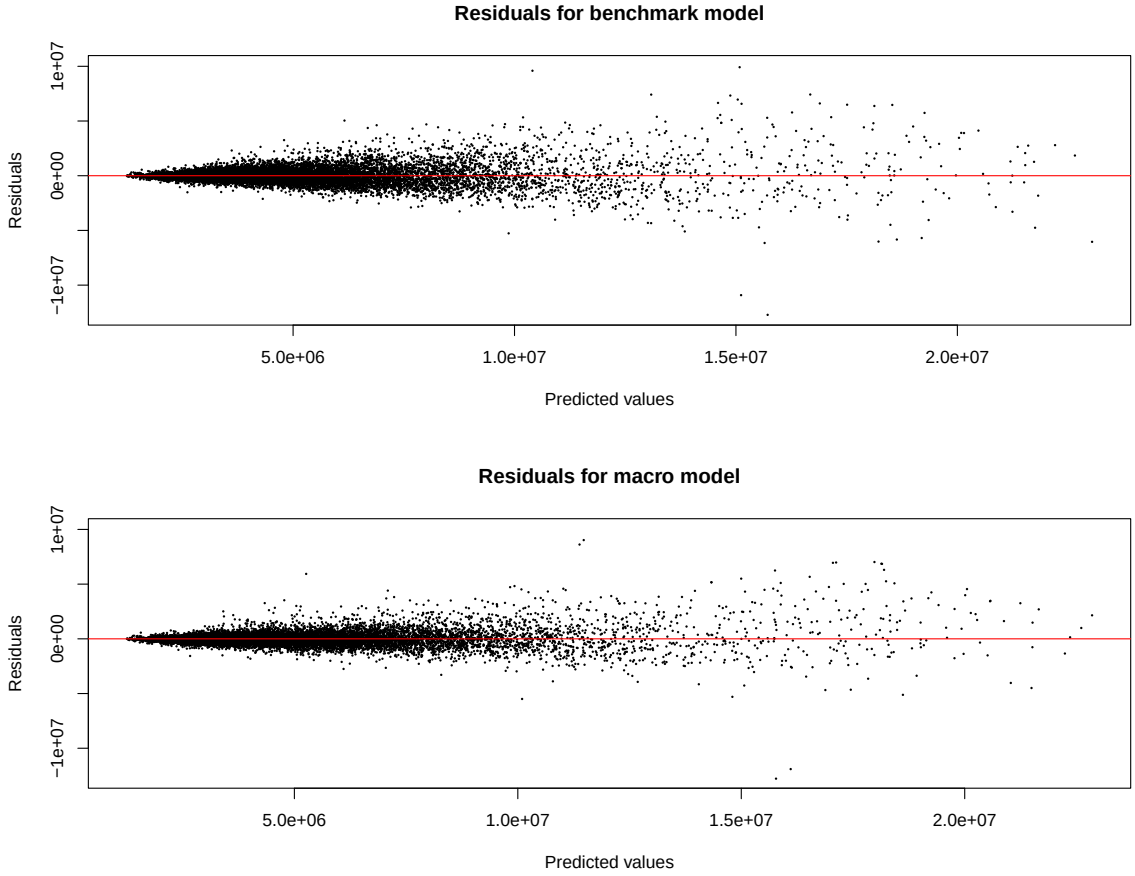

Figure 7: Residuals for predictions of both XGBoost models.

Figure 7 shows the residuals for each model, where the smaller variance of the Macro model is visualized compared to the Benchmark model. This is confirmed in Table 5. The figure indicates that the spread of the residuals is heteroscedastic. The variance of the residuals is lower when predictions are made on less expensive houses and increases when predicting more expensive houses. When looking at Figure 7, it is easy to forget that the majority of the house prices and hence house price predictions are under 5 million SEK. That means the majority of the predictions have lower variance.

In Figure 8, the macroeconomic features are at the bottom when ranking the importance of each feature. The house-specific features are generally of higher importance, with *living area* being of most importance. The observed R^2 value has increased by about 2.6 percentage

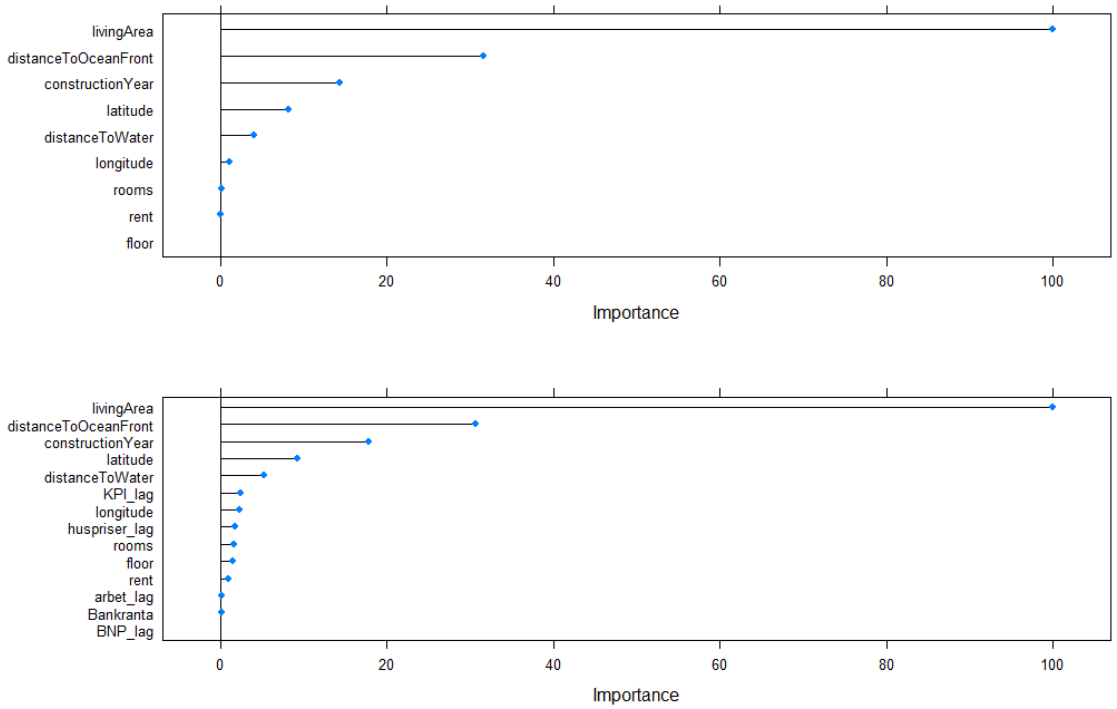

Figure 8: Variable importance score for XGBoost models.

points for the Macro model, which is a rather low increase when looking at the already high R^2 value. This could be one explanation of why the macroeconomic features score low when checking the variable importance.

4.3 Comparison between the algorithms

Sections 4.1 and 4.2 showed that the Macro model outperforms the Benchmark model, no matter if it is trained using XGBoost or random forest. In this part, we will investigate how the algorithms compare to each other in terms of predictive performance. A paired t-test (see Appendix) between the Macro models for random forest and XGBoost was performed, with the null hypothesis that there is no difference in population MAE and the alternative hypothesis that there is a difference in population MAE. The observed p-value is 10^{-16}. When performing another paired t-test (see Appendix) between the Benchmark model for XGBoost and random forest, with the same hypothesis, the observed p-value is 10^{-7}. This is also a significant result which means that the null hypothesis is rejected. These results are in line with Figure 9, which shows the confidence intervals for the difference in MAE for all performed tests. If the confidence intervals were to cross the red dotted line, the results of the test would be insignificant. The test between the Benchmark models (see Appendix) is, as seen in Figure 9, closer to accepting the null hypothesis when compared to the other tests. However, a p-value of 10^{-7} is still extremely low and strongly indicates that the XGBoost model is better. As seen in Tables 4 and 5, the standard deviations of the residuals are lower

for both the Macro and Benchmark models made with XGBoost when comparing them to the Macro and Benchmark models made with random forest. This means that the residuals are more closely distributed for the XGBoost models. The observed R^2 values are better for the XGBoost models, meaning they explain more of the variation in the data compared to the models made with random forest.

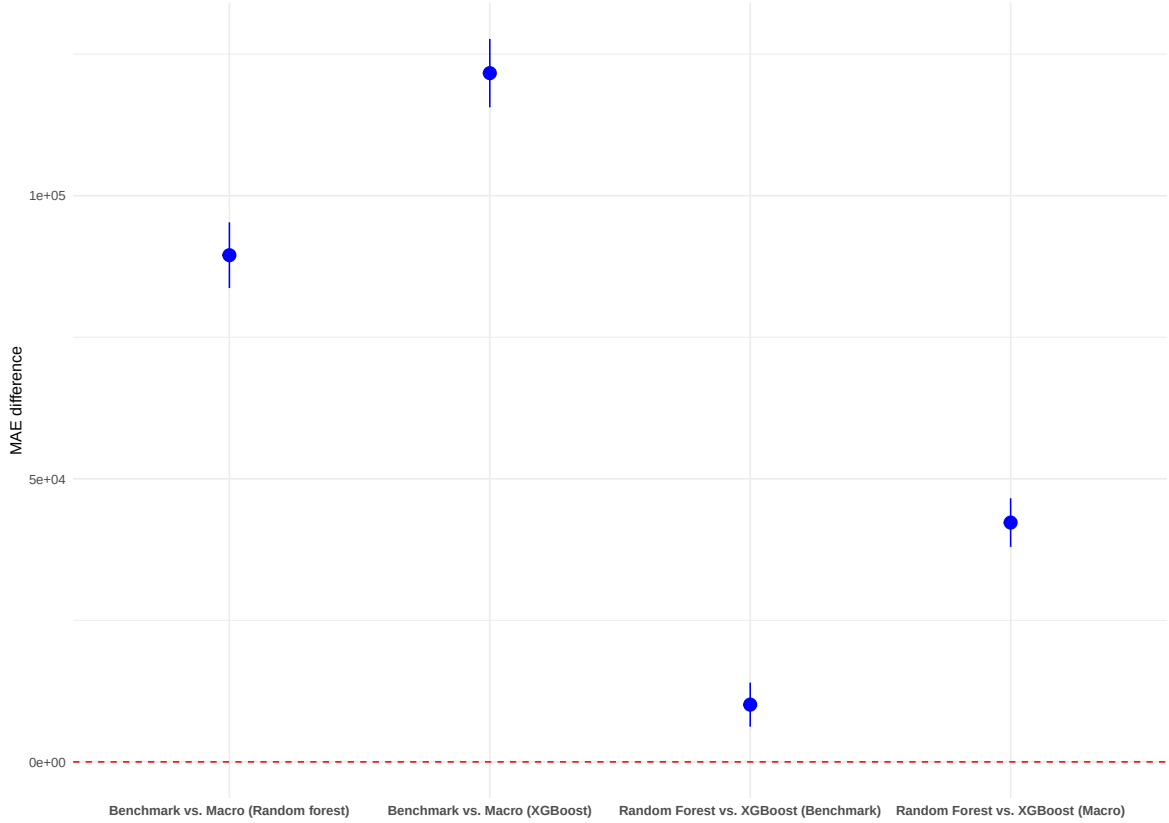

Figure 9: Confidence intervals for t-tests in the difference of MAE.

5 Discussion

This section will discuss the presented results, divided into three parts, first, a discussion of the features, then a discussion of the models, and lastly, we will discuss future research.

5.1 Comparison of features

As seen in Tables 4 and 5, the results show that, for both algorithms, the Macro model outperformed the Benchmark model in MAE, MAPE, and R^2 value. Additionally, the Macro models also have lower standard deviations. Finally, all tests showed that the difference in population MAE was significant at the 5% significance level when comparing the Benchmark and Macro models. This gives a strong indication that the macroeconomic features have had a positive impact on the accuracy of both prediction algorithms. Contrary to our initial belief, however, the added features in the Macro model got relatively low importance scores for both random forest and XGBoost. Even more unexpectedly, despite the low importance scores, the Macro models still outperformed the Benchmark. This result could be due to several factors. Firstly, all the macroeconomic features are highly correlated, meaning they may explain the same variation in the house price variable, which will lead to the features sharing the explained variation among them. This seems reasonable as, for example, the unemployment rate will depend mainly on other added macroeconomic features, for instance, GDP and inflation. However, correlation among the features is not necessarily a problem in tree-based models, as the features may synergize in a way that improves the model when able to predict together. Secondly, the macroeconomic features might not score the highest on the variable importance charts. However, all explained variation by these features is variation that was previously unexplained, as the Benchmark features have no information about the macroeconomic environment. For example, if the number of rooms is removed, the living area can cover much of the same variation, and thus the model will likely not lose that much accuracy. This indication is strengthened when looking at the R^2 values for the different sets of features, as it increases with the macroeconomic features included, meaning more of the variation in house price is explained. However, the increase in R^2 is just a couple of percentage points when adding the macroeconomic features, which is a rather low amount of explained variation added. Because of the high correlation among these features, they are given low importance scores individually but bring a noticeable improvement in explained variation collectively.

Since the variation in macroeconomic variables varies across time periods, the result may have been different if another time window was used. As this thesis only included sales between 2018 and 2021, the results are not necessarily fully generalizable outside of this time period, or rather, in time periods with significantly different values for the macroeconomic variables. Figure 4 shows the variation during our time period in the added features. This is

the variation that increases the performance of the model and that the Benchmark features were unable to explain. We expect that the macroeconomic features would have a larger impact on house prices the older the training data is, since over long periods of time, the change in house prices is heavily dependent on the macroeconomic environment. Thus, if there is a large gap in time between the observations in a training model, and the observations in the test set, the more important it is to include the macroeconomic variables. The old sold price would have been affected by things like inflation and GDP growth. Considering the results we got from only 4 years of data, the effects the incorporation of these features could have over longer time spans is interesting. It makes it possible to use larger data sets to train these prediction models compared to before, as the passed time otherwise would make the model less accurate.

5.2 Comparison of algorithms

The results show that the XGBoost model outperforms the random forest model for both sets of features. This is visualized in Figure 9, where it is clear that none of the 95 % confidence intervals crosses zero. The reason why XGBoost outperforms random forest could be many and is not obvious. For Example, XGBoost could gain an advantage over random forest as it is always able to consider all features at every split. In contrast, random forest are only able to use a random subset of the available features in the data set. This may cause XGBoost to better map the complex synergies among the features and capitalize on their patterns. It also makes randomness have an effect in the random forest model, as the wrong set of features at some splits may lead to bad decision trees, even though this effect probably is very low over hundreds of trees. Also, even though both algorithms are able to handle non-linear relationships, XGBoost is known to excel in this area and might therefore be able to exploit this to a further extent compared to random forest. The bottom line is that XGBoost was able to find better patterns and thus outperformed random forest when adding macroeconomic variables. The largest p-value is for the comparison of XGBoost and random forest using the Benchmark model. This being said the p-value is still very low at 10^{-7}. The reason why the p-values are so extreme is not only because the difference in MAE is so large but also because the test is made with over 16 000 pairs of observations. This makes a very small difference in MAE significant, as the impact of randomness is extremely small over such a large sample.

5.3 Future research

The topic of house price predictions has been widely covered in terms of tuning machine learning models and comparing them. Feature selection, however, does not seem to have gotten as much attention from academics, and this is where we see the potential to further improve the existing models. You can only improve a model by so much using tuning and

feature engineering methods, but adding new features can drastically increase the accuracy. As this thesis only focused on confirming the useability of macroeconomic features, future research could further explore what macroeconomic features are the best and what combinations of features synergize to improve the models the most. Additionally, other time frames could be interesting to inspect as macroeconomic features likely have a varying amount of impact across different time periods.

6 Conclusion

This section will address the following stated research questions:

- *Will the addition of macroeconomic features significantly improve the performance of house price prediction models?*
- *Is random forest or XGBoost better at predicting house prices?*

This report aimed to compare and evaluate the impact of including a set of macroeconomic features in models made with XGBoost and random forest. Additionally, it aimed to compare the methods themselves, investigating what algorithm performed best when predicting house prices. The results showed that the macroeconomic features significantly improved all models they were implemented in. It was also found that XGBoost outperformed random forest for both sets of features. The reason why XGBoost was better is not clear and could be due to several reasons, but the key takeaway is that XGBoost found better patterns among the features. Although the macroeconomic features improved the performance, they all scored low on the variable importance chart, which could be due to several factors, including synergistic effects and a share of additional explained variation. Thus, we can conclude that the macroeconomic features improved the model and that the XGBoost algorithm is better than random forest at predicting house prices.

References

N. Apergis et al. Housing prices and macroeconomic factors: prospects within the european monetary union. *International Real Estate Review*, 6(1):63–74, 2003.

K. Bohlin and M. Kellgren. The predictive relationship between swedish house prices and consumer price inflation. *Student essay, Univerity of Gothenburg, GUPEA*, 2021.

L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

J. Brownlee. *Deep learning for time series forecasting: predict the future with MLPs, CNNs and LSTMs in Python*. Machine Learning Mastery, 2018.

T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

A. D´Agostino. Why having many features can hinder your models performance, 2022. URL `https://towardsdatascience.com/why-having-many-features-can-hinder-your-models-performance-865369b6b8b1`. Accessed on 22 April 2023.

J. Ekberg and L. Johansson. Comparison of different machine learning methods' capability to predict housing prices. *Student essay, Royal Institute of Technology, Diva*, 2022.

Ekonomifakta. Arbetslöshet. `https://www.ekonomifakta.se/fakta/arbetsmarknad/arbetsloshet/arbetsloshet/`, 2023. Accessed 22 of April 2023.

M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim. Do we need hundreds of classifiers to solve real world classification problems? *The Journal of Machine Learning Research*, 15(1):3133–3181, 2014.

J. Gareth, D. Witten, T. Hastie, and R. Tibshirani. *An introduction to statistical learning: with applications in R*. Spinger, 2013.

A. Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, Inc., 2022.

H. D. Getahun. The effect of interest rates on housing prices in sweden:: The case of one and two dwelling buildings. *Students essay, Royal Institute of Technology, Diva*, 2011.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3(Mar):1157–1182, 2003.

T. Hastie, R. Tibshirani, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

R. A. Irizarry. *Introduction to data science: Data analysis and prediction algorithms with R*. CRC Press, 2019.

W. Revend. Predicting house prices on the countryside using boosted decision trees. *Students essay, Royal Institute of Technology, Diva*, 2020.

S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

SCB. Bnp från användningssidan (ens2010), försörjningsbalans efter användning. kvartal 1980k1 - 2022k4, 2023a. URL `https://www.statistikdatabasen.scb.se/pxweb/sv/ssd/START__FM__FM5001__FM5001C/RantaT01N/`. Accessed 24 of April 2023.

SCB. Konsumentprisindex (kpi), totalt, 1980=100. månad 1980m01 - 2023m03, 2023b. URL `https://www.statistikdatabasen.scb.se/pxweb/sv/ssd/START__PR__PR0101__PR0101A/KPItotM/`. Accessed 20 of April 2023.

SCB. Utlåningsräntor till hushåll och icke-finansiella företag fördelat på räntebindningstid. månad 1987m03 - 2023m03, 2023c. URL `https://www.statistikdatabasen.scb.se/pxweb/sv/ssd/START__FM__FM5001__FM5001C/RantaT01N/`. Accessed 24 of April 2023.

P. M. Swamidass. *Encyclopedia of production and manufacturing management*. Springer Science & Business Media, 2000.

S. Wickert. Kontakt. `https://www.booli.se/p/kontakta-oss`, 2023. Accessed 21 of April 2023.

C. J. Willmott and K. Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30 (1):79–82, 2005.

T.-T. Wong and P.-Y. Yeh. Reliable accuracy estimates from k-fold cross validation. *IEEE Transactions on Knowledge and Data Engineering*, 32(8):1586–1594, 2019.

T. Xu. The relationship between interest rates, income, gdp growth and house prices. *Research in Economics and Management*, 2(1):30–37, 2017.

Z.-H. Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012.

J. Öljemark. Bostadspriser. `https://www.ekonomifakta.se/Fakta/Ekonomi/bostader/Bostadspriser/`, 2023.

Appendix

Paired t-tests[1]

Significance level: $\alpha = 0.05$

Assumptions for the test:

- The observations in each group are independent of each other.
- MAE $\sim$ N(MAE, $\sigma_{\text{residuals}}/\sqrt{n_{\text{residuals}}}$)
- The dependent variable should not contain any extreme outliers

Test Statistic:

$$t = \frac{\bar{d} - 0}{\frac{s_d}{\sqrt{n}}}$$

where:

$$\begin{aligned} t &: \text{t-statistic} \\ \bar{d} &: \text{mean of the differences} \\ s_d &: \text{standard deviation of the differences} \\ n &: \text{number of paired observations} \end{aligned}$$

The degrees of freedom for this test is 32 623.
Results when calculating the t-statistic:

Table 6: Paired t-test Results

Hypothesis	**Algorithm**	**t-statistic**	**p-value**
$H_0 : MAE_{macro} = MAE_{benchmark}$ $H_1 : MAE_{macro} > MAE_{benchmark}$	Random forest	*30.2*	2.2×10^{-16}
$H_0 : MAE_{macro} = MAE_{benchmark}$ $H_1 : MAE_{macro} > MAE_{benchmark}$	XGBoost	*39.5*	2.2×10^{-16}
$H_0 : MAE_{macroRF} = MAE_{macroXGB}$ $H_1 : MAE_{macroRF} \neq MAE_{macroXGB}$	Random forest vs XGBoost	*19.31*	2.2×10^{-16}
$H_0 : MAE_{benchmarkRF} = MAE_{benchmarkXGB}$ $H_1 : MAE_{benchmarkRF} \neq MAE_{benchmarkXGB}$	Random forest vs XGBoost	*5.09*	3.6×10^{-7}

Note: The p-values reported as 2.2×10^{-16} *indicate values below the machine precision limit of R. R cannot display p-values smaller than this threshold.*

[1]The paired t-test is a variant of the Diebold-Mariano test.