Stockholm School of Economics
Saga Tran | 25770
Course: BE903 - Current Topics in Data Science for Business

Model Validation Documentation for Regression model on the Gothenburg Housing Market

Date: 2025/05/12 11:59 pm

1 Introduction

This project aims to develop a machine learning model capable of predicting property sales prices in Gothenburg based on historical transaction data and financial information from housing associations.

This document presents the validation report for the Gothenburg Housing Market Prediction model. Section 2 gives an overview of the available data, the steps taken to preprocess and engineer features, and any new variables created to enhance predictive performance. Section 3 discusses the rationale behind selecting the final model, and gives a comparison against a baseline and a challenger model. Section 4 outlines the steps taken to avoid overfitting or underfitting, and details on hyperparameter tuning and model optimisation. Section 5 gives an assessment of the model's sensitivity to data quality issues and extreme values. Finally, section 6 summarises the model's performance based on the Root Mean Squared Error (RMSE) metric, the contribution of different features to predictive power, and considerations regarding model drift over time.

2 Feature Engineering

2.1 Data description

The Apartment dataset is the primary dataset. It contains information about individual apartments in Gothenburg that were sold or listed for sale over a period of approximately 12 years. Each row in the dataset represents a unique apartment sale/listing, and is identified by an ID, and includes features that directly describe the apartments characteristics, location, and sale details. This dataset is mid-sized, with 38 columns (including the target variable) and approximately 70 000 rows, and provides the core features for predicting the sell-price.

The Annual report dataset contains financial information about the housing associations over multiple years. Each row represents a financial report for a specific housing association, uniquely defined by organisation number and fiscal year. This provides information about the financial health of the housing association, which may have an impact on the selling prince of the apartments within it. This dataset is larger in size (approximately 80 000 rows and 14 columns) than the Apartment dataset, but it is smaller in terms of unique entities, since each organisation number is mapped to multiple rows (different fiscal years).

Lastly, the Housing Association dataset contains information about the housing associations that manage the apartments in the Apartment Dataset. Each housing association is identified by a unique organisation number. This dataset provides information about the name and construction year of the housing association. It is the smallest in size with approximately 25 000 rows and 3 columns.

2.2 Degree of Completeness

We begin by checking the amount of null-values in each column. Tables 1 - 4 evaluates the completeness, i.e., the degree of non-missing values of each column, using equation (1).

$$\text{Completeness}[\%] = \left(1 - \frac{\text{Count of missing values}}{\text{Total no. of entries}}\right) \times 100 \tag{1}$$

By doing this, we can directly identify which columns that contain too low amount of data to provide reliable data for modelling. We have identified 3 thresholds. If the degree of completeness is above 75%, the column classifies as green and passes the test. If the degree of completeness is between 50% and 75%, the column is classified as yellow. These columns are retained for now, but we keep in mind that they may introduce noise if used in modelling. The ones that are classified as red, that is, have a degree of completeness below 50%, have too much missing data to be reliably used in modelling.

Table 1: Apartment data

Column	%
id	100.00
additional_area	0.90
agency_id	99.74
amenities	100.00
asking_price	51.64
brokers_description	100.00
cover_photo_description	2.02
customer_area_description	79.96
district	100.00
energy_class	39.27
floor	87.76
has_balcony	0.00
has_fireplace	0.00
has_patio	0.00
has_solar_panels	1.86
heating	8.36
height	2.49
housing_association_fee	98.75

Table 2: Apartment data cont'd

Column	%
housing_association_org_number	97.18
is_new_construction	1.91
key	2.74
latitude	100.00
living_area	99.70
locality	100.00
longitude	100.00
municipality	100.00
operating_cost	69.42
plot_area	0.14
postcode	100.00
primary_area	100.00
region	100.00
rooms	99.83
sell_date	100.00
street_address	100.00
width	2.49
sell_price	97.44

Table 3: AnnualReport data

Column	%
org_number	100.00
fiscal_year	100.00
association_liability	99.34
long_term_real_estate_debt	100.00
long_term_real_estate_debt_other	54.71
number_of_rental_units	98.01
number_of_units	100.00
plt_is_leased	99.03
savings	99.39
total_commercial_area	84.65
total_living_area	100.00
total_loan	100.00
total_plot_area	83.73
total_rental_area	81.60

Table 4: Housing Association data

Column	%
org_number	100.00
name	100.00
construction_year	82.02

From this analysis, we note that the additional area column is almost entirely missing. Most likely, the additional area represents the extra space, e.g. storage or outdoor areas, in the apartment which could influence the sell price. However, with only 0.90% completeness, it is not feasible to use. The cover photo description probably describes the cover photo on the listing, which may contain useful information like keywords about views or apartment conditions. With only 2.02% completeness, it becomes impractical to extract meaningful features. Additionally, energy class, has balcony, and has fireplace are all theoretically important for the sell price. Buyers often prefer energy efficient homes due to lower operating costs, and balconies and fireplaces are value-adding features in an apartment. However, these columns, as well as the other's that are marked red, lack sufficient data to be useful and are thus discarded from further tests.

The asking price, operating cost, and long term real estate debt other columns have a completeness degree between $50 - 75\%$ and are thus classified as yellow. The initial listing price, that is, the asking price, should theoretically be a very strong predictor for the selling price. Despite only having 51.64% completeness, this column seems too important to discard. However, we will need to make some missing value treatment. The operating costs can impact the affordability of an apartment and thereby the selling price. However, if there is little difference between apartments, i.e., if the data is almost uniformly distributed, this will have the same impact on all apartments and we should think about discarding it. The long term real estate debt other is probably some subset of debt, distinct from regular long term real estate debt. Since the long term real estate debt column is 100% complete, and its predictive power will probably overlap with the other long term debt, we can think about discarding this one too.

2.3 Primary Keys Uniqueness

It is important to perform a uniqueness test on each of the datasets to check if the primary keys (the columns that are supposed to make each row unique) really identifies each column as unique. Equation (2) is used to validate the datasets.

$$\text{Uniqueness} = \frac{\text{Unique primary keys}}{\text{Total no. of entries}} \tag{2}$$

All three datasets scored 100% of the uniqueness test, so there is no need for any further analysis.

2.4 Data Distribution

Ensuring the distribution of the data contained in each column exhibits some variability is important for determining their usefulness in predicting the sales price. Constant or near-constant values in a columns means that each row will experience the same effect if we use that column in modelling, which gives little to no predictive power. Figures 1, 3, and 4 shows the distribution of the columns not removed in the completeness analysis.

2.4.1 Apartment Data

We can directly note that location, municipality, populated area, and region displays distributions with very little variability. This means that most values are the same or nearly the same. Since such columns are unlikely to provide any predictive power, we will choose to discard them.

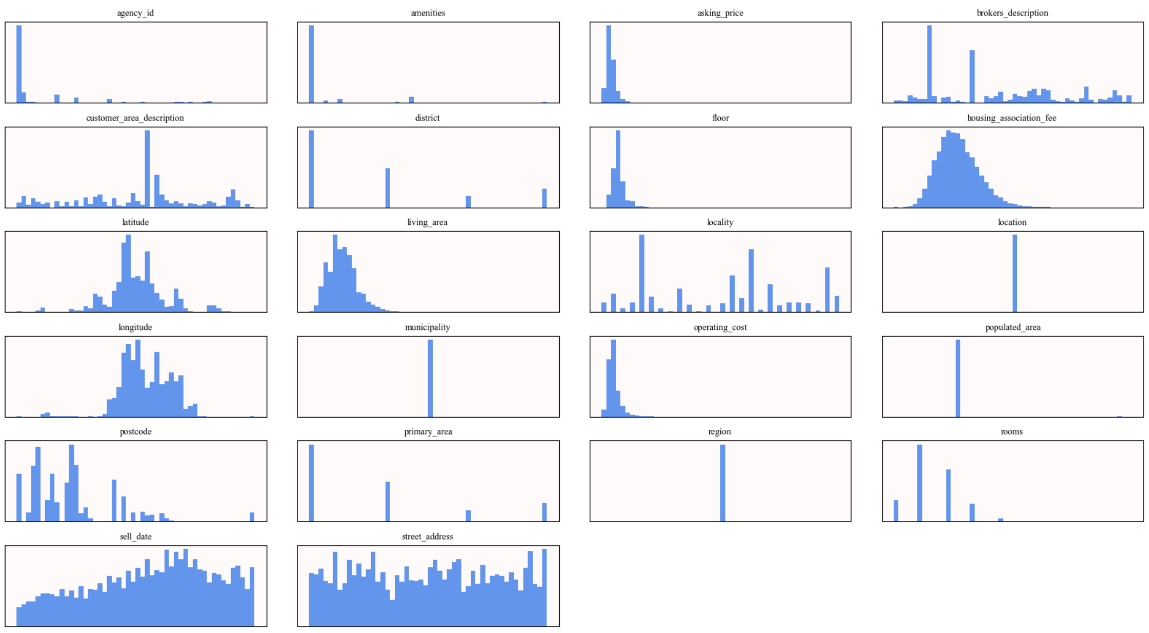

Figure 1: Distribution of Apartment data columns

Agency ID shows many unique values with one clear dominant agency. This indicates variability, but the high cardinality (many unique agencies) may make the column less useful as a feature for modelling, unless we use some type on encoding, e.g. group the rare agencies into an "other" category. The same goes for amenities. Brokers description and customer area description both show high cardinality with some description/areas being more frequent than others. These are text columns so variability is expected.

Many of the numerical columns, such as asking price, floor, living area and operating cost are right-skewed, with most values clustered at the lower end and a long tail of higher values. This phenomenon in real estate data can be explained by the fact that most apartments are smaller and cheaper, with a few large/expensive outliers. One solution to this is to apply log-values for these potential features.

The variability in the sell date column suggests potential temporal trends in the sell price. Looking further into this, figure 2 shows a steady increase in apartment sales from 2012 to 2021, with a peak at approximately 8000 sales per year. The decrease from 2022 and onwards can be due to many reasons. The downturn during 2022 and 2023 can be a delayed effect of the COVID pandemic, since these years experienced high interest rates and unemployment. Other reasons could be that there were not so many new apartments built, or that there generally was an economic dip.

Looking at the months, we directly note that June and December are the least popular months to buy an apartment in. Peak sales are during spring and fall. We will include these two new variables in the feature selection process.

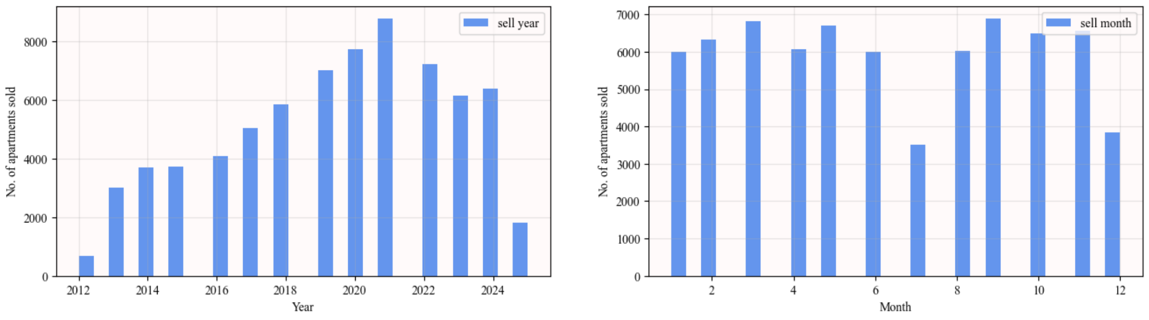

Figure 2: Temporal trends in no. of apartments sold

2.4.2 Annual Report Data

In the Annual Report data, we find that almost all columns except for long-term real estate debt, number of unit, total living area, and total loan exhibits no to very little variability. Thus, we only keep these four. We note the same right-skewed pattern here, which explains that most associations have smaller debt, fewer units, or smaller living areas, with some amount of outliers.

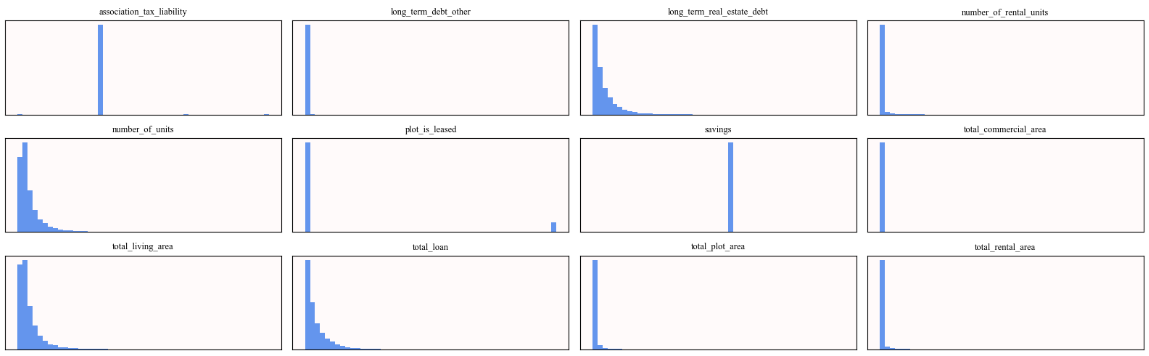

Figure 3: Distribution of Annual Report data columns

2.4.3 Housing Association Data

Since all Housing associations have different names, the distribution is almost completely uniform, which is not useful for predicting the sell price. However, construction year shows a multi-modal distribution with several peaks, which indicates that there is large variability in the construction years of the housing associations.

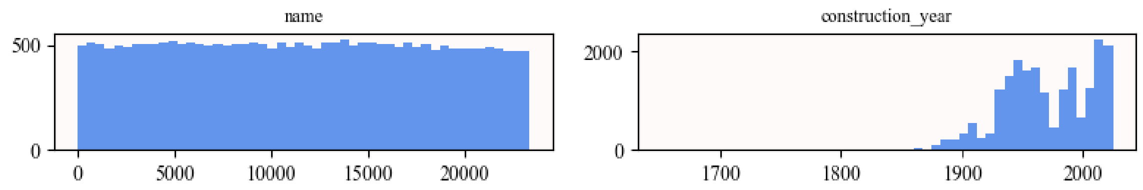

Figure 4: Distribution of Housing Association data columns

2.5 Aggregation of Annual Report Data

To incorporate the data from all years in the Annual Report dataset, we aggregate the historical financial data for each housing association. Since the Apartment dataset is linked to a housing association via housing association org number, and each housing association can have multiple annual report entries over different years, we need to make a summary of the financial data across the available years for each organisation number. In the aggregation, we compute mean, standard deviation, max, and min value for each of the four variables kept after the distribution test (long-term real estate debt, number of units, total living area, and total loan). These values are merged into the Apartment dataset.

This aggregation also allows for discovery of trends in the case where one housing association has more than more year reported. Trends are computed using the `linregress` function of the scipy.stats library, and the value of the trend is equal to the slope over all years. Figure 5 highlights the distribution of positive and negative debt and loan trends over all available years for each housing association. Moreover, these trend columns have approximately 85% completeness and will therefore be added to the set of potential features.

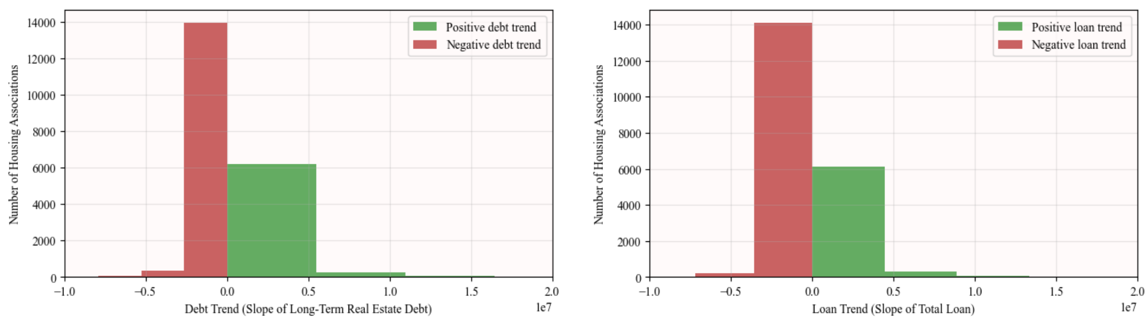

Figure 5: Loan and debt trends

Lastly, we also computed debt per unit, loan growth rate, and debts to loan ratio as per equations (3).

$$\begin{aligned}\text{Debt per unit} &= \frac{\text{log long term real estate mean debt}}{\text{log no. of units mean units}}\\ \text{Loan growth rate} &= \frac{\text{loan trend (log-scale)}}{\text{log total mean loan}}\\ \text{Debt to loan ratio} &= \frac{\text{log long term real estate mean debt}}{\text{log total mean loan}}\end{aligned} \tag{3}$$

These are also added to the set of potential features.

2.6 Target Variable Analysis

The target variable, sell price, has a degree of completeness of 97.44%. The objective is to predict the missing 2.56%. Table 5 summarizes the key statistics of the target variable. Here, we see that the sell price ranges from 150 000 SEK to 65 995 000 SEK, with a mean around 3 m SEK. The standard deviation is relatively high ($> 50\%$), which speaks for high variability in the data.

Table 5: Summary Statistics of target variable

Statistic	**Value (SEK)**
Count	69 437
Mean	2 982 585
Standard Deviation	1 483 259
Minimum	150 000
25th Percentile	2 025 000
50th Percentile (Median)	2 650 000
75th Percentile	3 550 000
Maximum	65 995 000

Figure 6 displays the distribution of sell price and shows a right-skewed distribution, with most apartments priced around 1 - 3 m SEK and a long tail of higher-priced apartments. The box plot confirms the presence of outliers, that is, luxury apartments. This aligns with the summary statistics, that shows a large max value compared to the 75th percentile. This project uses the Root Mean Squared Error (RMSE), given in equation (4) as the primary evaluation metric.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{4}$$

Here, $\hat{y}_i$ is the predicted value for the i-th observation and y_i is the actual value for the same observation. The skewness can affect the RMSE, since it squared the errors, meaning that larger errors are more heavily penalised. Say the model predicts that the sell price of an apartment is 3 m SEK while the actual value is 65 m SEK, then the 62 m SEK error (squared!!) will dominate the RMSE calculation. This phenomenon makes RMSE sensitive to outliers in a skewed distribution, which potentially can provide a misleading impression of the model performance. A model may perform very well on most apartments, but will still have a high RMSE due to a few large errors on outliers.

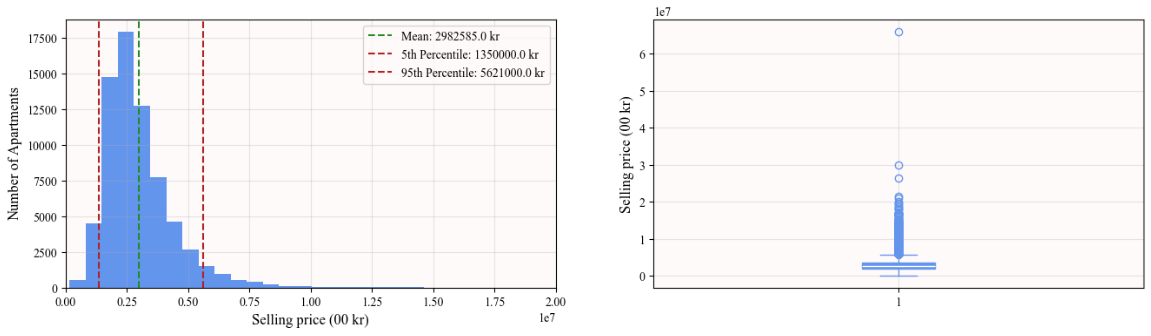

Figure 6: Distribution of target variable

To address this, we will use the log-transformed selling price as the target variable. This is done using the numpy library's `log1p` function which computes the log-transform according to equation (5). This ensures the transformation works for small values as well.

$$\text{log\_x} = \log(1 + x) \tag{5}$$

Figure 7 shows the distribution of the log-sell price, which has reduced skewness. This log-transformation stabilises variance (44%) and ensures that the errors are more evenly distributed across the price range.

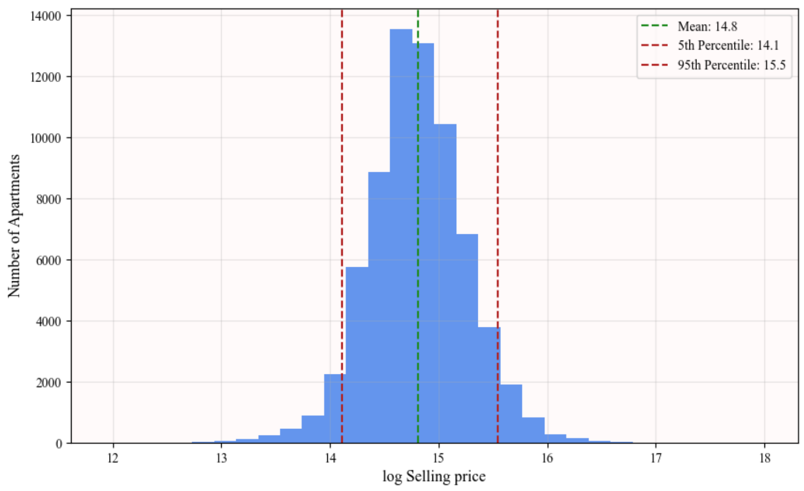

Figure 7: Distribution of target variable

2.7 Target-Feature Relationships

Let us begin by thoroughly explaining the processing that has been applied to each individual potential feature. Table 6 gives a summary of the columns that are selected for further analysis.

Feature name	**Processing**
agency_selling_price_mean	Apply target encoding, log_sell_price per agency_id
has_amenity $= \begin{cases} \text{has\_elevator} \\ \text{has\_elevator,balcony} \end{cases}$	Parse into individual amenities, create binary features for top amenities
log_asking_price	Apply log-transformation to reduce skewness
district	Numeric conversion
log_floor	Apply log-transformation to reduce skewness
log_housing_association_fee	Apply log-transformation to reduce skewness
distance_to_center	Computed as the distance from the coordinate of (latitude, longitude) to the coordinate of the city centre.
log_living_area	Apply log-transformation to reduce skewness
log_operating_cost	Apply log-transformation to reduce skewness
primary_area	Numeric conversion
rooms	Not processed
sell_year	Extracted from sell_date
sell_month	Extracted from sell_date
age	Computed as current year less construction_year
log_long_term_real_estate_debt_mean_debt	Apply log-transformation to reduce skewness
log_long_term_real_estate_debt_std_debt	Apply log-transformation to reduce skewness
log_total_loan_mean_loan	Apply log-transformation to reduce skewness
log_total_loan_std_loan	Apply log-transformation to reduce skewness
log_number_of_units_mean_units	Apply log-transformation to reduce skewness
log_number_of_units_std_units	Apply log-transformation to reduce skewness
log_total_living_area_mean_total_area	Apply log-transformation to reduce skewness
debt_trend	Log-variable, computed as the slope of log of the long term mean debt
loan_trend	Log-variable, computed as the slope of log of the total mean loan
debt_per_unit	Computed from equation (3)
loan_growth_rate	Computed from equation (3)
debt_to_loan_ratio	Computed from equation (3)

Table 6: Final features selected for analysis

Agency ID has high cardinality. We solve this by target encoding, where we map each agency to a mean selling price.

Amenities is made binary. We fill the NA-values with empty strings and parse out the top 30% amenity categories and then assign a value based on equation (6). Note that we create 30% extra (binary) columns, one for each amenity. It turns out some of the top values was an empty bracket, "[]", and "None". These values are removed.

$$\text{has\_amenity}_i, = \begin{cases} 1 & \text{if row i } \in \{\text{top 30\% amenities}\} \\ 0 & \text{else} \end{cases} \tag{6}$$

Essentially, we aim to capture the location effect of latitude and longitude. Therefore, we collect data for the coordinates of Gothenburg's city centre (approximately 57.7089, 11.9746), and compute the distance from the apartment location to the centre of Gothenburg by using the `geodesic` function in geopy.

Brokers description have 484 unique brokers description entries, and customer area description have 94 different customer area descriptions. These columns, along with locality, postcode, and street address are dropped due to redundancy, as we already have computed the agency mean selling price and the distance to city centre for each apartment.

The processing that has been applied to the remaining columns has either been described above or is trivial.

Figure 8 plots, for each feature, the processed feature against the log-transformed target. By doing this, we can understand the relationship between the individual feature and the log-target, which helps in the feature selection process.

The agent selling price mean shows a weak and noisy relationship with log-selling price. This feature captures the average price of apartments sold by each agent, but the high variance in the log-sell price for each agent suggests that the agent ID does not have a strong influence on price. Thus, it seems like agents are likely to handle a wide range of different properties.

The has elevator feature tells us that apartments in buildings with elevators tend to have slightly higher log-sell price compared to the apartments without elevator. This is reasonable since elevators are more common in newer or higher-end buildings. However, the overlap between the two groups is quite significant, which indicates a weak effect on the target. Has elevator and balcony displays similar relationship.

The log-asking price displays a strong linear, almost proportional, relationship to log-selling price, and is thus a very strong predictor of the target. This is likely the most important feature in the dataset. Rooms and log-living area also displays a quite strong positive linear relationship.

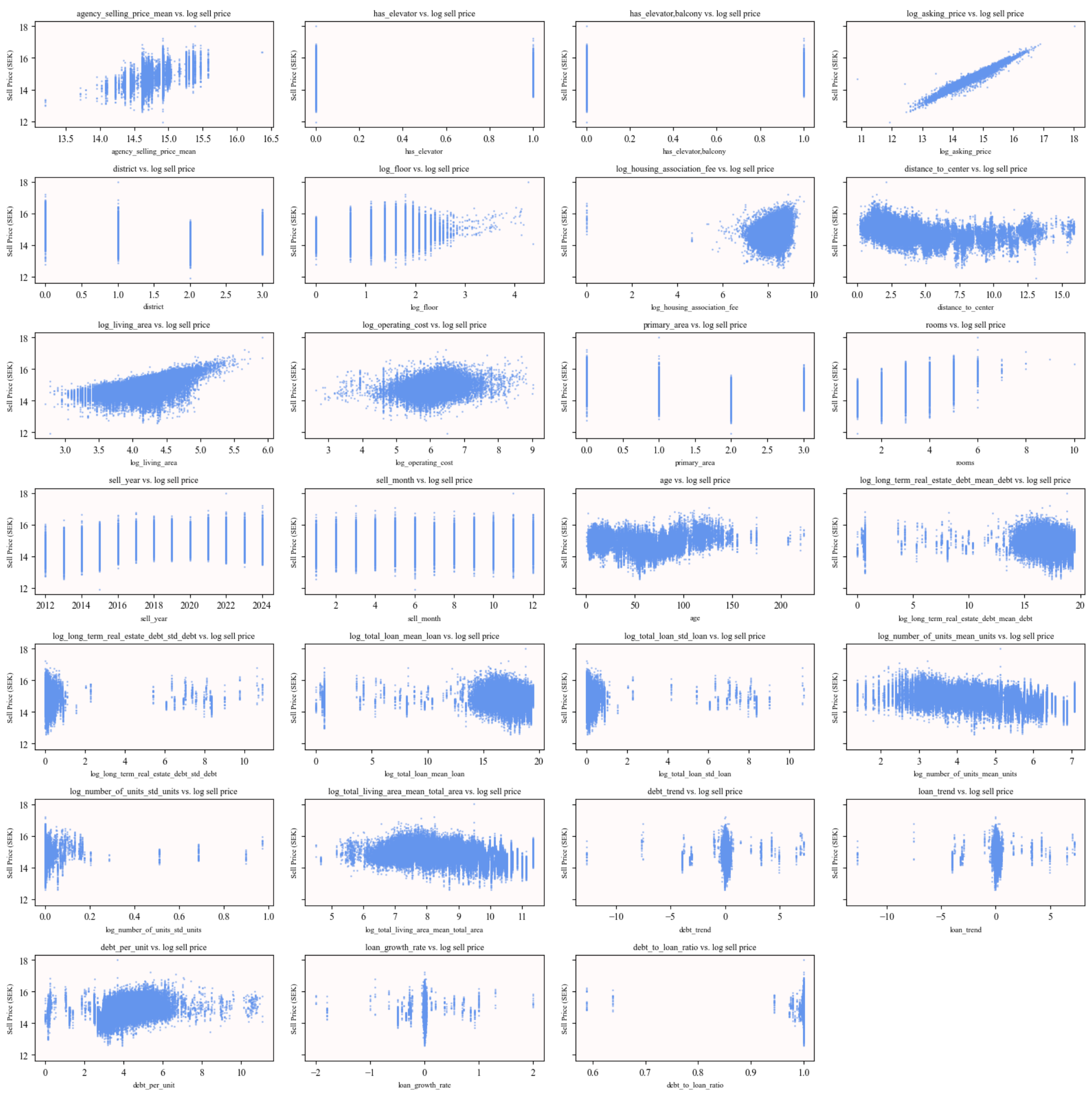

Figure 8: Relationship between log-target and selected features

As the distance to center increases, we can make out a relatively weak negative linear relationship, meaning that apartments closer to the city centre are more pricy. The same goes for log-mean no. of units and log-mean total living area. At first thought, one should have guess that al least one these relationships also should have been positive, i.e., that the greater the living area the higher the log-selling price. On the other hand, housing associations with a very large number of units may be older complexes with less exclusive units.

2.8 Correlation

Figure 9 displays the correlation between all features, including the log-target variable. The darker blue colours corresponds to negative correlations and red to positive correlations.

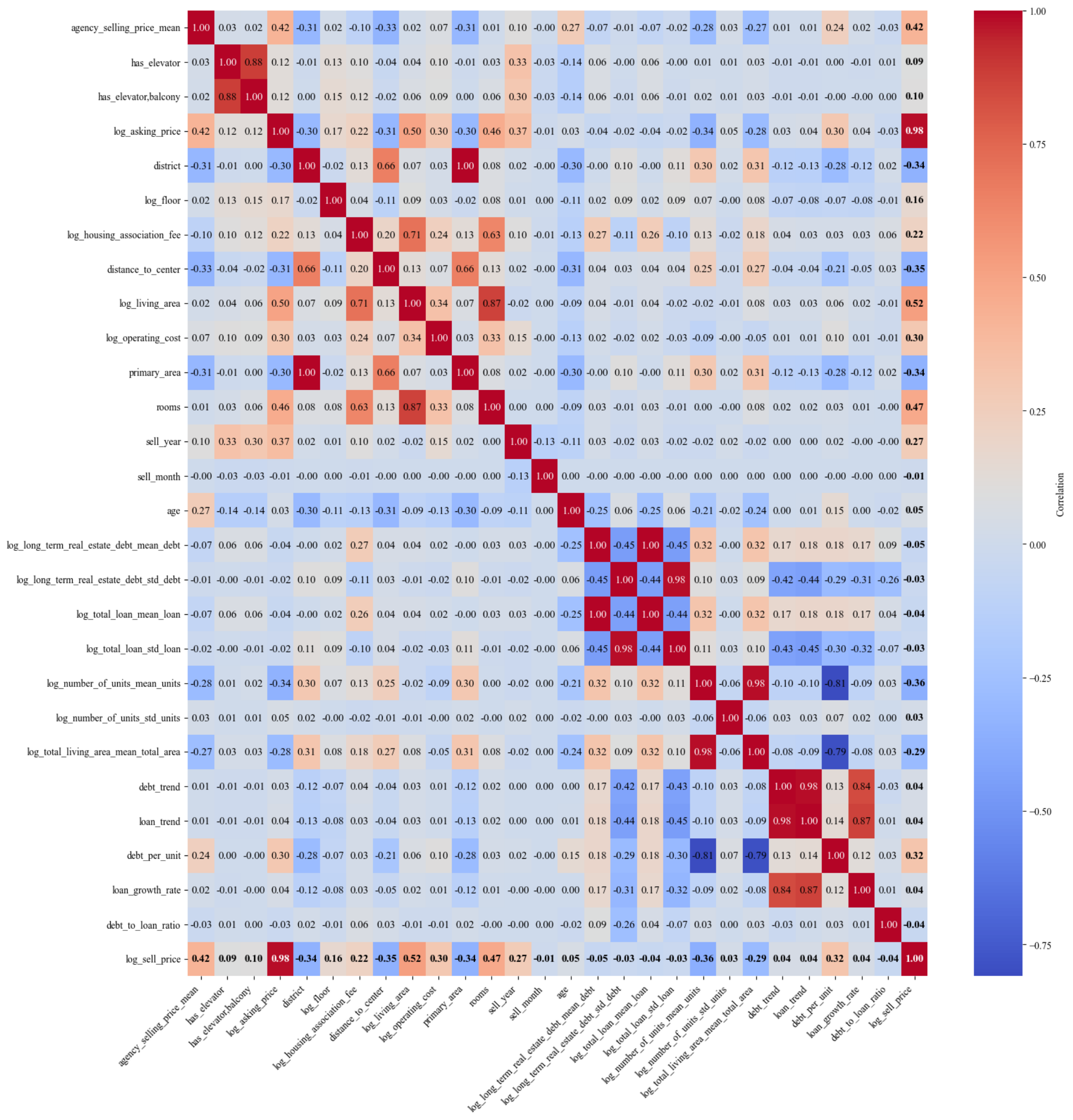

Figure 9: Correlation Heatmap

We see here that the target do correlate very strongly (0.98) with the log-selling price, as noted from the scatter relations plot. Moreover, log-living area shows a correlation of 0.52, and agency mean selling price, which to the eye looked like a weak and noisy relationship, has a correlation of 0.42 with the target. Rooms ($\rho = 0.47$) and debt per unit (ρ=0.32) also shows quite strong positive correlations.

District, Distance to center, log-mean no. of units, and log-mean total living area are all displaying a substantial negative correlation with the target.

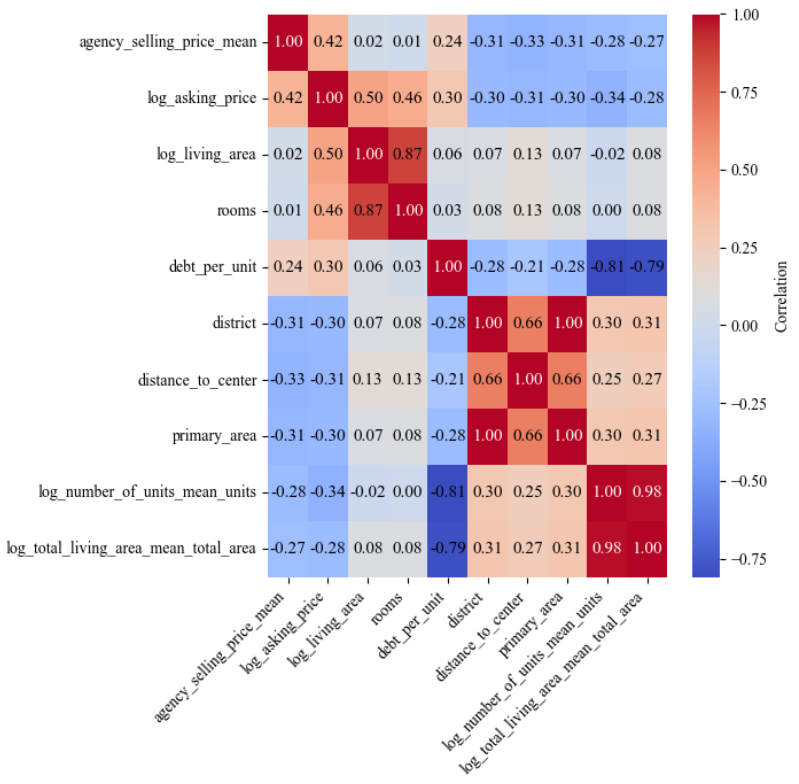

Figure 10: Correlation Heatmap (primary features)

Unfortunately, several of these features are also strongly correlated to each other, which is more clearly displayed in figure 10. This makes it hard to manually select features. Manual feature selection is also prone to large subjectivity and risks overlooking certain important criteria. Therefore, the following section explores some feature selection algorithms.

3 Model Selection

To prevent data leakage, we being by splitting the dataset into a training and test set. Performing feature selection on the whole dataset introduces leakage because the process uses information from the validation/test set to decide which features to keep. Hence, the selected features could be overly tuned to patterns in the whole dataset, including noise or outliers. This reduces generalisability to new data.

First off, we will perform supervised learning and so we cannot use the rows where the log-target is NaN. These are removed before dividing the dataset into training/testing. We call the data set with missing target values for the prediction set. The test set will consist of 20% and the training set 80% of the remaining data after removing the missing target values. Note that since the missing values is approximately 2% of the full data, this removal is very unlikely to affect the previously established relationships.

3.1 Handling of null and inf values

We will impute the remaining features with the median of the training set. We compute the imputation values on the training set only to avoid data leakage, and then apply it to the test and prediction sets. As all features are already processed (and are numerical), we apply the same treatment to all features.

Moreover, when log-transforming some features, we could potentially get some infinity/minus infinity values. For example, if the floor is -1, this value is going to be converted to $-\infty$. To address this, we compute the minimum of each log-transformed feature and replace the infinity values with $\min\{\text{feature}\} - 1$. This is dine before splitting the dataset.

3.2 Hybrid feature selection process

We will use a hybrid feature selection process, with three steps; a filter step, an embedded step, and finally a wrapper step. The filter step directly removes features with very low correlation to the target variable. This reduces the initial feature set. Next, we will apply random forest feature importance to rank the remaining features, since the random forest algorithm can capture non-linear relationships and interactions. Lastly, we will apply recursive feature elimination (RFE) with random forest to select the optimal subset of features. This should optimise for later applying RMSE on the validation set.

3.2.1 Filter step

We set the limit for dropping to $\rho \leq 5\%$. Table 7 shows the features that displays a correlation below 0.05 and thus are dropped.

Feature name
log_total_loan_std_loan
debt_trend
log_long_term_real_estate_debt_std_debt
age
sell_month
log_long_term_real_estate_debt_mean_debt
log_total_loan_mean_loan
loan_trend
log_number_of_units_std_units
log_sell_price
debt_to_loan_ratio
loan_growth_rate

Table 7: Dropped features through filtering

3.2.2 Embedded step

The random forest regressor creates multiple subsets of the training data by using bootstrap sampling. Each subset is roughly the same size as the original dataset but contains some duplicate rows and omits others. Around 60% of the rows in each sample will be unique.

For each sample, the algorithm builds a decision tree. Each tree splits the data into regions based on feature values to minimise the variance (MSE) of the taget. Since there is random feature selection at each node, this ensures that the trees that are built are diverse and not overly correlated, which is supposed to reduce overfitting.

The final prediction from the Random forest regressor is the average of predictions from all trees. We have applied 100 estimators in this case. This should make the feature selection more robust to noise compared to a single decision tree.

Feature importance is computed based on how much each feature contributes to reducing the variance of the target across all trees. For each split, we compute the impurity (variance reduction) achieved by splitting on that feature. The final importance is the normalised average impurity across all splits where that feature is used. Normalisation is used so that the sum of all feature importances equals 1.

The importance of a feature is the average reduction in variance across all splits where that feature is used, normalized so that the sum of all feature importances equals 1. Figure 11 shows the importance of the remaining features after filtering.

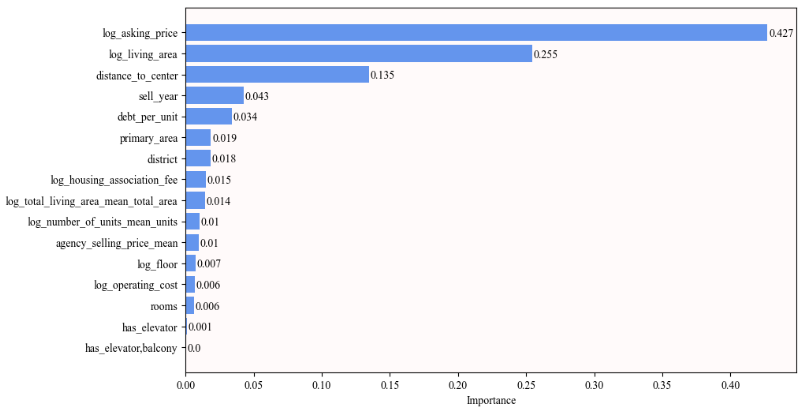

Figure 11: Random Forest Regressor Feature importance

The strongest predictor is the log-asking price, which aligns with its high correlation to the target. The second most important predictor, log-living area, tells us that the size of the apartment is important in relation to price. As well is the distance to center. Sell year and debt per unit probably captures non-linear effects that are not evident from the previous correlation analysis.

3.2.3 Wrapper step

We use RFE to select the most important features for predicting the target. RFE is a wrapper method that recursively removes the least important features until a specified number of features remain, we have set the limit at 5. It is called a wrapper method since it "wraps" around the random forest regressor, using its performance via feature importance to evaluate a subset of features. This is a lot more computationally intensive than, for example, filter methods, but it considers feature interactions and model performance.

The model removes one feature at a time, allowing it to re-evaluate the importance after each removal. Table 8 displays the final features that are selected.

Figure 12: Correlation of Selected features

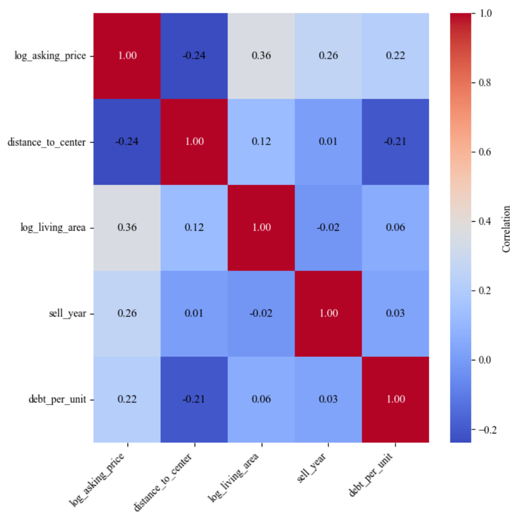

Table 8: Selected features

Feature name
log_asking_price
distance_to_center
log_living_area
sell_year
debt_per_unit

We can also note from the correlations in figure 12 that they are not too high. This is good since we want the features as correlated to the target as possible, but not to each other.

3.3 Choice of ML Algorithm

Now, it is time to start trying out different ML models, compare their performance and establish a baseline model. We will select linear regression as a baseline model, and random forest regressor as a primary model, as we have already used it for feature selection. We will also try out a gradient boosting regressor and a neural network regressor to see if these models can challenge the random forest regressor. Lastly, the champion model will be selected as the one with the lowest RMSE.

3.3.1 Baseline Model

A baseline model is a simple or naive model that can provide a reference point for performance. I have chosen a linear regression model, since it is very straight forward and interpretable. The target, log-selling price is a continuos variable, which makes regression a natural fit. Moreover, many of the selected features have a stronger or weaker linear relationship to the target, which suggests that a linear regression model will perform reasonably well. Table 9 displays the training error, and the test error in both log-transformed and standard scale.

	RMSE
Training RMSE (log_sell_price)	0.2439
Test RMSE (log_sell_price)	0.2455
Test RMSE (sell_price)	836 165.95

Table 9: RMSE for linear Regression model

The training and testing errors here are very close, differing only by 0.0016. This means that the model does not overfit, and that it generalises well to unseen data. However, when translating the log-RMSE on the test data to standard scale, we see a quite large number. Comparing this error to the summary statistics in table 5, where the mean selling price is around 3 m SEK, we see that an estimate of the selling price with this model would be of no practical use. If we estimate the apartment to cost 3 m SEK, it can in fact range in price between 2.2 m SEK to 3.8 m SEK. However, this gives us a baseline model.

3.3.2 Primary model

For the primary model, we implement a random forest regressor using the `sklearn.ensemble` library. The primary model is our initial choice of a strong model for predicting the sales prices. It is the starting point for the model comparison. Table 10 provides the RMSE for the training and testing set.

	RMSE
Training RMSE (log_sell_price)	0.0465
Test RMSE (log_sell_price)	0.1200
Test RMSE (sell_price)	434 433.99

Table 10: RMSE for Random Forest regressor model

Comparing table 10 to table 9, we see that all errors are much smaller. However, the training RMSE for the random forest regressor model is a lot smaller than the testing error, with a difference of 0.0735. This indicates overfitting during training. However, the standard form RMSE is still half as large as for the linear regression model.

3.3.3 XGBoost Regressor model

XGBoost, or extreme gradient booting (implemented using the `xgboost` library) is also a tree-based ensemble learning method. It differs from the random forest algorithm in that it uses boosting, whereas random forest uses bagging (bootstrap aggregating). Instead of building independent trees on random subsets of data and then average predictions to reduce variance, XGBoost builds trees sequentially, where each tree corrects the errors in the previous one. Table 11 provides the RMSE for the training and testing set.

	RMSE
Training RMSE (log_sell_price)	0.1358
Test RMSE (log_sell_price)	0.1436
Test RMSE (sell_price)	497 261.14

Table 11: RMSE for XGBoost regressor model

The training and testing errors are close, which indicates minimal to no overfitting, so the model generalises well to new data. However, the standard form RMSE is larger than for the random forest model.

3.3.4 Neural Network MPL Regressor

The MPLRegressor is a type of multi-layer perception (MLP), that is, a feedforward neural network with multiple layers of nodes. I have implemented it using the `sklearn.neural_network`. The MPLRegressor models non-linear relationships through hidden layers and activation functions (I have used relu) and learns a singe continuos function via gradient descent. Table 12 provides the RMSE for the training and testing set.

	RMSE
Training RMSE (log_sell_price)	0.1657
Test RMSE (log_sell_price)	0.1707
Test RMSE (sell_price)	558 834.23

Table 12: RMSE for XGBoost regressor model

This model also outperforms the linear regression basemodel, but has weaker performance compared to the ensemble learning models. However, it shows the least overfitting/error increase between the training and test errors (0.005). I made this model using only a two-layer architechture, which may not be "deep" enough to capture all patterns. This could be an explaination for the models "average" performance.

3.4 Champion Model

Comparing all four models, the random forest regressor shows the smallest test error, which is highlighted in figure 13. Thus, it is chosen as out champion model. In the coming section, I will use tuning to try and reduce the overfitting and also explore the impact of feature selection.

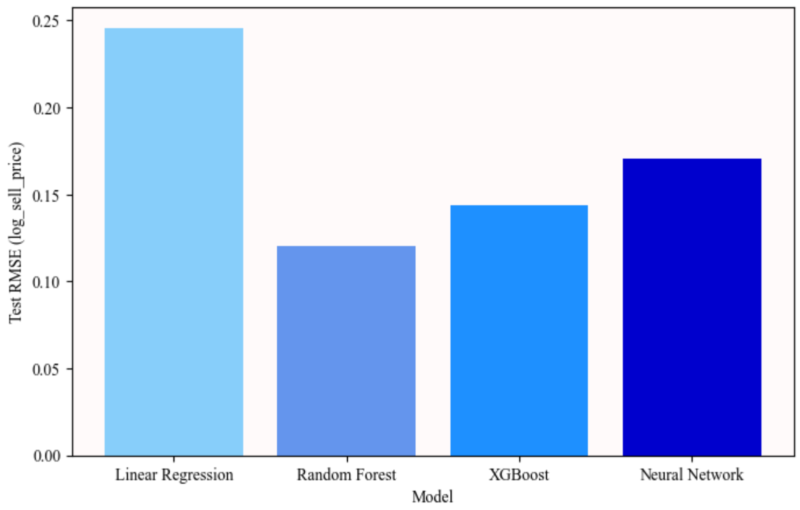

Figure 13: Model Comparison: Test RMSE

4 Model Testing and Tuning

Table 10 describes the training and testing errors before tuning. To maximise the models performance, I will perform a comprehensive parameter tuning and also investigate if adding more features will enhance model performance.

4.1 Testing Data and discussion of historical data

The testing data that is used is the remaning 20% after removing the prediction set, and selecting 80% for training in the previous section. That is, we use all the data (all years) in both training and testing.

Older data points may not be as valid as new data points since the underlying market dynamics have changed since 2012 which is the earliest data entries. The house prices have generally increased, both due to natural causes (inflation) and due to competitive dynamics and housing shortages. However, older data points can still be useful if the model is able to capture these shifts. Since we are including sell year as a feature, it will help the model contextualise older points, which can improve generalisation and robustnes.

4.2 Hyperparameter Tuning

Table 13 displays the hyperparameters and value ranges that we are going to tune. The higher the no. of trees are, the more variance is reduced, which sounds promising. However, more trees also increases the computational cost, so we will have to find a middle ground. The max depth controls overfitting. If we use a higher number of min_samples_split, this should also reduce overfitting. Higher values of min_sample_leaf contributes to higher robustness. Max features introduces randomness into the model, which also should reduce overfitting. Lastly, the Bootstrap parameter can assume the values *True* and *False*.

Hyperparameters	**Range**	**Description**
n_estimators	$[100, 200, 300, 500]$	No. of trees
max_depth	$[10, 20, 30, 40, None]$	Maximum depth of trees
min_samples_split	$[2, 5, 10, 20]$	Minimum samples to split a node
min_samples_leaf	$[1, 2, 4, 8]$	Minimum samples in a leaf
max_features	$[\text{sqrt}, \log 2, 0.5, 0.8, 1.0]$	Number of features to consider per split
bootstrap	[True, False]	

Table 13: Hyperparameters

We will perform `Randomised search 5-fold cross validation`, which samples random combinations of hyperparameters, which allows us to explore a large parameter space. The result from the tuning is summarised in table 14. The best cross-validation RMSE for the log-target was 0.1194. This is a robust estimate of the model's performance on unseen data, averaged on the 5 folds used in the random search cross validation.

Hyperparameters	best value
n_estimators	500
max_depth	30
min_samples_split	5
min_samples_leaf	2
max_features	0.5
bootstrap	False

Table 14: Hyperparameter tuning result

Re-training the model and testing it provides the results given in table 15. We can see that the training RMSE is higher than for the original model, displayed in table 10. We also get a lower standard form testing error of approximately 417 000 SEK.

	RMSE
Training RMSE (log_sell_price)	0.0506
Test RMSE (log_sell_price)	0.1173
Test RMSE (sell_price)	417 938.66

Table 15: RMSE for tuned Random Forest regressor model

4.2.1 Overfitting introduced in tuning

There is still a quite substantial difference between the log-training and log-testing error, implying that the model still overfits, although lesser than previously since the training error is higher and the testing error is lower. Thus, we can be sure that we are not introducing additional overfitting when tuning the hyperparameters.

Additionally, we note that the CV RMSE (of 0.1194) is very close to the actual test RMSE (of 0.1173). This indicates that the cross-validation process accurately reflects the model's ability to generalise. However, it is higher than the training RMSE, which again confirms some overfitting.

4.3 Re-introduction of features

I was curious as to how re-introducing all features from table 6 would affect the model. These features, even though some of them have a very weak correlation to the target, are thoroughly processed and are ready to use in modelling. It turns out that we achieve an even lower RMSE for both training and testing. This is shown in table 16.

	RMSE
Training RMSE (log_sell_price)	0.0237
Test RMSE (log_sell_price)	0.0997
Test RMSE (sell_price)	365 940.11

Table 16: RMSE for tuned Random Forest regressor model (all features)

The achievement of a lower RMSE when using all features must point to the fact that these actually introduce additional information, and are not completely uncorrelated, noisy, or have heavy collinearity.

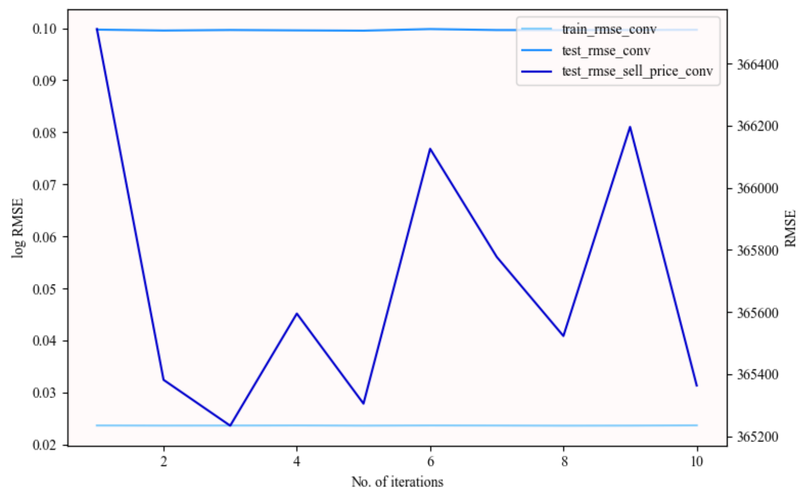

Figure 14: RMSE stability through 10 iterations

Additionally, random forest algorithms in general are quite resistant to irrelevant features, they just do not get picked at each split. Therefore, the algorithm are able to take advantage of the additional information provided, while sorting out the irrelevant information. Sometimes simple features like floor or has elevator can help the model correct for hidden differences that are not captures by continuos features. For example, the floor and whether the building has an elevator could affect the price in ways that are not captured by log-living area. There may also exist synergies between features, for example, the distance to center may only matter for small apartments (since maybe larger apartments all lie within close distance to the centre). Or, the debt per unit may matter more for certain sell years. The random forest regressor algorithm is able to learn such interactions automatically, given that all the features are present in training.

Since we are using a quite large number of trees (500), the training process is quite slow and computationally expensive. However, over 10 iterations, we can see that the RMSE's are quite stable, which speaks for the stability of the model. An RMSE of approximately 365 000 SEK starts to be manageable in a real setting. For example, when participating in a bidding/auction for an apartment an unexpected price increase of around 300 to 400 k SEK is not unusual.

4.4 Performance Gain in Tuning

The final model performance after tuning has a RMSE of 365 940.11 SEK (table 16). It incorporates all processed features displayed in table 6. We managed to get a reduction of 434 433.99 - 365 940.11 = 68 493.88 SEK, which is quite substantial in this setting.

5 Robustness

5.1 Examination of Extreme Values

To investigate the impact of extreme values, probably due to outliers, data errors or statistical artifacts, we begin by identifying potential outliers in the features and the target by using z-scores. We already know how the model performs with the extreme values, so we will remove these and evaluate, again, the model performance.

The z-score is given by equation 7. Typically z-scores higher or lower than 3 indicates a potential outlier.

$$z = \frac{x - \mu}{\sigma} \tag{7}$$

Table 17 displays the model result without outliers. There is a very small difference noted in the training error. This implies that the model fits both datasets (with and without extreme values) equally well on training data. This shows no signs of the model "freaking out" during training.

	RMSE
Training RMSE (log_sell_price)	0.0229
Test RMSE (log_sell_price)	0.0945
Test RMSE (sell_price)	321 523.39

Table 17: RMSE for tuned Random Forest regressor model (all features)

The test log-RMSE, however, decreases which shows more clearly in the standard form RMSE which decreases to approximately 320 k SEK. This suggests that the extreme values do have an impact on the model testing error, especially on unseen data. This again, could be a result of the previously observed overfitting behaviour.

5.2 Predictive Ability in the case of Missing Features

We examine the predictive ability by simulating the complete loss of a feature one-by-one. Table 18 shows the results. Most features are, surprisingly, showing very high jumps in RMSE if a particular feature where to experience data loss. Therefore, complete data is highly important for the model. At this point, it would give better performance to just eliminate that column completely.

Feature name		log-test RMSE	Δ test RMSE
agency_selling_price_mean		0.1572	274 933.91
has_amenity=	has_elevator	0.1236	180 562.37
	has_elevator,balcony	0.1236	180 562.37
log_asking_price		0.8501	1 782 546.71
district		0.1534	213 314.15
log_floor		0.1321	216 396.39
log_housing_association_fee		0.1257	170 802.55
distance_to_center		0.1871	279 228.96
log_living_area		0.2916	776 921.0
log_operating_cost		0.1261	187 131.78
primary_area		0.1531	212 433.76
rooms		0.1552	319 563.14
sell_year		0.2882	620 402.63
sell_month		0.1252	183 037.88
age		0.1524	223 468.88
log_long_term_real_estate_debt_mean_debt		0.1299	197 639.78
log_long_term_real_estate_debt_std_debt		0.1276	191 258.87
log_total_loan_mean_loan		0.1297	197 283.99
log_total_loan_std_loan		0.1279	191 180.47
log_number_of_units_mean_units		0.1358	197 423.04
log_number_of_units_std_units		0.1240	181 365.43
log_total_living_area_mean_total_area		0.1354	213 064.13
debt_trend		0.1256	185 243.15
loan_trend		0.1253	183 588.7
debt_per_unit		0.1376	221 989.55
loan_growth_rate		0.1252	183 587.15
debt_to_loan_ratio		0.1248	184 945.1

Table 18: Simulated feature loss

6 Evaluation

6.1 Model Performance

The model displays stronger and weaker sides. On the positive note, the final model has a quite low testing RMSE, of around 360 000 SEK. This value is stable, and is a reasonable uncertainty in a real-world apartment pricing setting. On the other hand, the model seems to be overly sensitive to large cases of inaccurate data, e.g., when a whole column is set to zero, the performance decreases drastically.

Predicting the sell prices yields the distribution presented in figure 15. It looks fairly similar to that of figure 6.

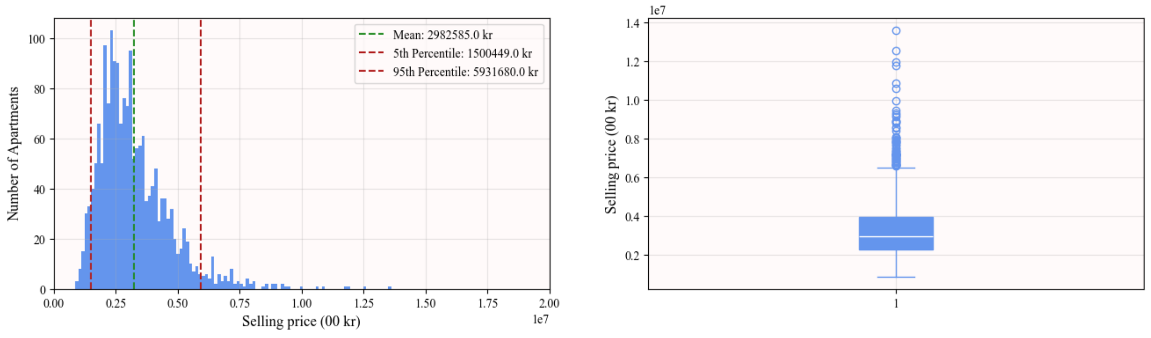

Figure 15: Distribution of predicted sell price

6.2 Model Complexity

The model complexity is justified since the relationship between features can be both non-linear and interdependent. Simpler models, like the linear regression baseline model cannot capture this.

6.3 Predictive Power of Features

The log-asking price has the highest predictive power, following the features listed in table 8. However, the model may capture relationships, such as for example how debt per unit may matter more for certain sell years, which makes weaker predictors important as well. All features importances are listed in table 19.

Surprisingly, the amenities are ranked the lowest. Balcony and elevator are personally important features that I would be willing to pay more for.

Feature name	importance weight
log_asking_price	0.351375
log_living_area	0.210241
distance_to_center	0.105353
sell_year	0.052181
rooms	0.051885
district	0.043362
primary_area	0.041888
age	0.035571
debt_per_unit	0.022928
log_number_of_units_mean_units	0.013247
log_housing_association_fee	0.011773
log_total_living_area_mean_total_area	0.011187
agency_selling_price_mean	0.008810
log_floor	0.005305
log_long_term_real_estate_debt_mean_debt	0.005303
log_total_loan_mean_loan	0.005199
log_operating_cost	0.003779
log_total_loan_std_loan	0.003463
log_long_term_real_estate_debt_std_debt	0.003415
sell_month	0.003395
debt_trend	0.002637
loan_growth_rate	0.002586
loan_trend	0.002543
log_number_of_units_std_units	0.001169
debt_to_loan_ratio	0.000708
has_amenity= { has_elevator	0.000418
has_elevator,balcony	0.000279

Table 19: Feature Importance

6.4 Model Degradation

Features like log-asking price, debt trend, and loan trend do have temporal dependencies and could potentially change drastically if market conditions, inflation, or consumer preferences change. This may result in performance degradation.

We could potentially mitigate this by regularly retrain the model with updated data each quarter or each year. There is also an option to introduce further techniques that detects drifts, such as the population stability index (PSI) test, that tests the distribution of at specified time points.

6.5 Cross-sectional Model Analysis

To analyse whether the model performs equally well through all of real estate or if its performance is superior for some specific segment, we check the distribution of errors across different segments. I chose to look at three different segments: the distance to Gothenburg centre, the selling year, and the asking price. Each segment is binned into smaller subsegments and the RSME is computed. The log and standard RMSE for each segment and subsegment is shown in figure 16.

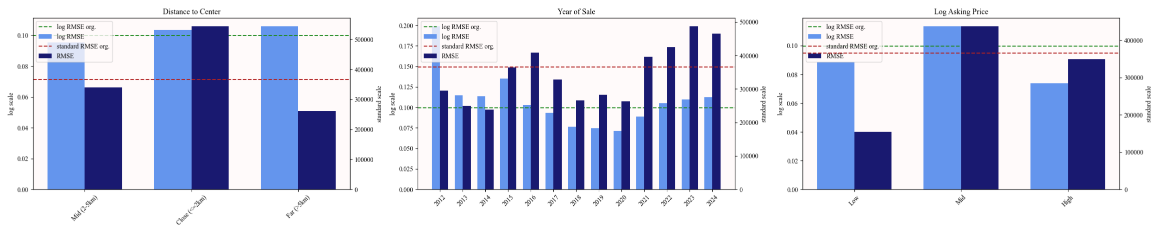

Figure 16: Error distribution over different subsets

The log RMSE is relatively stable for the different subsets of the distance to centre feature. However, in terms of the standard form RMSE, the middle bar (close distance to centre) is a lot higher than that of the original models. Central Gotherburg is likely a high-value, competitive market, and such large errors could lead to significant mispricing. In turn, this could affect competitiveness and profitability.

In terms of years of sale, the model performance varies a lot across different years. It performs the absolute worst during 2012, but also it does not do very well during more recent years. This is concerning since the recent year represent the current market conditions.

The mid segment performs the worst out the low, mid, ask high log-asking prices. As the mid asking price category is main portion of this column, this represents a large portion of the market. Higher errors are directly correlated to larger mispricing, but it could also mean that this segment has a lot more different characteristics than the low and high asking price categories.

7 Conclusion and Final Word

The development of the Gothenburg Housing Market Prediction model has followed a rigorous process, including careful feature engineering, methodical model selection, and robust validation procedures. The final model demonstrates solid performance on test data based on the RMSE metric and satisfies key regulatory and engineering requirements for transparency, reproducibility, and interpretability. The Random Forest Regressor provides a functional baseline for predicting apartment sale prices in Gothenburg, but its inconsistent performance across key segments and sensitivity to missing features poses significant risks. These errors could lead to substantial financial losses or missed opportunities in high-value and competitive market segments.

To further strengthen the model, I would recommend incorporation of additional data such as proximity to public transport, school quality, or economic indicators, that can add predictive power to the specific segments. Additionally, we could introduce prediction intervals to give users an estimate of confidence in each price prediction, which is particularly useful for risk-sensitive decisions.