Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Håvard Faxvaag Johnsen

From Listings to Valuations: Integrating LLM-Based Feature Extraction into Automated Valuation Models

Master's thesis in Cybernetics and Robotics
Supervisor: Ole Morten Aamo
Co-supervisor: Ulf Jakob Flø Aarsnes
June 2024

NTNU
Norwegian University of
Science and Technology

Håvard Faxvaag Johnsen

From Listings to Valuations: Integrating LLM-Based Feature Extraction into Automated Valuation Models

Master's thesis in Cybernetics and Robotics
Supervisor: Ole Morten Aamo
Co-supervisor: Ulf Jakob Flø Aarsnes
June 2024

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Preface

This master's thesis is written as part of the study program Cybernetics and Robotics at the Norwegian University of Science and Technology (NTNU) in Trondheim. The thesis is worth 30 credits and the work presented in this project was carried out during the spring of 2024. This thesis is given by *solgt.no* under the supervision of Ulf Jakob Flø Aarsnes. The project has given me useful insight into the Norwegian real estate market and access to the latest technology within the field of artificial intelligence. I would like to thank my external supervisor Ulf Jakob Flø Aarsnes, and my internal supervisor Professor Ole Morten Aamo for sharing their expertise and feedback during the work on the thesis.

Håvard Faxvaag Johnsen

Trondheim, June 2024

Abstract

The use of automated valuation models for price estimation in the real estate industry is a valuable and important asset, but precise predictions remain a challenging task. Traditional valuation models are mainly based on structured data from housing advertisements to predict housing prices. However, these advertisements also have listing texts that hold important information about a housing unit's condition and attractiveness. This thesis aims to utilize large language models to extract valuable features from the textual part of the housing advertisement. Furthermore, by combining the extracted features with the structured information, it is possible to develop a new automated valuation model that might offer enhanced price predictions. It was found that the large language model's ability to return accurate feature values from the listing text was overall high, but could vary based on what information that were extracted. Further observations revealed that the listing text's contents varied considerably from one housing advertisement to another, making a substantial amount of information unavailable for the language model resulting in features with low variation in the data. Despite this, the valuation model improved by 20.14 % for RMSE and 17.46 % for MAPE when the features extracted by the large language model were included, compared to the model without these features. This shows significant potential for implementing large language models for feature extraction to enhance the accuracy of valuation models used in the real estate industry.

Sammendrag

Bruken av automatiserte verdsettingsmodeller for prisestimering er et viktig verktøy i eiendomsbransjen, men nøyaktige estimeringer har vist seg å være en utfordrende oppgave. Tradisjonelt har verdsettingsmodeller benyttet strukturerte data fra boligannonser for å estimere boligpriser, men disse annonsene inneholder også mye nyttig informasjon i tekstbeskrivelsene av boligene. Dette prosjektet har som mål å bruke språkmodeller til å hente ut nyttige parametere fra tekstbeskrivelsene av boligannonser. Videre skal denne informasjonen kunne brukes sammen med de strukturerte dataene for å øke treffsikkerheten til verdiestimeringen av boligen. Det viser seg at språkmodellen returnerer data av høy kvalitet fra tekstbeskrivelser, men med noe variasjon basert på hvilken informasjon som ønskes å hentes ut. Undersøkelser viser imidlertid at på grunn av formuleringer og innholdet i tekstbeskrivelser i boligannonser er det en del informasjon som ikke er tilgjengelig i alle annonser. Selv med noe varierende data viser det seg at nøyaktigheten til verdsettingsmodellen øker med 20.14 % for RMSE og 17.46 % for MAPE når man bruker parametere hentet ut av språkmodeller, sammenlignet med uten. Dette viser det store potensialet språkmodeller har for å hente ut informasjon fra tekstbeskrivelser som kan brukes å øke presisjonen til verdsettingsmodeller.

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence. 7

ANN Artificial Neural Network. 6

API Application Programming Interface. 8

AVM Automated Valuation Model. 1

BERT Bidirectional Encoder Representations from Transformers. 5

GPT Generative Pre-trained Transformer. 5

IE Information Extraction. 5

IF Isolation Forest. 41

LLM Large Language Model. 1

MAE Mean Average Error. 43

MAPE Mean Average Percentage Error. 43

ML Machine Learning. 7, 9

MSE Mean Squared Error. 11

NLP Natural Language Processing. 4

OHE One-Hot Encoding. 12

RMSE Root Mean Squared Error. 43

SHAP SHapley Additive exPlanations. 48

Chapter 1

Introduction

This thesis investigates how the performance of *automated valuation models* (AVM) could be enhanced by introducing *large language models* (LLM) for feature extraction. The motivation behind the work done is described in this chapter by discussing the importance of AVMs and the real estate market in Oslo, Norway. Furthermore, the problem description is formulated, before the method and report structure are described.

1.1 Background and Motivation

The private real estate market has seen significant growth over the last years and making accurate price predictions is an important asset for both buyer and seller. Traditional AVMs are mainly based on structured data about the housing unit such as total area, build year, and number of rooms to estimate the housing value. This information is often based on structured information from publicly available data such as housing advertisements from *finn.no*. However, these advertisements also have listing texts which contain a considerable amount of unstructured data such as text descriptions and captions from pictures. Even though the unstructured data isn't necessarily readily quantifiable, it holds important information about the housing's condition and attractiveness. Sophisticated methods using LLMs are required to analyze, extract, and quantify the important information the unstructured data contains.

Since its start in 2020 has *solgt.no* delivered services related to the real estate market. They started with iBuying[1] before they in May 2024, changed this business model

[1]Business model for buying properties, then flip them for a higher price.

to deliver software as a service (SaaS) where they offer several insightful features for the real estate market. In connection with this, they want to investigate the possibilities of using LLMs to analyze and extract important information from the listing texts in housing advertisements. This information will be combined with existing structured information and then be used to develop an AVM, which their customers can use to get a precise estimated price on a housing they are interested in.

This is a new field and the combination of using LLM extracted features, and existing features for price prediction, has not been tried to the same extent before.

1.2 Problem Description

This project aims to investigate the degree to which unstructured data, such as listing titles, descriptions, and picture captions from housing advertisements, can be integrated and utilized effectively to enhance the performance of an AVM in estimating the turnover value of housing.

To extract the unstructured data, a program that takes the listing text of a housing advertisement as input and uses an LLM to return quantitative values based on the advertisement's content, needs to be developed. Prompt engineering based on available literature should be performed to make the LLM return precise and consistent values. To ensure the quality of the values, the feature extraction process should be validated and compared to the actual information.

Housing advertisements consist of both structured and unstructured data. *Solgt.no* has a large dataset with structured information from the advertisements, used to develop their AVM. The objective is to validate the effectiveness and reliability of a new AVM, by integrating both structured and unstructured data from the housing advertisements. This includes the best combination of these types of data to achieve the most precise housing valuations. The goal is both to validate the new AVM's performance and investigate how the combination of structured and unstructured data can enhance the accuracy of the valuation. The list below summarizes the problem with additional information, while Figure 1.1 illustrates the process.

- Perform a literature study on methods for structuring unstructured information using LLMs including information extraction and prompt engineering.
- Develop a pipeline that accesses OpenAIs GPT-models, takes housing advert-

isement as input and returns quantitative values.

- Perform prompt engineering based on the available literature, to receive quality data from the LLM and collect data from the pipeline.
- Validate the effectiveness and reliability of the LLM.
- Develop an AVM that uses structured data from *solgt.no* and unstructured data from the LLM.
- Validate the effectiveness and reliability of the AVM.

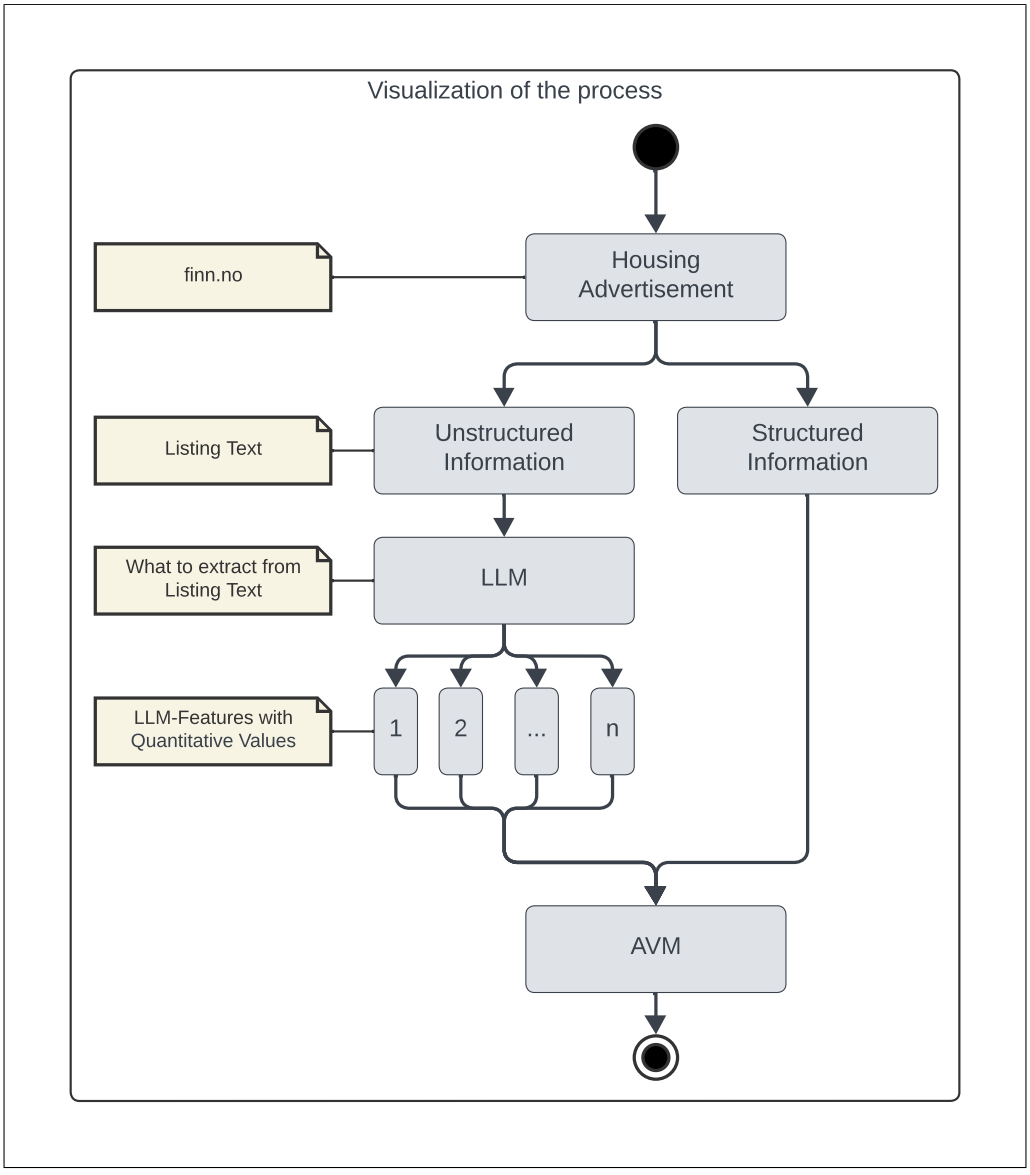


Figure 1.1: Visualization of the process.

1.3 Delimitations

This study is limited to using OpenAI's GPT models for information extraction from the listing texts. There are several other options when it comes to LLMs and other Natural Language Processing (NLP) methods, but due to accessibility, performance, and what's clarified in the problem description, other options are not considered in this thesis.

1.4 Structure

The report is structured with respect to the scope of this thesis. First is the relevant theory described in Chapter 2, before explaining the datasets in Chapter 3. The methodology is divided into two phases, the first one for the LLM in Chapter 4 followed by results in Chapter 5, and the second for the AVM in Chapter 6 followed by its results in Chapter 7. Further are the discussion and conclusion in Chapter 8 and 9.

The work done in this thesis is partially based on observations from the preliminary project carried out in the fall of 2023 [1]. Some similarities will therefore occur in the following sections:

- Section 2.2 Fundamentals of LLMs
- Section 2.3 Fundamentals of Machine Learning
- Section 8.3 Sustainable Development Goals

Chapter 2

Theory

This chapter presents a literature review and foundational theories regarding the capabilities of LLMs in information retrieval and AVMs' usage in housing price prediction. It begins with a thorough review of existing literature in the field, highlighting key observations within language models and machine learning. In addition, the theoretical framework and other observations in the same field are derived.

2.1 Literature Review

The idea of using computers to mimic human language using NLP has a long history. In the 1960s Joseph Weizenbaum developed the first chatbot called ELIZA [2], and was a simple simulated conversation program between human and machine [3]. The NLP technology has been significantly developed, with modern LLMs such as BERT[1] and GPT[2] demonstrating unique and powerful abilities for both text understanding and generating.

Within the GPT series, the OpenAI GPT-3 model contains 175 billion parameters and has shown strong performance on various NLP tasks, and even matched fine-tuned systems in some cases [4]. These newer and larger-sized models enable learning situations such as zero-shot and few-shot, with impressive results in Information Extraction (IE) tasks [5]. The newer and bigger GPT-3.5 and GPT-4 models used in ChatGPT[3], outperform the older GPT-3 model giving a greater potential for

[1]Bidirectional Encoder Representations from Transformers is a model developed by Google.
[2]Generative pre-trained transformers is a model developed by OpenAI.
[3]Used in ChatGPT

information retrieval from real estate advertisements [6].

Literature states that fine-tuned models for specific tasks usually outperform the general pre-trained models [7]. The fine-tuning requires high-quality data for the particular task to better understand the process and context, which leads to effective and accurate results. Narrowness in the specific task, on the other hand, will limit the model's versatility by only being able to extract one type of information. This means that if several features were to be extracted, it would be needed one model for each feature.

Prompt engineering has proven to be important for getting precise responses with high quality from the GPT models [8]. This approach includes formulating and designing the prompt[4] in a way that will result in the desired output. The prompt will guide the model more effectively and is seen as a good alternative for fine-tuning models by unlocking the potential in the GPT models for a certain task [9].

The idea of feature extraction from the textual part of housing advertisements has been tried before both with and without the use of LLMs [10] [11]. The preliminary project showed promising results when extracting values from the textual part, but a recurring error was the inconsistency of the model output [1].

The article "Fine-tuning BERT for a regression task: is a description enough to predict a property's list price?" by Anthony Galtier demonstrates the use of LLMs to extract information from the textual part of the housing advertisements, and using this info to predict the housing units price with strong results [12]. As the title states, only the textual information from the listing texts was used, not additional structured information about the housings. To the best of knowledge, no other studies have been found to combine both features extracted from listing texts, and structured data about the housing unit for price estimation to the same extent as this thesis.

Earlier stages of price estimation using AVMs were based on hedonic price models where the price is expressed by a function with characteristics such as the area in square meters, number of rooms, and proximity to public transport [13]. Literature from the early 2000s suggested that Artificial Neural Network (ANN) have great potential compared to the traditional hedonic models [14].

In later years, when it comes to housing price prediction based on existing data, ANNs and non-linear machine learning techniques have outperformed traditional regression models, despite some inconsistency in the results [15]. Even though these

[4]Input

models have shown good results, some skepticism based on the inconsistency of the results has been discussed [16].

Several ML models have been used with varying success. Still, the techniques that stand out with good performance are model algorithms such as *Random Forests*, *XGBoost*, and *LightGBM* [17]. All different models have their pros and cons, and when combined we have a technique called *Stacked Generalization* or *Stacked Regression Models*[5]. This technique has been slightly better at predicting housing prices than each model individually [17] [18].

In addition to the use of a strong-performing model, has use of additional features for improved price estimation also had an effect on the output. Literature suggests that the inclusion of condition attributes significantly increases the model's performance [19]. On the other side, the inclusion of energy labels won't affect to model output to the same extent as the housing standard [20].

2.2 Fundamentals of LLMs

Within the field of *artificial intelligence* (AI), LLMs are specialized models for understanding and generating text that mimics natural language. These models are trained on extensive amounts of data, enabling them to predict the next word in a sentence or sequence with impressive accuracy. Based on input received from the user, the models can then generate content- and context-relevant text, making them a powerful tool for linguistic tasks and applications. The models' learning efficiency and adaptability are central to developing intuitive human-like AI systems.

Different models have different areas of use within NLP tasks and are usually divided into categories such as masked or generative models. The first is focused on prediction tasks and context understanding, based on the training data, while generative models are designed to generate new data from the training data, such as text sequences. The field of NLP also uses a great variety of other techniques and model types for different language tasks.

2.2.1 GPT Models

Thanks to ChatGPT, the GPT models are one of the most used and well-known LLMs. The GPT models represent a significant advancement in NLP based on

[5]Technique that combines several models.

the Transformer architecture, first introduced by Google in the paper "Attention is All You Need" [21]. This architecture is designed around the idea of attention, a mechanism within the model that weighs the importance of different words within the same input to a varying degree. This enables models contextual understanding, capturing dependencies and relationships between words in a sequence, regardless of their distance from each other [21].

A part of this thesis aims to utilize the capabilities of the newer GPT models from OpenAI such as GPT-3.5-Turbo and GPT-4. The only way of accessing these models is through OpenAIs *Application Programming Interface* or API for short. In late January 2024, the newest model in the GPT-3.5 series *gpt-3.5-turbo-0125* was released, but there are several other variations of both GPT-3.5 and GPT-4 available as well [22]. What sets them apart is how the API is accessed and the cost associated with their use.

Earlier models access the *Completions API* endpoint, but the later models access the *Chat Completions API* endpoint which has a different interface. The first provides completion for a single prompt as input and is what was used in the preliminary study. The last requires the input to be structured in a way that represents the message history of a conversation, including how the system should behave and the content of the message [23].

2.2.2 Tokens

Within the field of NLP, models process text in parts called tokens, which commonly represent a symbol, word, or part of a word. Language models today are based on these tokens to analyze and find context and relations in a sentence [24]. OpenAI also uses tokens to price the models according to the number of tokens in an input message, and the corresponding output response. There are big differences in pricing between each model, where the *gpt-3.5-turbo-0125* model costs \$0.50/1M tokens for the input, and \$1.50/1M tokens for the output, while other could cost 10-20 times as much. Table 2.1 below lists the newest and most common GPT models, where prices are given per million tokens[6] [25].

[6]Pricese are as of June 2024

Table 2.1: GPT model pricing per 1 million tokens.

Model	Input	Output
gpt-4o	$5	$15
gpt-4-turbo	$10	$30
gpt-3.5-turbo-0125	$0.50	$1.50

2.2.3 Prompt Engineering

Prompt engineering is an approach to leveraging and improving the performance of LLMs for specific tasks. It involves creating the prompt that's given to a model in such a way that will result in the desired output. With good prompt engineering, the performance and efficiency of the model can increase significantly potentially reaching the abilities of cutting-edge fine-tuned[7] models. Where fine-tuning often limits the model for a single specific task, prompt engineering allows the model to perform multiple tasks without needing a separate fine-tuned model for each task [9]. Prompt engineering is a critical part of this thesis and is crucial for the results and consistency. The formulations of the inputs are described later in this thesis in 5.

2.3 Fundamentals of Machine Learning

Within AI, *Machine Learning* (ML) is the field of study referring to using statistical tools and algorithms that learn and understand the patterns in data. By generalizing the patterns from the training, ML algorithms can perform tasks on unseen data without explicit instructions [26]. These tools used to train the algorithm are mainly divided into two categories, *supervised* and *unsupervised*. In unsupervised learning, the algorithms use a dataset containing several features and then learn the properties and structure of the dataset. Supervised learning algorithms also use a dataset with several features, but the features are also given a label or target [26]. As this thesis aims to predict housing prices based on existing data, it falls under the category of supervised learning.

The overall goal of supervised learning is to develop models that can accurately predict outcomes based on inputs. In terms of this thesis, the model will predict

[7]Fine-tuning is the process of optimizing a model for a specific task, by further training the model on a new dataset.

y as a quantitative value, based on a set of predictors x_1, x_2, ..., x_p. In its most generalized form, when assuming there exists a relationship between y and x_p, it can be expressed in the following way:

$$y = f(x_p) + \epsilon, \tag{2.1}$$

where y is the target, x_p is predictors, f is the function that the algorithm aims to learn, and ϵ is the error[8] [27].

When it comes to ML, the main concern is the accuracy and reliability of predictions, rather than the underlying nature of the relationship between input and the target. The prediction of y is based on the predictors and can be described by the following expression:

$$\hat{y} = \hat{f}(x), \tag{2.2}$$

where $\hat{y}$ is the prediction of y, and $\hat{f}$ is the estimated function of f that minimizes the prediction error the most.

2.3.1 Data Partitioning

In machine learning, data partitioning is a fundamental practice of dividing a dataset into subsets for more accurate evaluation of a model's training and performance. It's split into three subsets, a *training set*, a *validation set*, and a *testing set* to make sure the model generalizes well to new, unseen data [28].

The training set is used to train the model and is usually between 65-85% of the total dataset. Both size, quality, and variety of the data are crucial for the model's ability to make accurate decisions and learn. The validation set is usually between 10-25% of the total dataset and is used to provide an evaluation of the model's performance while training and helping to tune the parameters. The test set is used to give an unbiased evaluation of the final model's performance. This set is kept isolated from the training and valuation set to assess the performance of the finished model. Usually, around 10-25% of the total dataset is used for testing. In this thesis, the dataset was split to 85% for training and validation, and the remaining 15% of the dataset was used for testing.

[8]Deviation between the predicted value and target value. The error is independent of x_p and has statistically mean zero [27].

2.3.2 Resampling method

Resampling methods are used to ensure that models perform well on new and unseen data, especially when the dataset used is of a smaller size. There are several approaches, but common to all of them is that they re-use the training data by continuously drawing out samples from it to replicate new validation sets [27]. There are several resampling methods, among the most commonly used is *k-Fold Cross-Validation* (k-fold CV).

K-fold CV includes dividing the training set into k folds, or parts. These folds are more or less the same size, where the first fold is used as the validation set or hold-out set, and the remaining folds are used to fit the model. The prediction error, Mean Squared Error (MSE) is then calculated on the hold-out set, and then the process is repeated k times, one for each fold [27]. *K-fold CV* is what is used in this thesis and can be illustrated by the following relation.

$$CV_{(k)} = \frac{1}{k} \sum_{i=1}^{k} MSE_i \tag{2.3}$$

2.3.3 Data Preprocessing

Data preprocessing is a critical part of preparing the data for an ML training process. As this thesis aims to predict housing prices as numerical values, all features the model is trained on, also need to be numerical. In Chapter 3 the dataset and features used by the LLM and AVM will be explained in detail, but some information is derived in this section.

Solgt.no has a dataset consisting of information gathered from housing advertisements, including features like date sold, address, size, housing type, ownership type, and year built.

Ordinal Encoding

Ordinal Encoding (OE) is a simple technique where each category in a feature is assigned an integer value and is especially useful when there is a natural order between each category [29]. For example, features consisting of dates can be converted to integers, that can represent days since a given reference date. That simply converts the information from the format YYYY-MM-DD to an integer.

One-Hot Encoding

One-Hot Encoding (OHE) is a way of giving categorical features numerical values through a numeric array without any inherent order between categories, making it preferable for nominal data [29]. An example can be the feature housing type where the category column is replaced by columns for each category, and binary values are assigned to the corresponding category. Table 2.2 illustrates OHE where the housing type is replaced by colors.

Table 2.2: Illustration of One-hot Encoding.

Color	ColorRed	ColorBlue	ColorGreen
Red	1	0	0
Blue	0	1	0
Green	0	0	1

2.3.4 Feature Engineering

Feature engineering is an important step of further preparing the dataset for training in ML [30]. It involves creating new features by transforming existing features that improve the predictive performance of the model [30]. The goal is to make the relationship between the data points easier for the model to learn the underlying patterns, thereby improving the model's accuracy and efficiency. An example is to use the feature build year and calculate the age of a housing.

2.3.5 Overfitting

Within the field of ML *overfitting* is a common problem where the trained model follows the error or noise in the training data too closely [27]. This will result in the model not generalizing well to new unseen data, because of the model's complexity. An indication of overfitting is when the model has low error on the training data, but high error on the test data. Both data prepossessing and feature engineering are critical parts within ML to make the model find the underlying patterns of the data, instead of the noise [31].

2.4 Machine Learning Model - XGBoost

There are several popular algorithms used for ML tasks, including Random Forest and Extreme Gradient Boosting, among others. Extreme Gradient Boosting or XGBoost is used for supervised learning problems, which is what is used for housing price estimation in this thesis. As mentioned earlier in this chapter, supervised learning uses a dataset with several features x_p to predict a target value $\hat{y}$. XGBoost was introduced in 2016 by Chen and Guestrin [32], but is based on the work of Friedman from 1999 regarding Gradient Tree Boosting [33].

Gradient Boosting is an ML technique that's based on *decision trees* and the concept of *boosting*. Decision trees work by splitting up the input data into smaller parts based on features. This breaks down complicated decisions by dividing them into a series of several simpler choices [26]. Gradient Boosting utilizes decision trees as "weak" learners by combining them in a sequence. Each tree in the sequence aims to correct the error from the previous tree, thereby "boosting" the model by progressively reducing each tree's weakness, at the end resulting in a stronger model [34].

The XGBoost model has several additional advantages compared to Gradient Boosting, such as speed due to its ability for parallel processing, in addition to performance, and flexibility making it beneficial for complex datasets and problems [32]. The following objective function is what XGBoost minimizes during training [32].

$$\text{Obj} = \sum_{i=1}^{n} L(y_i, \hat{y}_i) + \sum_{k} \Omega(f_k) \tag{2.4}$$

$$\text{where } \Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2 \tag{2.5}$$

In Equation (2.4) and (2.5) $L(y_i, \hat{y}_i)$ is the loss function that calculates the deviation between the predicted and target value, and $\Omega(f_k)$ is the regularization term used on each tree to reduce overfitting. T is the number of terminal nodes or leaves in each tree f_k, while the parameters γ and λ control the complexity of the tree.

2.4.1 Automatic Hyperparameter Optimization

Many ML algorithms perform well after training, but the performance can benefit significantly with tuned hyperparameters. These are parameters that are set before training and used to control how the model trains in terms of behavior and efficiency.

The parameters can be set and tested manually, but using algorithms to tune them is beneficial for both performance and efficiency [26].

Based on the number of hyperparameters the most common algorithms are *grid search* and *randomized search*, where *grid search* usually are used when there are around three hyperparameters, and *randomized search* are used when there are more hyperparameters [26]. The grid algorithm works by systematically testing every possible combination and thereby finding the best possible combination of parameter values. The drawback is that the computational cost grows exponentially with the number of hyperparameters as $O(n^m)$, where m is the number of hyperparameters, and n is the number of possible values for each hyperparameter [26].

The random algorithm is as mentioned useful when having a larger number of hyperparameters as it randomly selects combinations of them. This makes the algorithm test a wide range of possibilities without evaluating every possible combination, and nevertheless finding a close to optimal combination of hyperparameters. Randomized search therefore offers a well-balanced trade-off between computational cost and model accuracy [35].

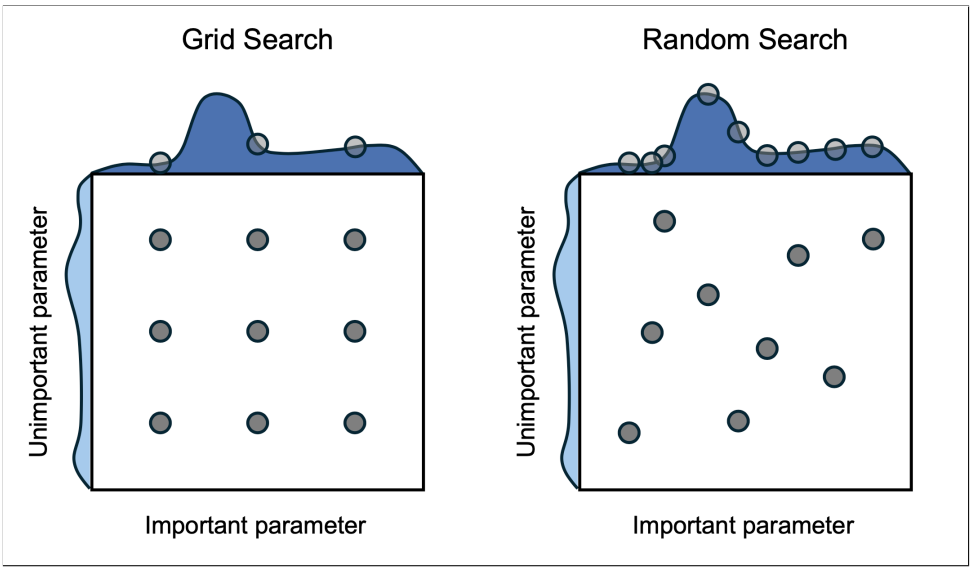


Figure 2.1: Grid Search Versus Random Search. The figure is inspired by work done by Bergstra and Bengio [35].

Chapter 3

Data

As mentioned in the introduction in Chapter 1, the overall goal is to validate if the AVM is more accurate with additional features extracted by the LLM. In order to lay the best foundation for the AVM, the quality of the LLM-features must therefore be as high as possible. The feature extraction process done by the LLM will be described in later sections. In this chapter, the datasets used in this thesis are described.

3.1 About The Data

The datasets were provided by *solgt.no* and are in two parts, one part used by the LLM, and one part used for developing the AVM. The data in the datasets are both based on residential housing advertisements from *finn.no*[1], sold between January 2022 to March 2023 in Oslo, Norway. In total, the datasets overlap with 9,842 individual transactions each but are based on the same housing advertisements identified by a *finncode*[2].

The data used by the LLM consisted of all the textual information from the housing advertisements distributed on the features finncode, title, description, and image captions. Together the three latter will be referred to as the housing advertisements *listing text* in this thesis. The data in the *listing text* consists of clean and full sentences from the advertisements, without any unnecessary whitespace.

The dataset used by the AVM consists of a total of 25 features including finncode, address, date sold, and sale price. Furthermore, the dataset has quantitative data

[1] *finn.no* is a popular platform for listing real estate for sale in Norway.
[2] A *finncode* is a unique sequence of numbers used to identify each advertisement on *finn.no*.

such as size, floors, number of rooms and bedrooms, bathrooms, and build year. In addition, the ownership type, housing type, and geographical features like district, longitude, and latitude are in the dataset.

Common to both the listing texts and dataset is that they are based on identical housing transactions. In Figure 3.1, a heat map over Oslo visualizes the frequency of property transactions in the city. Warmer colors represent a higher number of transactions in a particular area. This distribution is to get an idea of where in the city the transactions have taken place, with a majority in the city center and towards the east.

Meanwhile Figure 3.2 represents the distribution of housing prices expressed in MNOK[3] from the dataset, showcasing how the prices are distributed in the dataset. This shows that the majority of the transactions were in the range of 3.5 to 5 MNOK in the period from January 2022 to March 2023.

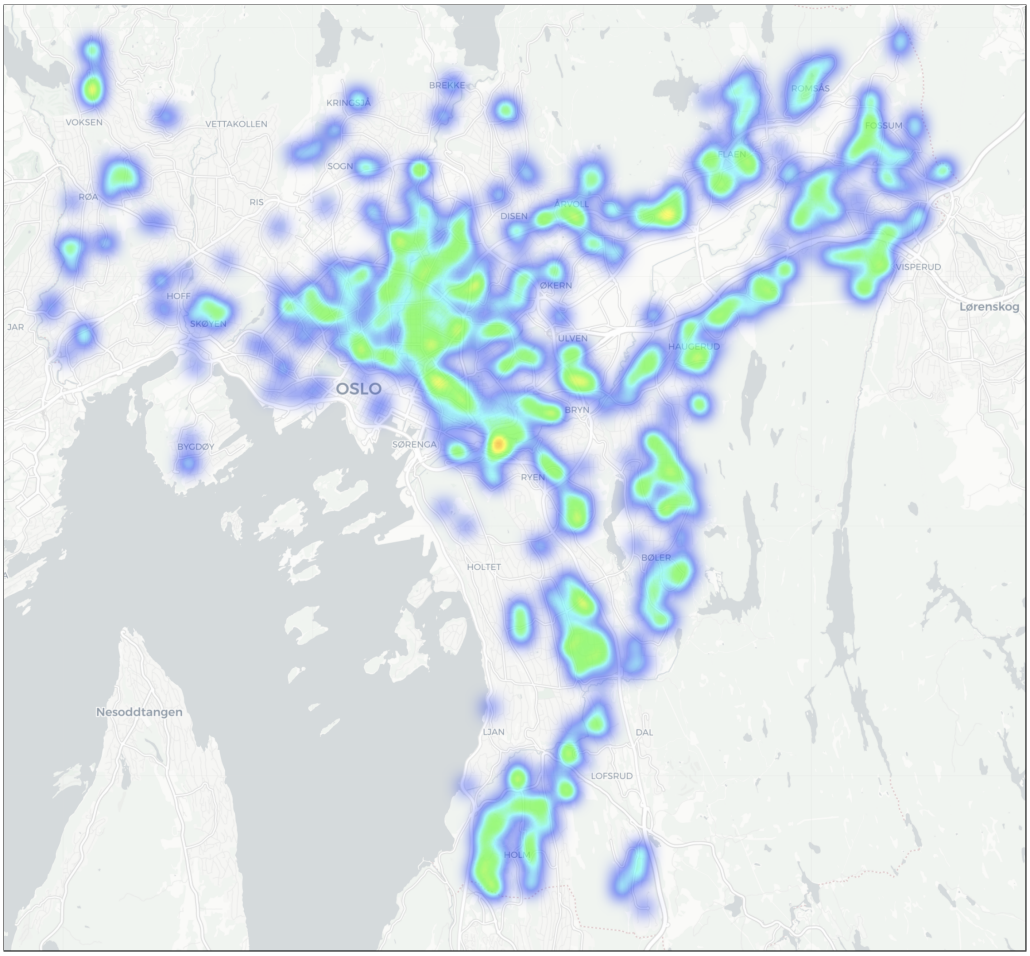


Figure 3.1: Heat-map over the residential property transactions in Oslo, used in the dataset. A warmer color indicates a higher number of transactions in the area.

[3]Million NOK

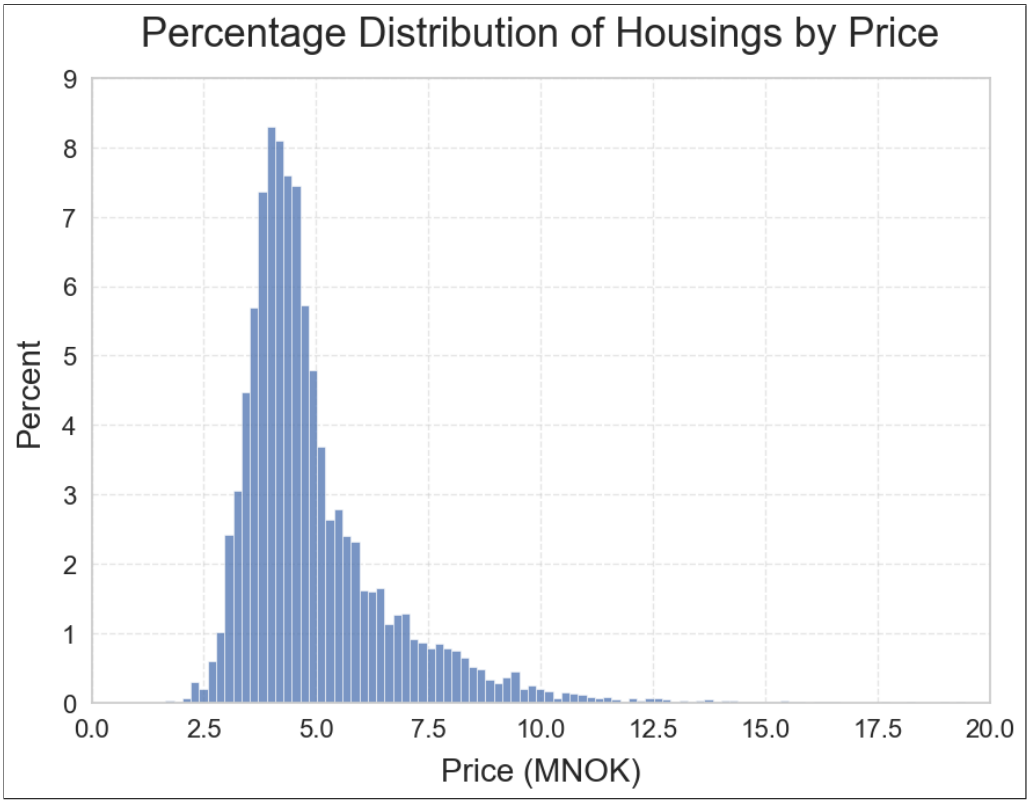


Figure 3.2: The distribution of housings by price in the dataset.

Table 3.1 shows a summary of the dataset and its important statistics for each of the 24 usable features. This includes information about the housing such as size, rooms, bathrooms, and housing type. In addition, information about the area and location of the housing such as the address, postal code, longitude, and latitude are displayed. Additional information about the features is explained in the footnotes under the table.

The data in the dataset has several missing values that need to be handled to have a fully usable dataset. Furthermore, some values need to be pre-processed in order to be processed by an ML algorithm as it's raw and unfiltered. The dataset will also be expanded with additional features the LLM extracts from the advertisements. All of this is going to be explained and derived in later chapters.

Table 3.1: Summary of the statistical information of all features in the dataset. The setup of this table is inspired by "House price prediction with gradient boosted trees under different loss functions" by Hjort et al. [36].

Feature	Unit	Min	Mean	Max	Type
Price[1]	MNOK	1.65	4.95	19.35	Numerical
Common Debt[2]	MNOK	0	0.30	5.94	Numerical
Common Costs[3]	NOK	530	4,557	21,490	Numerical
Housing Type[4]	–	–	–	–	Categorical
Ownership Type[5]	–	–	–	–	Categorical
Building Type Code[6]	–	–	–	–	Categorical
Energy Label[7]	–	–	–	–	Categorical
Build year	–	1860	1958	2023	Numerical
BRA[8]	m^2	2	65	365	Numerical
PROM[9]	m^2	13	63	365	Numerical
Lot Size[10]	m^2	0	105	4142	Numerical
Floor[11]	–	1	2.98	19	Numerical
Rooms[12]	–	0	2.71	7	Numerical
Bedrooms[13]	–	0	1.74	6	Numerical
Bathrooms[14]	–	0	0.76	3	Numerical
WC[15]	–	0	0.80	4	Numerical
Elevator	–	–	–	–	Binary
Balcony	–	–	–	–	Binary
Parking[16]	–	–	–	–	Binary
Longitude	Degrees	–	–	–	Numerical
Latitude	Degrees	–	–	–	Numerical
Postal Code[17]	–	–	–	–	Numerical
District[18]	–	–	–	–	Numerical
Date Sold	–	–	–	–	Date
Address	–	–	–	–	String

[1] Price including depth given in million NOK.
[2] The common loan of the housing association.
[3] Monthly common costs.
[4] Housing types: Detached home, duplex, row house, apartment, and other.
[5] Ownership types: Housing cooperative, sole ownership, and other.
[6] A standard for categorizing building types according to their primary use or function. This includes the land, size, and addresses [37].
[7] Energy labels are a scale from A to G for housing's expected energy consumption.
[8] The size of the total area in m^2.
[9] The size of the living area in m^2.
[10] The size of the lot in m^2.
[11] The floor the housing is located on. For detached homes, the floor is set to 1. If a housing has multiple floors, the lowest floor is used.
[12] Total number of rooms in the housing.
[13] Number of bedrooms in the housing.
[14] Number of full bathrooms in the housing.
[15] Number of additional bathrooms in the housing.
[16] If there are parking facilities adjacent to the home.
[17] A standard four-digit code that indicates a geographical area in Norway.
[18] District or Basic Statistical Unit is a numerical id for *"grunnkrets"*, used for geographical areas for statistical and property purposes in Norway.

3.2 Software

The development of the software used in this thesis is done using the programming language *Python*. Two programs were made, one for the feature extraction process and one for the AVM development. The first program used the *openai* package which includes a framework for setting up and accessing the GPT models through OpenAI´s API, more on this in the next chapter [38]. For developing the AVM, the package *sklearn* was used which provides a toolbox for development and visualizations of results [39]. Furthermore was the package *xgboost* used to access the XGBoost model used in this thesis [40].

Both programs including development, training, and testing were done on a MacBook Pro with a six-core CPU and 16 GB of RAM. For the size of the dataset used in this thesis, this worked just fine.

Chapter 4

Methodology Phase One: Feature Extraction with LLM

The work done in this thesis will be divided into two phases, split between two chapters. Phase One, covered in this chapter, describes the methodology behind implementing and utilizing AI and the GPT models to extract information from housing advertisements listing texts. The first section will cover what information the model aims to extract, and later sections cover how the models are accessed and an analysis of the result. Phase Two, covered in Chapter 6 covers the implementation of the AVM and is partially based on the results from this chapter.

The analysis in Phase One aims to investigate how well GPT models can understand and extract information from the listing texts in housing advertisements, compared to the actual information. This is done by developing a program that takes the full listing text as input, sends it to the model, and systematically saves all extracted numerical values in an output file. These values will correspond to a set of new features that will be used in testing after Phase Two. The analysis includes comparing the data with manually labeled data and tuning the model through prompt engineering to increase quality and accuracy. Figure 4.1 displays a simple visualization of the program.

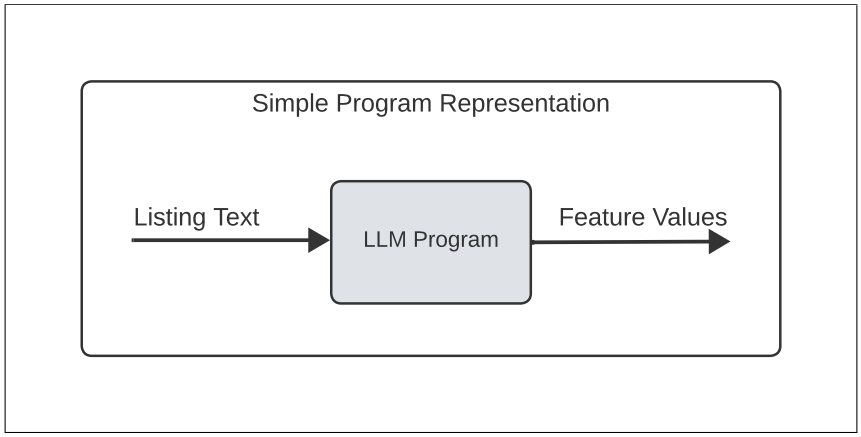


Figure 4.1: Simple Program Representation of LLM.

4.1 Feature Extraction Using LLM

Housing advertisement websites such as the well-established *finn.no* but also the newer *hjem.no*[1] have both structured information and unstructured information in the listing texts. The listing text in these advertisements, including image captions and descriptions, has important information about the housing in it. The idea is to make a program that utilizes AI's NLP characteristics to turn the textual information into quantitative values, by defining a set of rules for the model to follow for each feature. In this way, the program will take the full listing text as input, and the model will return a score for each feature based on the rules for each of them. The structured part of the advertisement is already collected by *solgt.no* and is part of the dataset explained in Chapter 3.

4.1.1 Features To Extract

The chosen features that the LLM extracts are based on recommendations and wishes from *solgt.no* and what is believed to affect the prediction of the price the most, both positively and negatively. This is supposed to give a more accurate housing price prediction in the end. All features that the LLM extracts and an explanation of them are listed in Table 4.1. The rules that the LLM follows will be described in Chapter 4.2.

[1]In the start of 2024 *hjem.no* was released as a competitor to *finn.no*, but is based on the same set up in terms of information in the housing advertisements.

Table 4.1: Features for the LLM to extract. All features have numerical values.

Feature	Explanation
Parking	Best parking option in the area.
Kitchen	Year of the last upgrade.
Bathroom	Year of the last upgrade.
Fireplace	Number of fireplaces.
Storage	Number of storage units.
Storage Size	Total size of storage units in m^2.
Outdoor Area	Type of outdoor area.
Outdoor Area Size	Total size of outdoor area in m^2.
Housing Standard	Overall impression of housing.

4.2 LLM Program

This section provides a detailed description of the program for feature extraction. Figure 4.2 visually illustrates the program, while the following sections will explore and describe every aspect of the program. The program takes one housing advertisement from the dataset of 10,000 at a time and saves the results in a new file as new features. For the sake of simplicity, the program is explained in terms of one advertisement.

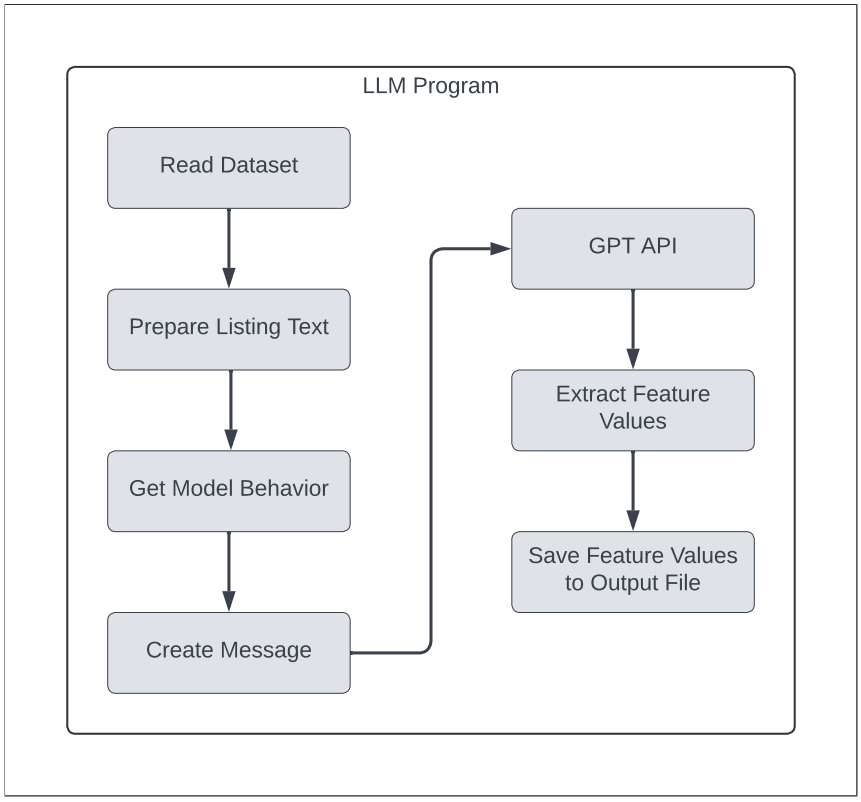


Figure 4.2: LLM Program

4.2.1 Data Preparation

The first part of the program reads the dataset with the housing advertisements. As described earlier the dataset consists of the title, image captions, and description that are combined into what is referred to as the advertisement *listing text*. As the data consists of clean full sentences it isn't necessary with any other processing of the data. It is from this text the LLM should be able to return values based on each feature's distinct rules.

4.2.2 Creating Message

After assembling the listing text, the data is ready for the GPT model. The next part of the program consists of accessing the GPT model, as well as generating a response from it. The newer models use what is called *Chat Completions API* endpoint, which requires a special way of defining the model's input. To generate a response from the LLM, the input must be constructed in a way that represents the model's behavior. This is an instruction for how the model should behave to the prompt or input. In this case, the *prompt* is the listing text, and the *model_behavior* is a text defining how the listing text should be analyzed. This combination of *model_behavior* and *prompt* are called *message* and are defined in the following way.

```
message=[
            {"role": "system", "content": f"{model_behavior}"},
            {"role": "user", "content": f"{prompt}"},
        ]
```

The *model_behavior* is a string constructed as a set of rules for the model to follow for each feature. The full set of rules is formulated in Appendix A. This describes in detail what the model can expect as input, in this case, a listing text. Furthermore, the string includes how the model should return the extracted information based on the rules for each feature. In total, the program accesses the GPT model once per program cycle, in other words, one message per listing text. The process of *prompt engineering* to get desired output values in the right format is described in Chapter 5. Figure 4.3 illustrates the process of making a message.

The *model_behavior* a continuation from the preliminary project, and the formulation in this thesis are therefore similar to the prompts from the project [1].

OpenAI offers several models in different price ranges, limitations, and performance,

but the model used in this thesis is OpenAIs GPT-3.5 model *gpt-3.5-turbo-0125* with a limitation of 16,000 input tokens. Both *model_behavior* and *prompt* are considered as input, so the program also handles cases where the total of input tokens exceeds the limit. The output from the LLM is a string with all features and corresponding scores in the following format, `'parking':'p_score'`, `'kitchen':'k_score'`, `'bathroom':'b_score'` and so on for all features, where `'x_score'` are the values the LLM returns as most probable based on *model_behavior*.

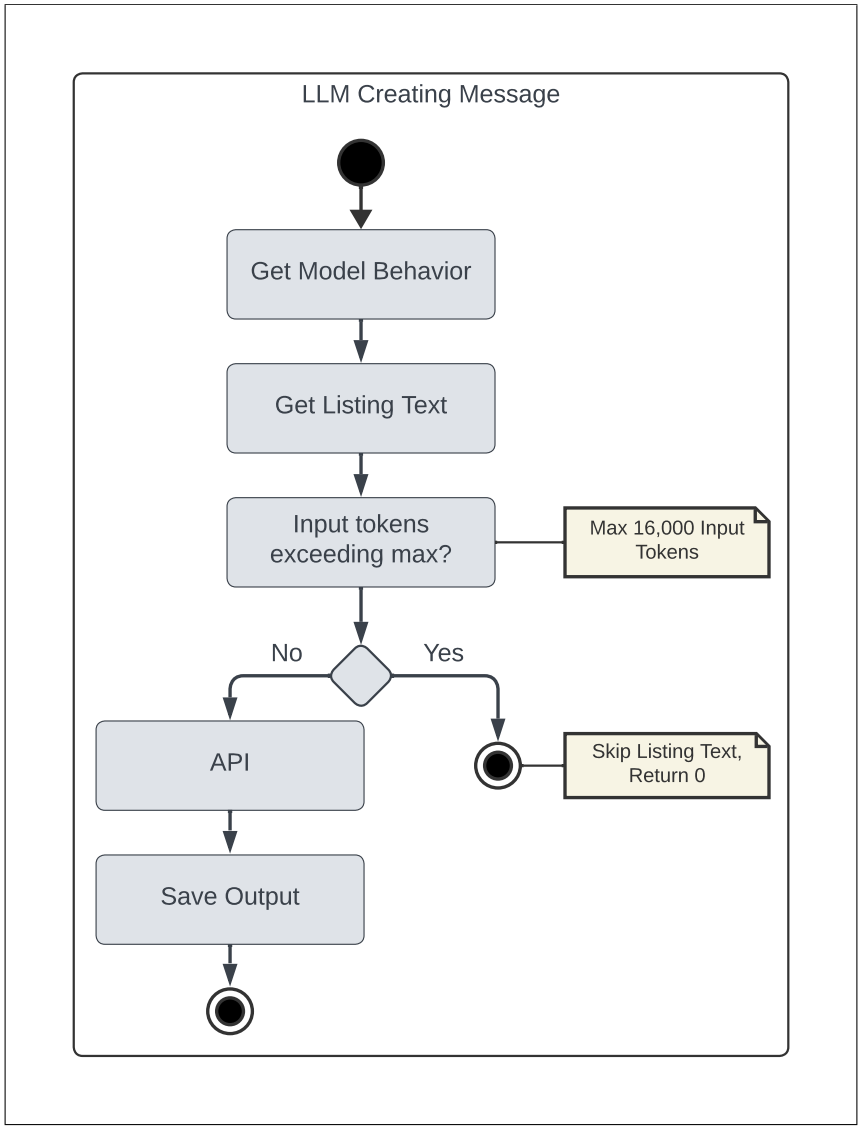


Figure 4.3: Visualization of the program's method for preparing and creating a message for the LLM. This also shows how the program handles when a message exceeds the input token limit.

Finn.no is a Norwegian online marketplace for listing a wide range of products and services, including real estate. The majority of these housing advertisements are

written in Norwegian *Bokmål*[2], and the *model_behavior* is therefore also written in the same language, as the advertisements are from Oslo, Norway. This is to match the *model_behavior* and *prompt*, as it is easier to check if the model has missed some information when checking the results from the output.

4.2.3 Value Extraction

The output from the LLM is a string with all features and corresponding scores as described in the last section. The next part of the program analyses the output string from the LLM by cycling through the features and extracting the corresponding scores. These output scores are saved systematically in a .CSV[3] file and will result in the new LLM-features for the AVM, as explained in Table 4.1.

[2] *Bokmål* and *Nynorsk* the two written languages in Norway. The latter is only used in the western part of the country.

[3] CSV is a file format that is used to store large amounts of data in a tabular form. In terms of this thesis, each feature represents one column.

Chapter 5

Results Phase One: Feature Extraction with LLM

In this chapter, the results from the GPT model will be analyzed. Each result for each feature extracted by the model will be described in detail, but firstly some observations during the development of the program are derived.

5.1 Testing During Development

During the development of the program, extensive *prompt engineering* was done to make the *model_behavior* as precise as possible. As mentioned in Chapter 2, this is a process of formulating the input in a way that results in the desired output. Earlier drafts of the *model_behavior* were simple and vague and had several deficiencies that resulted in varying output format and values. As mentioned in Chapter 2, the preliminary project had the same issue due to an older model and different endpoint, and showed inconsistency in the results [1]. For this thesis, the earlier draft made these values difficult to interpret and use in a good way as the output format varied greatly. The latest version of the *model_behavior* was based on observations from earlier drafts and was more direct and concise, resulting in a more consistent output format and values. An excerpt from the early and final drafts goes as follows,

Early draft: *"Teksten er et utdrag fra en boligannonse. Vi ønsker å score for kjøkken basert på teksten. Returner året kjøkkenet ble bygget/pusset opp. Returner et .json objekt der score er året."*

Final draft: *"Din oppgave er å analysere innholdet i boligannonsen. Du skal return-*

ere følgende informasjon om boligen på følgende format: (...) 'kitchen': 'k_score', (...). 'k_score' er det fulle årstallet kjøkkenet er fra eller sist det ble renovert, pusset opp eller oppgradert, eventuelt boligens byggeår. Hvis årstall ikke er nevnt, returner 0."

An observation from earlier drafts is that the model often returned a full sentence instead of the year, and the year was often confused with other years from the advertisement text. The final draft makes the model more consistent and follows the defined format. This has been a crucial part of the thesis and the observations and formulations have consisted of trial and error, as well as recommendations from OpenAI for formulating inputs [9], as mentioned in Chapter 2.

5.2 Testing Process For The LLM

An important part of the thesis was to validate the effectiveness and reliability of the LLM. This is done by conducting several tests with the LLM and comparing them to manually labeled data. The manually labeled data consists of information from 200 housing advertisements based on the same rules described by model behavior used by the LLM.

In total, 15 tests were conducted and analyzed all with the same identical model behavior and the same 200 housing advertisements. The test was grouped into three groups with five tests per group, as illustrated in Table 5.1. This was done in order to be able to compare the results directly both to the manually labeled data and to each group. Several methods were used to validate the results based on the format of each feature such as precision-recall methods, correlation, and percent-wise accuracy compared to the manually labeled data. The 200 advertisements resulted in 200 API requests, one per program cycle.

Table 5.1: Illustration of the testing process. The total of 15 tests was divided into three groups of five tests each.

Test Groups	**Tests**
Group 1	Test 1 - 5
Group 2	Test 6 - 10
Group 3	Test 11 - 15

5.3 Feature-Wise Results

Figure 5.1 displays the average feature-wise accuracy of the model when compared to the manually labeled data. The figure displays the percentage of matching values compared to manual labeling, but due to how the data is analyzed it isn't necessarily a good way to present results for all features. Especially for feature `parking` and `housing_standard` the results are seemingly worse than the other features, but in the sections below the results will be derived further.

5.3.1 Percent-Wise Accuracy

The data in Figure 5.1 display the results from Test 1 to Test 5 in Group 1. The overall outcome showed a strong performance for the majority of the features extracted. The features `kitchen`, `bathroom`, `fireplace`, `storage_size` and `outdoor_area_size`, had all over 90 % match with the manually labeled data. This indicates that the model effectively handles the defined task from the model behavior and prompts for these features in the tests well. For `kitchen` and `bathroom` the model extracts a year for the last significant upgrade and returned a strong performance of 93.0 % and 90.8 % compared to the labeled data, respectively.

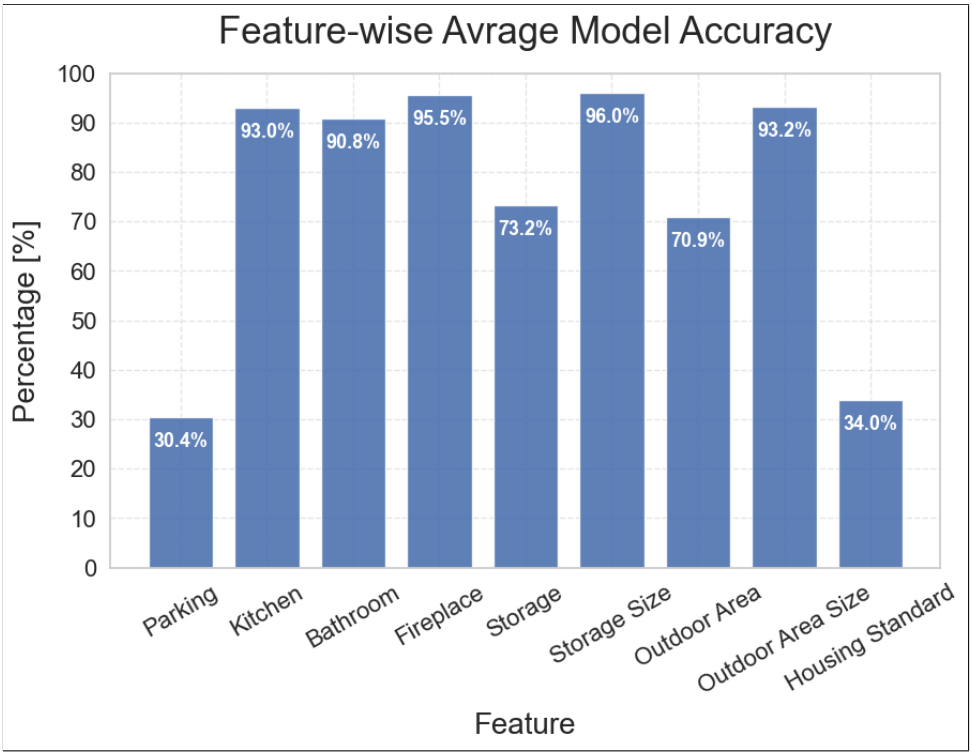


Figure 5.1: The average feature-wise model accuracy compared to manually labeled data for Group 1, Test 1 - 5.

Figure 5.1 only displays the results from Group 1, but when all tests across the groups are compared to each other, the output returned for all features is relatively consistent. The biggest difference between the results was 5.0 percentage points for `housing_standard` where the weakest result was 31.5 % in Test 10 and the strongest was 36.5 % in Test 11. The average difference between the weakest and strongest results for all features was 2.2 percentage points. The average feature-wise model accuracy for all tests is summarized in Table 5.2, as well as each feature´s correlation compared to the manually labeled data. This is an indication of how well the LLM-data follows the labeled data.

Table 5.2: Average feature accuracy and correlation compared to manually labeled data.

Features	**Accuracy**	**Correlation**
Parking	29.37 %	0.7920
Kitchen	92.87 %	0.8730
Bathroom	90.70 %	0.8400
Fireplace	95.47 %	0.8455
Storage	73.37 %	0.6080
Storage Size	95.70 %	0.8282
Outdoor Area	70.53 %	0.7032
Outdoor Area Size	93.57 %	0.9801
Housing Standard	33.67 %	0.7146

5.3.2 Precision-Recall

The feature `fireplace` had an average accuracy of 95.47 %, but an observation from the labeled data showed all samples either had a fireplace or not[1]. The feature is therefore treated as a binary feature when analyzing the results. In total 77.00 % of the samples did not have a fireplace, and in such cases where one observation is significantly more frequent than the other, the precision-recall method is a way of validating the results from the model [41].

When extracting information, recall measures the amount of relevant results returned, while precision measures the relevancy of the results. Precision-recall displays a trade-off where high recall and low precision return many results with mostly incorrect predictions. If the precision is high and recall is low the system will return few results, but with mostly correct predictions. Therefore, a strong system has high

[1]Of the nearly 10,000 samples used later in the thesis, 0.60 % has two or more fireplaces.

precision and high recall [26]. Precision (P) and recall (R) are a number between 0 and 1 and are given by the following equations

$$P = \frac{T_p}{T_p + F_p}, \tag{5.1}$$

$$R = \frac{T_p}{T_p + F_n}, \tag{5.2}$$

where T_p are true positives, F_p are false positives, and F_n are false negatives. To summarize the performance with a single number one can calculate the harmonic mean of the precision and recall. This is referred to as the F_{score} and is given by the following equation [26],

$$F_{score} = \frac{2PR}{P + R}. \tag{5.3}$$

During the testing process, 14 of the 15 tests conducted produced the exact same predictions for `fireplace` showing both consistency and strong results where $F_{score} = 0.9072$, with $P = 0.8627$ and $R = 0.9565$. Test 12 deviated from the other test with a slightly lower precision of 0.8462 resulting in a F_{score} of 0.8980.

Most housings either have a fireplace or not, and very few have multiple fireplaces which was the case in this dataset. As described in Table 3.1, the dataset consists of mostly smaller apartments where a fireplace isn't common. If it had been assumed that no housing had a fireplace, the test would have returned a 77 % accuracy, meaning that the GPT model performs well, as the F_{score} indicates.

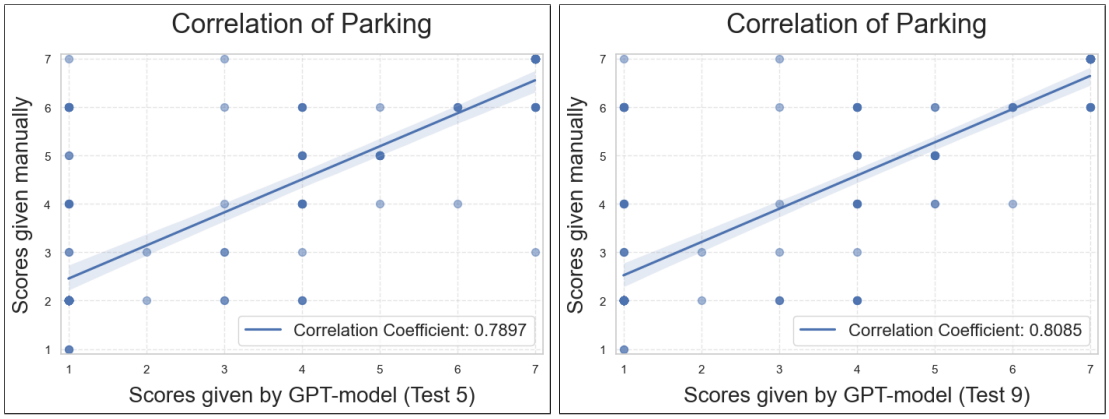
5.3.3 Correlation

As briefly mentioned earlier the accuracy and correlation of each feature were listed in Table 5.2. The table reveals that the features `parking` and `housing_standard` had significantly lower accuracy than the other features. In contrast, these features had more complex rules described in the *model behavior*. By calculating the correlation coefficient the trends in the data are captured, even though the complicated rules and criteria might be hard to hit exactly. The coefficients for both features showed a relatively high correlation, and therefore a stronger performance than the accuracy first indicated.

Figure 5.2 displays the correlation coefficient for `parking` between the weakest **(a)** and strongest **(b)** test, respectively. Compared to the manually labeled data the

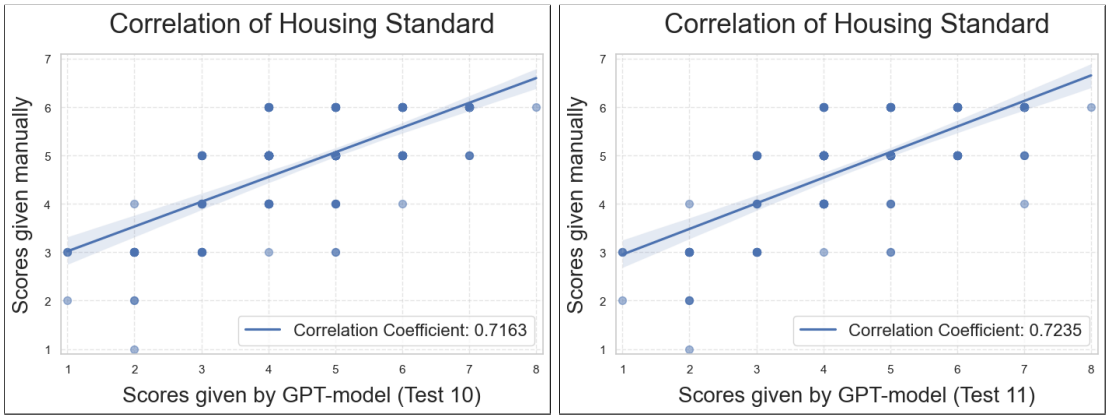
weakest and strongest test resulted in a correlation coefficient of 0.7897 and 0.8085. This indicates a stronger relationship and result than the percent-wise accuracy indicated.

When it comes to feature `housing_standard`, the correlation is plotted in Figure 5.3, and shows the weakest **(a)** and strongest **(b)** test for the feature. The correlation coefficients for the plots were 0.7163 and 0.7235 which indicate a stronger relationship and performance than the percent-wise accuracy first indicated.



(a) **(b)**

Figure 5.2: Correlation of Parking compared to manually labeled data.



(a) **(b)**

Figure 5.3: Correlation of Housing Standard compared to manually labeled data.

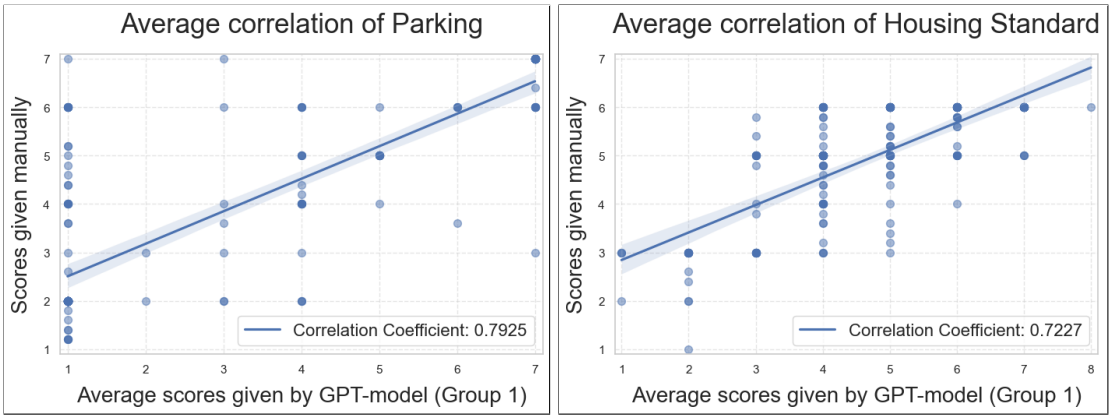
As language models have some uncertainty and unpredictability in the model's response the correlations and results will vary. The average correlation coefficients for `parking` and `housing_standard` for each group are summarized in Table 5.3 bellow. Observations form the tests suggests that a weaker performance for one feature also resulted in a weaker result for the other feature, the same goes for the stronger

performance. An example is Group 2 where both `parking` and `housing_standard` had a higher correlation coefficient than the other groups.

Table 5.3: Average of the correlations between LLM and manually labeled data. (Average of the correlations.)

Groups	Parking	Standard
Group 1	0.7744	0.7041
Group 2	0.8057	0.7251
Group 3	0.7958	0.7146
Average	0.7920	0.7146

In a separate analysis of the output data, the average of the output scores was calculated, before calculating the correlation between this average and the manually labeled data. This resulted in correlation coefficients of 0.7925 and 0.7227 for `parking_average` and `housing_standard_average` in Group 1 respectively. The results are illustrated in Figure 5.4 and summarized in Table 5.4. When comparing these results to the results above, the latter analysis results in a slightly higher correlation coefficient for both features and scores higher in all groups for both features. This indicates that averaging the LLM-outputs might reduce noise and unusual deviations from the labeled dataset.



(a) **(b)**

Figure 5.4: Correlation of the average parking and Housing Standard compared to manually labeled data.

Table 5.4: Correlation of the average scores given by the LLM compared to manually labeled data. (Correlation of the average.)

Groups	Parking	Standard
Group 1	0.7925	0.7227
Group 2	0.8147	0.7451
Group 3	0.8096	0.7355
Average	0.8056	0.7344

Since the last analysis indicated a better performance when averaging the LLM-output, the same was done for all features for all tests. After calculating the average of all 15 tests and then calculating the correlation, the coefficients for all features showed an increase. As mentioned this indicates a reduction in noise and deviations from the manually labeled data. Table 5.5 compares both approaches derived in this section.

Table 5.5: Comparison between average correlation, and correlation of the average.

Features	Average Correlation	Correlation of the Average
Parking	0.7920	0.8186
Kitchen	0.8730	0.8740
Bathroom	0.8400	0.8450
Fireplace	0.8455	0.8461
Storage	0.6080	0.6171
Storage Size	0.8282	0.8547
Outdoor Area	0.7032	0.7463
Outdoor Area Size	0.9801	0.9812
Housing Standard	0.7146	0.7474

5.3.4 Runtime and Cost

As mentioned the tests above are done with 200 housing advertisements, which had an average runtime of 4,33 minutes and cost of $0.21 per test. The total number of listing texts given by *solgt.no* is 9,984 all of which were sent through the program and analyzed by the LLM. This resulted in 9,984 API requests totaling 23.68 million tokens, with 23.0 million input tokens, and 0.68 million output tokens. The total cost of these was $24.33 with a computational time of 4.3 hours.

5.4 LLM Discussion and Observations

The LLM shows strong overall performance with high accuracy and high correlation for all features when compared to the manually labeled data. This is a good indication of how well the GPT model understands the goal, and its ability to extract the correct value from the listing test.

Even with the strong performance, there are some important observations from the listing texts and the results. The listing text provided by *solgt.no* did not include all text from the housing advertisements. Both the sections *key information* and *facilities*[2] were left out of the provided advertisements, leaving out some additional information that could improve the results of the LLM further.

Another observation from the housing advertisement revealed generally varying information about the facilities, sizes of outdoor spaces, and recent upgrades. Where some had detailed information about everything, and others offered nothing. Table 5.6 illustrates the variation in the data.

The table shows the number of times the information about a given feature isn't available, or the model could extract it. The manually labeled data in the first row of the table emphasizes that 47.00 % and 26.00 % of the listing text didn't have information on when the last upgrade of the kitchen and bathroom was. This correlates well with the GPT model's ability to extract the same information, as the last two rows in the table suggest. When it comes to the storage area and outdoor area, the manually labeled data suggests that in 79.50 % and 38.00 % of the housing advertisements, the area isn't mentioned. Furthermore, if a measurable storage area or outdoor area is mentioned in the listing texts, 40.00 % and 26.50 % of these sizes aren't, mentioned. Again the GPT model's ability to extract information suggests the same.

Table 5.6: Lack of information from the housing advertisement.

	Kitchen	**Bathroom**	**Storage Size**	**Outdoor Size**
Labeled (200 Samples)	47.00 %	26.00 %	79.50 %	38.00 %
Test Sets (200 Samples)	51.50 %	33.00 %	76.50 %	37.00 %
AVM Set (10 K Samples)	47.38 %	32.71 %	70.05 %	38.49 %

Another observation from the listing text was that if the year wasn't explicitly mentioned in context, the model had difficulty returning the correct year for the

[2]Facilities often summarizes the information from the housing advertisement.

kitchen and bathroom. In addition, if a housing had several upgrades over multiple years the model didn't always choose the most recent for kitchen and bathroom. Despite this, the model shows strong performance in finding the correct information.

The model's ability to extract the number of storage units was also relatively high, resulting in a correlation of 0.6080 compared to the manually labeled data. The same goes for the outdoor area, with a correlation of 0.7032 the model shows a strong performance.

When it comes to the formulation of the listing text, it generally under-communicates the standard of a housing condition if the condition is considered worse. If the housing condition is considered better, the opposite goes and the listing text focuses on the details. It's the same case for parking, where worse parking options are often under-communicated. This makes it hard for the LLM to distinguish the differences in the lower end.

Chapter 6

Methodology Phase Two: Automated Valuation Model

The work done in Phase Two covers the implementation and training of the AVM and is partially based on observation and results from Phase One. This included making a program that takes a selection of features from the dataset and the LLM as input, training the model based on the features, and calculating the model's performance. The first section includes preparing the data through data pre-processing and feature engineering. Later sections include partitioning the data and training the model.

The analysis in Phase Two aims to investigate how the features extracted by the LLM influence the AVM and validate the effectiveness and reliability of the AVM. A simple illustration of the program from features to trained model is explained in Figure 6.1.

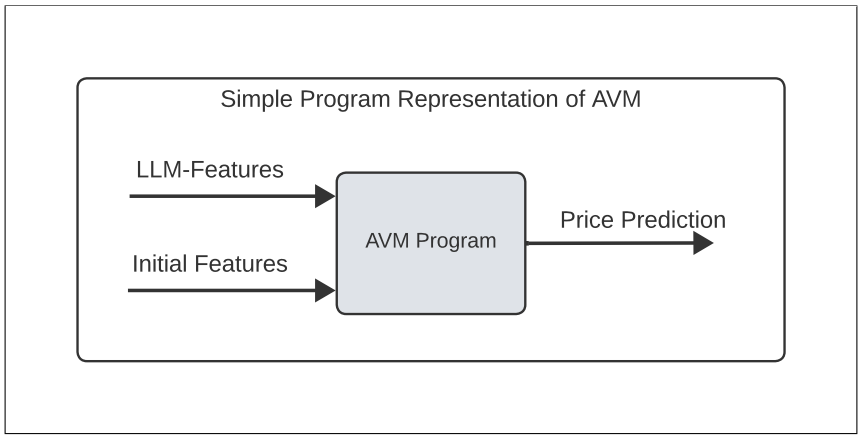


Figure 6.1: Simple Program Representation of AVM.

6.1 Available Features To The AVM

The features available have been presented earlier and are a combination of the original dataset, and what the LLM extracted from the housing advertisements. The Table 4.1 and Table 3.1 from earlier chapters explain all the features available to the AVM, and the combined available features are listed in Table 6.1. As the feature `address` is unique to all housings and is given as a string, the feature isn't suitable for the model as it only takes numerical values. Instead a combination of `lng` and `lat` will give a representation of the location of the housing, in addition to `district_id` which represents a small geographical area. The total of available features is then 33, including 24 features from the original dataset and nine extracted by the LLM.

Table 6.1: All available features for the AVM.

Features from dataset		**Features from LLM**
balcony	housing_type	parking_llm[1]
bathrooms	lat	kitchen
bedrooms	lng	bathroom
BRA	lot_size	fireplace
build_year	ownership_type	storage
building_type_code	parking[2]	storage_size
common_costs	postal_code	outdoor_area
common_debt	price	outdoor_area_size
district_id	PROM	housing_standard
elevator	rooms	
energy_label	sold_date	
floor	WC	

6.2 Program for Model Training and Tuning

This section provides a detailed explanation of the program for training and tuning the AVM. The following sections will describe different aspects and parts of the program, from preparing the data, training the model, and evaluating the model's performance. For a visual explanation of the program see Figure 6.2. The program

[1]Parking from the LLM is given as a numeric value. Earlier in the thesis referred to as `parking`.
[2]Parking in the dataset is given as boolean.

is based on a subset of the features from the 9,842 housing advertisements and produces an estimation of how well the model works on new and unseen data.

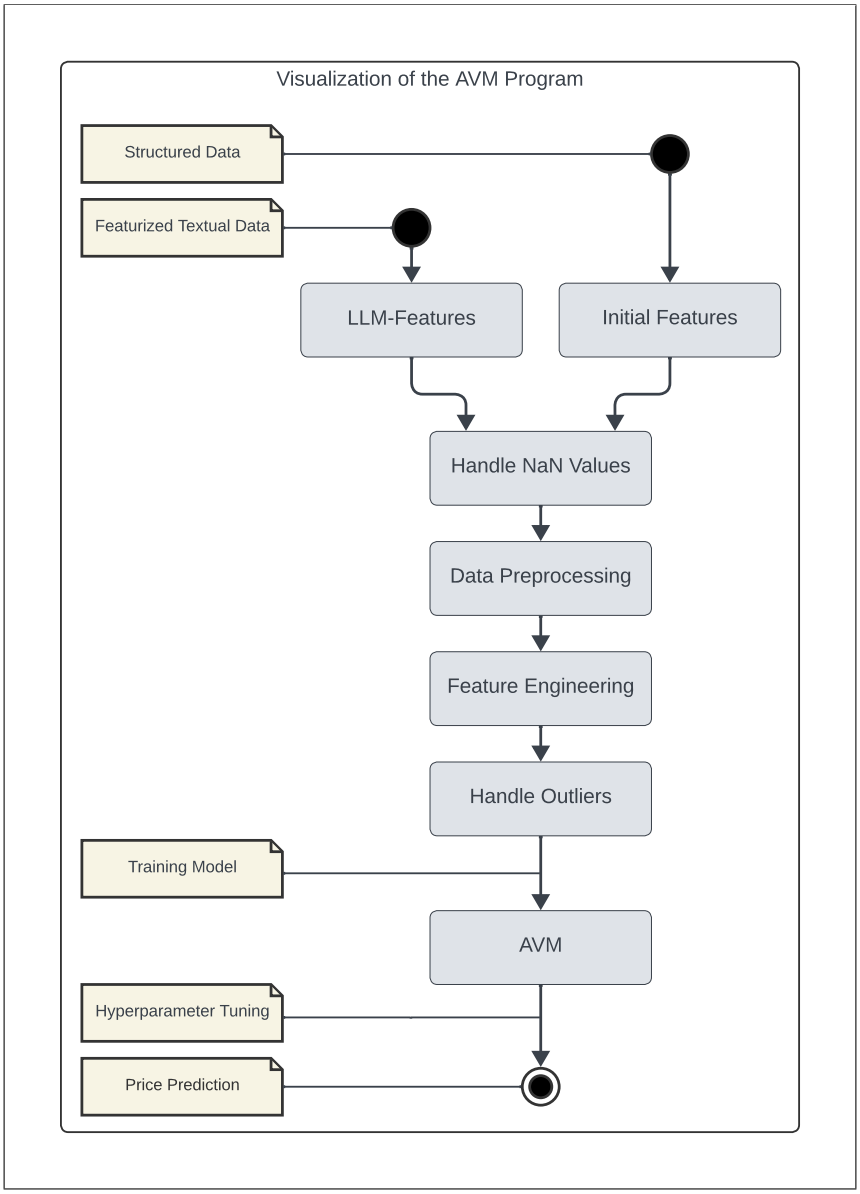


Figure 6.2: Program representation of the AVM program.

6.2.1 Handling NaN Values

The first part of the program reads the datasets and assembles the data frame based on the features chosen for the model. Since the dataset has several missing values, these are needed to be handled in order to train the model. There are several ways of handling these, but the simplest way is to delete the entire row when a missing value is registered. As our dataset is of a smaller size, doing so would reduce the dataset by 12.35 % to 8,623 samples and would affect the model performance negat-

ively. Because of the differences in the features and their domain, the missing values are handled differently based on the features the missing values belong to. The results in Chapter 5 indicated that the LLM had difficulty returning values for some of the LLM-features. As LLMs do have a certain randomness, it's not surprising that the model missed some values when extracting them from the listing text. The "missing" values of `kitchen` and `bathroom` from the LLM-features were handled to compensate for the LLM output. Totally nine of the available features had missing values and are summed up in Table 6.2, where they are split between mode[3], mean[4], and median[5] as these had best results. The two LLM-features `kitchen` and `bathroom` have the by far highest amount of "missing" values due to the lack of information in the listing texts as discussed in Chapter 5.

Table 6.2: Handling NaN/missing values

Feature	Number of NaN	NaN Replacement
bedrooms	213	mode
district_id	45	mode
building_type_code	353	mode
common_costs	28	mean
BRA	119	mean
build_year	12	mean
lot_size	861	mean
kitchen	4730	median
bathroom	3266	median

6.2.2 Data Preprocessing

After preparing the dataset and handling missing values, some features undergo *data preprocessing* which transforms data into a usable format. In this case the features `sold_date`, `housing_type` and `ownership_type` undergo preprocessing to convert date/time and categorical values to usable formats.

The feature `sold_date` is converted to `sold_date_days`, transforming the data from Year-Month-Date format to numerical values representing days since the housing is sold, from a reference day. This makes the new feature a direct indication of when

[3]Mode is the value that appears most frequently.
[4]Mean is the average value.
[5]Median is the middle value in a sorted list.

a property is sold, describing changes in the real estate market.

Both `housing_type` and `ownership_type` are features with categorical values, that undergo OHE to prepare them for the model training process. As mentioned in Chapter 2 and illustrated in Figure 2.2, one-hot encoding creates a binary column for each category in a feature. The original feature `housing_type` is replaced with five new columns, and `ownership_type` is replaced with three new columns, enabling the model to handle the categorical values. These columns are given in Table 6.3.

Table 6.3: One-Hot Encoding of the features `housing_type` and `ownership_type`.

Feature	
`housing_type`	`ownership_type`
`_detached`	`_co-op`
`_duplex`	`_sole`
`_row`	`_other`
`_apartment`	
`_other`	

6.2.3 Feature Engineering

Feature engineering are used to prepare the data further and increase the data quality in the dataset. As derived in Chapter 2, *feature engineering* involves changing the relationship between the data points for a feature to make it easier for the model to learn the patterns on the data. From the available features in Table 6.1, `kitchen`, `bathroom`, and `build_year` are all features where the values are a year. These values have been converted to age, directly providing information on how old the kitchen, bathroom, or building is by transforming the data and reducing the model's complexity by eliminating the need to handle this relationship. After calculating the age, by simply subtracting the feature year from the current year, the original features are replaced by `kitchen_age`, `bathroom_age`, and `building_age`.

6.2.4 Remove Outliers

Even though handling missing values, *data preprocessing*, and *feature engineering* ensure that the data are in the right format, there are still data points for each feature that deviate significantly from other data in the dataset. These differences,

referred to as *outliers*, are not meant to be in the dataset and can negatively impact model performances.

In this thesis, *Isolation Forest* (IF) has been used to identify and remove *outliers* in the dataset. This method has been proven to be effective and preferred over other approaches, especially for datasets with many abnormal deviations [42]. By conducting a test it was found that the *Isolation Forest* model found 1,041 possible *outliers*, meaning 10.58 % of the dataset. Because of the sample size of the total dataset, removing too many data points would remove real values and thereby the diversity in the data.

Further testing found that removing 9.5 % of the possible *outliers* both gave the best results while keeping the majority of the dataset. This resulted in filtering out 1 % corresponding to 99 samples from the original dataset reducing it from 9,842 to 9,743.

6.2.5 Data Partitioning and Resampling Method

In this thesis, the training and test split of the data is 85 % for training and the remaining 15 % of the data for testing. The testing set is what the model uses to compute the error and performance of the model, and was chosen as it provides a good balance between having sufficient data for testing and training.

As mentioned earlier, the resampling method for validation used during training is *K-fold CV*. The number of folds is set to $k = 5$, which provides a strong balance between computational efficiency and the number of samples per fold. This choice offers a well-balanced trade-off between the amount of training data and validation data, as the number of folds reduces the variance of the performance estimate. A higher number of folds could increase the robustness of the estimate, it also would increase the computational cost. To prevent overfitting the models are trained with *early stopping*, which is a mechanism that stops the model training before it learns the patterns in the data too well. At this point, the base model with the chosen features is done and tested towards the testing set.

6.2.6 Hyperparameter Tuning

To enhance the model's performance, the XGBoost model has several hyperparameters that need to be tuned. The chosen parameters are listed in Table 6.4, and are based on observations from existing literature and the documentation for the model

[43] [44] [45]. To prepare for the tuning process each hyperparameter is defined with a set of values, and then different combinations of these values constitute the optimization process. These values range based on the literature mentioned and the dataset size. In addition to enhancing the model's performance, will this also reduce the overfitting of the model [45]. As mentioned in Chapter 2, there are beneficial to use the *random search* algorithm for the number of hyperparameters chosen as this will find the good values for hyperparameters much faster than *grid search* [35]. This means that even with fewer hyperparameter combinations tried, the *random search* fits the model comparable to what *grid search* would. In this case, 100 different combinations of hyperparameters are tried out. Table 6.4 and Table 6.5 explain and display the different hyperparameter ranges.

Table 6.4: The range and explanation of the hyperparameters are from the XGBoost documentation [45].

Hyperparameter	**Range**	**Explanation**
learning_rate	[0, 1]	Learning rate
gamma	[0, ∞]	Minimum loss reduction required for partition
min_child_weight	[0, ∞]	Minimum sum of instance weight in child
subsample	[0, 1]	Ratio of training set sampled for each tree
colsample_bytree	[0, 1]	Ratio of columns when constructing a tree
max_depth	[0, ∞]	Maximum depth of the tree
n_estimators	[0, ∞]	Number of trees
alpha	[0, ∞]	L1 regularization term on weights
lambda	[0, ∞]	L2 regularization term on weights

Table 6.5: The hyperparameter tuning range is based on the literature mentioned in this section.

Hyperparameter	**Value Range**
learning_rate	(0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9)
gamma	(0.0, 0.1, 0.2, 0.3, 0.4, 0.5)
min_child_weight	(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
subsample	(0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0)
colsample_bytree	(0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0)
max_depth	(3, 4, 5, 6, 7, 8, 10)
n_estimators	(100, 200, 300, 400, 500, 600, ..., 1800, 1900, 2000)
alpha	(0, 0.01, 0.1, 1, 10, 100)
lambda	(0, 0.01, 0.1, 1, 10, 100)

6.3 Model Evaluation

The model used in this thesis, XGBoost, aims to reduce the Mean Squared Error (MSE) between the prediction and actual value but default. As this thesis seeks to investigate what impact the features extracted by the LLM have on the AVM, it is therefore natural to use MSE or Root Mean Squared Error (RMSE), which is directly linked to MSE, to measure the AVMs performance. In addition, further comparisons between the output of the AVMs are also Mean Average Error (MAE) and Mean Average Percentage Error (MAPE) used. RMSE is given by the following formula

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \tag{6.1}$$

where n is number of observations, y_i is true value, and $\hat{y}_i$ is predicted value [46]. To get clearer and easier comparisons the percentage deviation between the RMSE and the average target value are used in this thesis. This is referred to as RMSE% in this thesis and is given by the following

$$RMSE\% = \frac{RMSE}{average\_target} \times 100. \tag{6.2}$$

When it comes to Mean Average Error (MAE) and Mean Average Percentage Error (MAPE) they are given by the following

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \tag{6.3}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100, \tag{6.4}$$

where n is number of observations, y_i is true value, and $\hat{y}_i$ is predicted value [47]. By comparing the different error measures the impact and distribution of the errors can be explained. MAE is a measure of the average size of errors, without considering if the error is positive or negative. The same goes for MAPE, but given as a percentage error from the true value, giving a relative measure of the size of the error. RMSE measures the square root of the average squared error. Equation (6.1) describes how

large errors or outliers[6] are given a higher weight, making it penalize larger errors to a greater extent than MAE [46].

[6]Significant errors are often referred to outliers.

Chapter 7

Results Phase Two: Automated Valuation Model

In this chapter, the testing and results from the AVM will be presented and analyzed. The work and observations done in Phase One and Phase Two of this thesis are combined to produce comparable results. Firstly the testing process and assumptions for the process are explained, this includes some observations of the model's performance. Secondly, models are trained on different combinations of the initial features and compared to the inclusion of LLM features. Lastly, the performance observations are discussed and put in context wit observations from Phase One.

7.1 AVM Testing Process

The main part of this thesis is to validate the effectiveness and reliability of the AVM, when the features extracted by the LLM are used. To compare the model performances several tests are conducted with different combinations of the features available, both without the LLM-features and with them.

The models trained are based on the same 9,743 samples in the dataset, and the same data preprocessing and feature engineering as the program described in Chapter 6. In addition, the data partitioning and resampling methods are also identical in all tests. The features used in training are the only aspect that is different during the processes, and both models are tuned through hyperparameter tuning. The range of the hyperparameter values can be found in Table 6.5 in Chapter 6.

The testing process started with training the AVM on all available features and then

checking each feature's importance. The LLM-features importance and impact on the AVM output were investigated by removing and analyzing one LLM-feature at a time, and then calculate the model's new performance. Later the models are mainly trained on a subset of the features available, and there are several *feature selection* techniques to filter out the weak features. As the main goal of this thesis is to investigate to what degree the LLM-features impact the AVM, the *feature selection* process hasn't been described in detail, as this is outside the scope of this thesis.

Observations show a slight difference in the performance between the *training set* and *test set* for all models trained. Although hyperparameter tuning corrects parts of the deviation, the model fits the training set lightly better than the test, indicating a mild sign of *overfitting*. Even with this small indication the model fits the *test set* relatively well, suggesting a respectable performance on unseen data and viable results.

7.2 AVM Comparisons

Each model is trained on all or a subset of the initial features from the dataset from Table 6.1, and is compared to different combinations of the LLM features. This will show the impact of the LLM-features in a clear and systematic way.

7.2.1 Test 1 - All LLM-features

For the first model, all 24 of the initial features available were used and were hyperparameter-tuned to enhance the model performance. The trained and tuned model, referred to as the *base model*, resulted in an RMSE% of 10.53 % and a MAPE of 6.36 % compared to the *test set*. When adding all nine LLM-features the model was trained on a total of 35 features and the model performance resulted in an RMSE% of 9.40 % and a MAPE of 5.35 %. This increased the model accuracy for both measures by 1.13 and 1.01 percentage points respectively, resulting in an improvement of 10.73 % for RMSE% and 15.88 % for MAPE. Table 7.1 summarizes the result from Test 1.

Table 7.1: AVM results compared to base model for Test 1.

		Base AVM		Difference	Improvement
Test 1	RMSE%	10.53 %	9.40 %	-1.13	10.73 %
	MAPE	6.36 %	5.35 %	-1.01	15.88 %

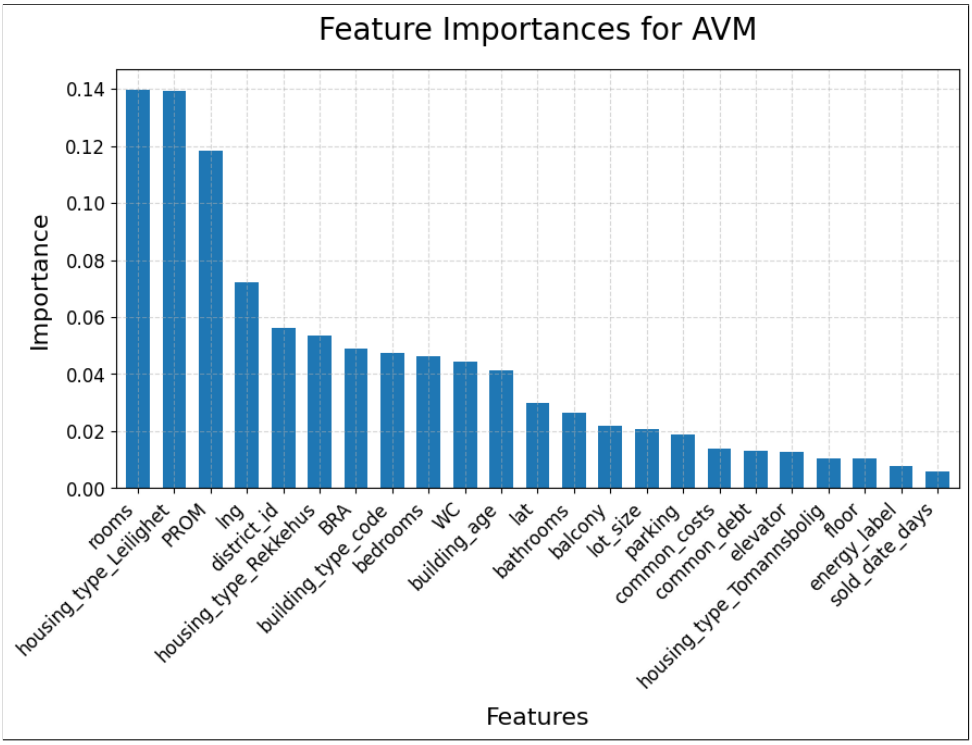


Figure 7.1: Feature Importance of AVM in Test 1.

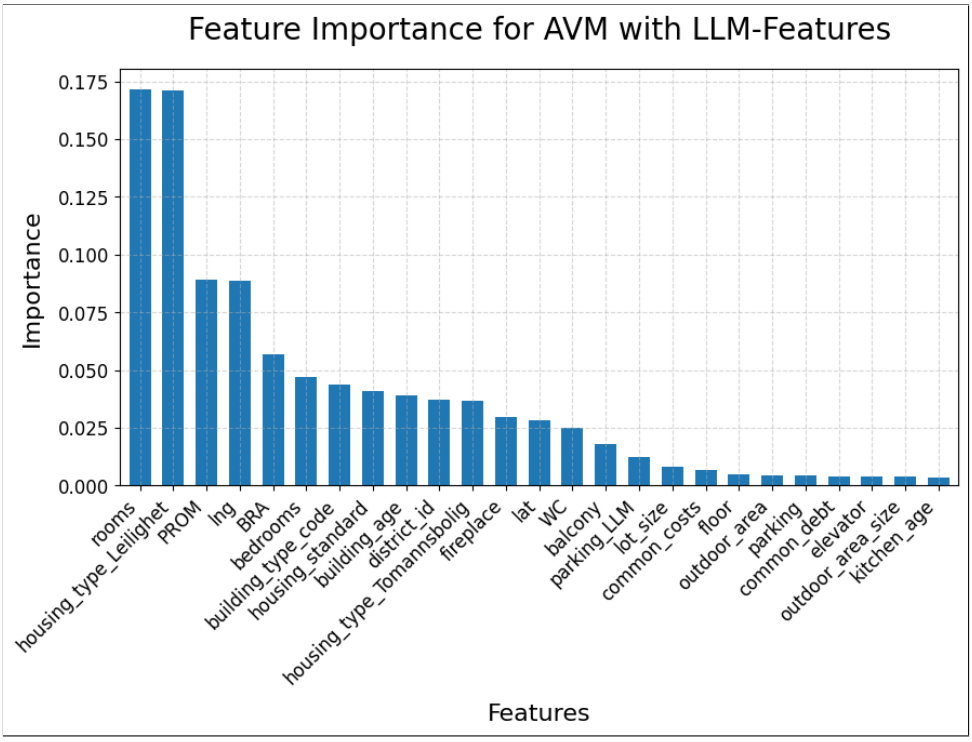


Figure 7.2: Feature Importance of AVM with LLM-features in Test 1.

Figure 7.1 and Figure 7.2 visualize the feature importance for the AVM and the AVM with LLM-features respectively. The plots show how much each feature improves the performance of the model where the features are listed on the x-axis, and the importance score is on the y-axis. The second plot only displays 25 of the most important features due to the lack of space. It is clear from both figures that the features `rooms`, `housing_type_apartment`, `PROM`, and `lng` are the features that have the greatest impact on both models. On the other end, the least impactful features are slightly different for each of the models. For the AVM `sold_date_days` have the least impact on the model, in addition to `housing_type_detached`, `housing_type_other`, and `ownership_type_co-op` which was omitted from the figure. An observation during the test was that all samples from the dataset had the same ownership type, making the feature `ownership_type` constant, which means it doesn't contribute to the performances of any of the models.

The housing types mentioned above also have the least impact on the AVM with LLM-features. In addition `storage`, `bathrooms` and `storage_size` are also weak for the latter model in this test. Furthermore, among the bottom third of the weakest feature importances for this model was a total of five LLM-features. This includes `kitchen_age`, `bathroom_age`, and `outdoor_area_size` in addition to the two LLM-features mentioned above.

Figure 7.3 and Figure 7.4 are SHAP (SHapley Additive exPlanations) summary plots for the AVMs. These plots also display the importance of features but with the degree of change and their direction for the 15 most impactful features. The y-axis shows the level of importance, and the x-axis indicates the degree of change. Each dot represents one data point for each feature from the dataset, and the color indicates the value of the data point, where red is a higher and blue is a lower value. The plot shows that `PROM`, `lng` and `BRA` are among the most impactful features, as the feature importance plots above also showed. The size of a housing is understandably an important factor when estimating its housing. When considering the SHAP plot for the AVM with LLM-features suggests that `housing_standard` and `parking_llm` have a relatively strong impact on the model's output, while other features more or less stay the same. This is in line with the model performance enhancement as described in Table 7.1. When analyzing longitude further, one can see that lower data point values mostly have positive SHAP values and higher data point values have negative SHAP values. As a smaller longitude means further to the west, the figure indicates that housing location in the east-west plane significantly impacts the model output. This is in line with the statistics of the real estate market trends in Oslo [48].

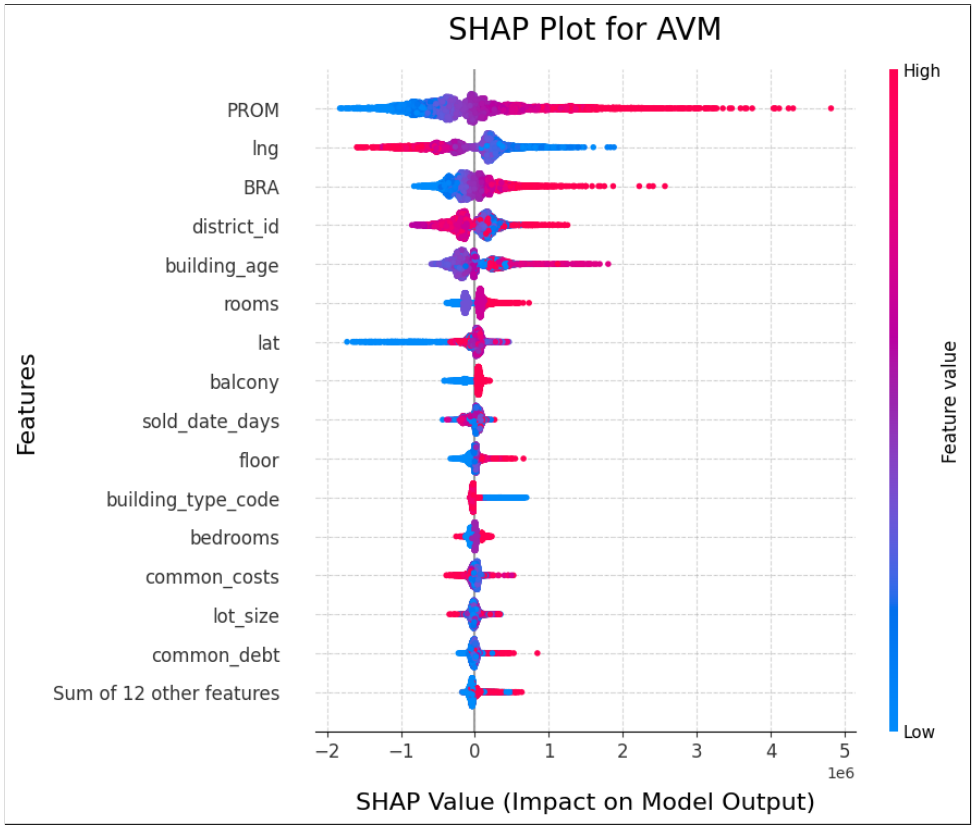


Figure 7.3: SHAP Values for AVM in Test 1.

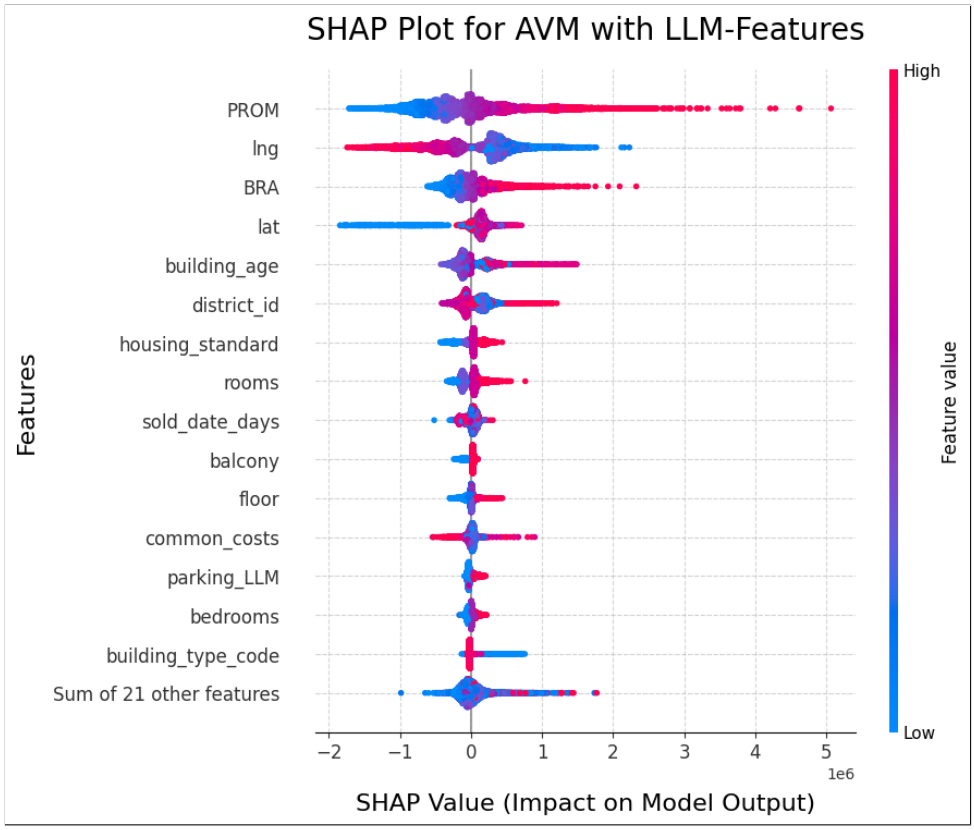
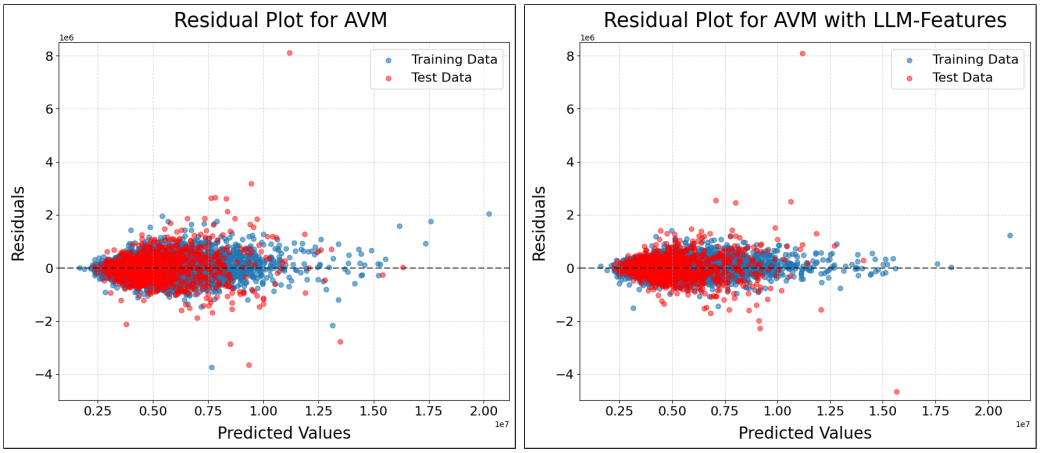
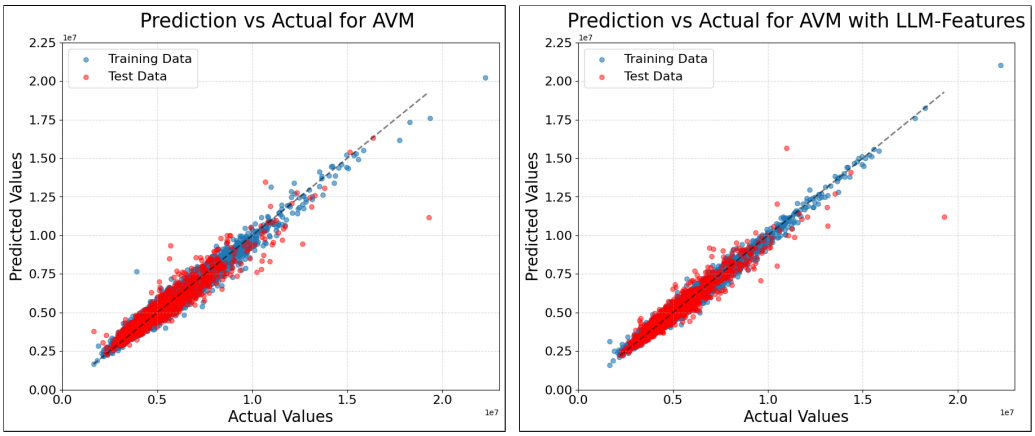


Figure 7.4: SHAP Values for AVM with LLM-features in Test 1.



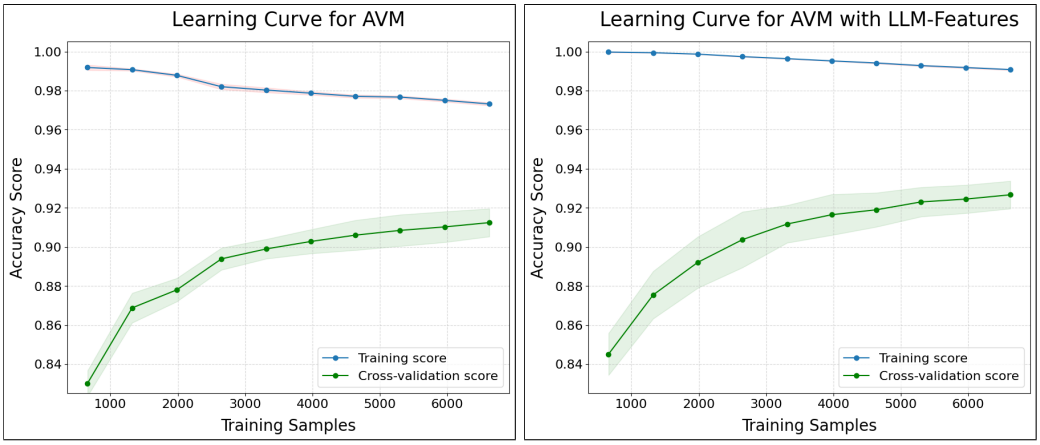
(a) **(b)**

Figure 7.5: Residual plot and its distribution.



(a) **(b)**

Figure 7.6: Prediction versus actual values.



(a) **(b)**

Figure 7.7: Learning curves

Figure 7.5 compares the residual[1] plots for the models, for both training and test set. In Figure 7.6 the prediction versus the actual values are displayed. In both figures **(a)** are more noisy and have more outliers, while **(b)** have a more compact plot. Even though the test set has several outliers, the majority of the data points are more compact when compared to **(a)** for both figures. These outliers are due to some rare cases where all LLM-featrues was 0. The overall observations are in line with the results of RMSE % and MAPE indicating a more accurate price estimation when the LLM-features are included.

The learning curves in Figure 7.7 describe the learning process where a bigger gap between the training curve and the learning curve (Cross-validation score) indicates *overfitting* of the model. A perfect model would have an accuracy score of 1, while worse performing model would have an accuracy score towards 0. As described earlier, the models fit the test data relatively well, but the gap can indicate a slight overfitting of the models.

7.2.2 Test 2 - Subset of LLM-features

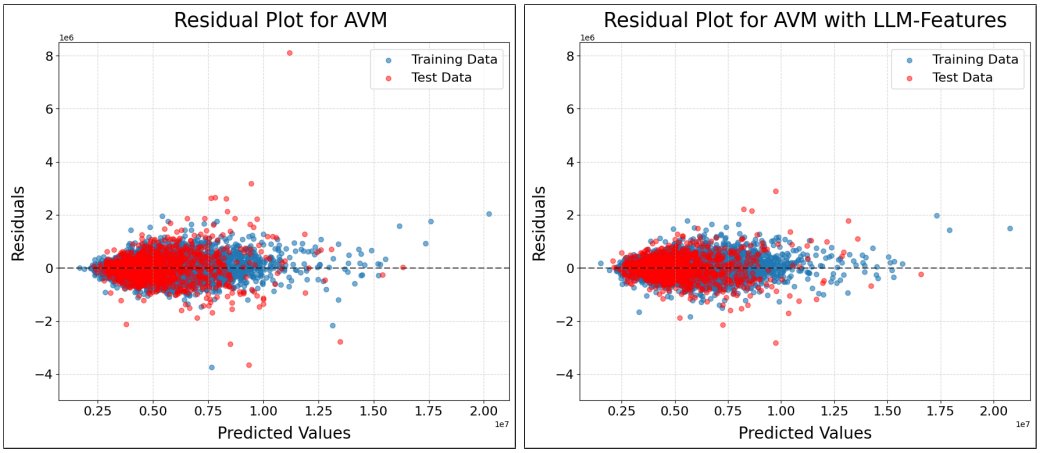
Test 1 indicated that the LLM-features had a positive impact on the model's performance in total, but the four features `bathroom_age`, `kitchen_age`, `storage_size`, and `outdoor_area_size` still showed low a feature importance compared to others. In Test 2 these weak features were removed and compared to the same base model as in Test 1.

The removal of the weaker LLM-features resulted in an RMSE% of 8.14 % and a MAPE of 5.59 % compared to the *test set*. When compared to the base model, this showed an increased accuracy by 2.39 and 0.71 percentage points, resulting in an improvement of 22.70 % for RMSE% and 12.11 % for MAPE. The results are summarized in Table 7.2 below, and also compared to results from Test 1.

Table 7.2: AVM results compared to base model for Test 1 and Test 2.

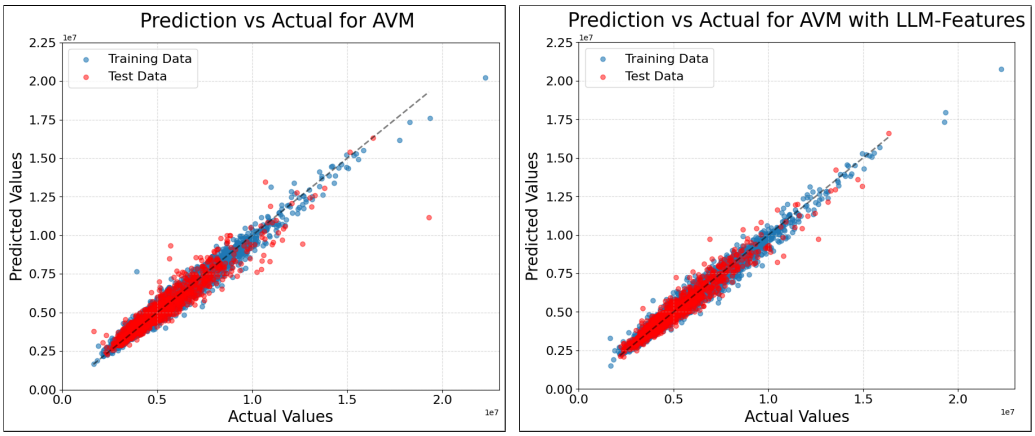
		Base AVM		Difference	Improvement
Test 1	RMSE%	10.53 %	9.40 %	-1.13	10.73 %
	MAPE	6.36 %	5.35 %	-1.01	15.88 %
Test 2	RMSE%	10.53 %	8.14 %	-2.39	22.70 %
	MAPE	6.36 %	5.59 %	-0.77	12.11 %

[1]Residual is the difference between the actual true value and estimated value.



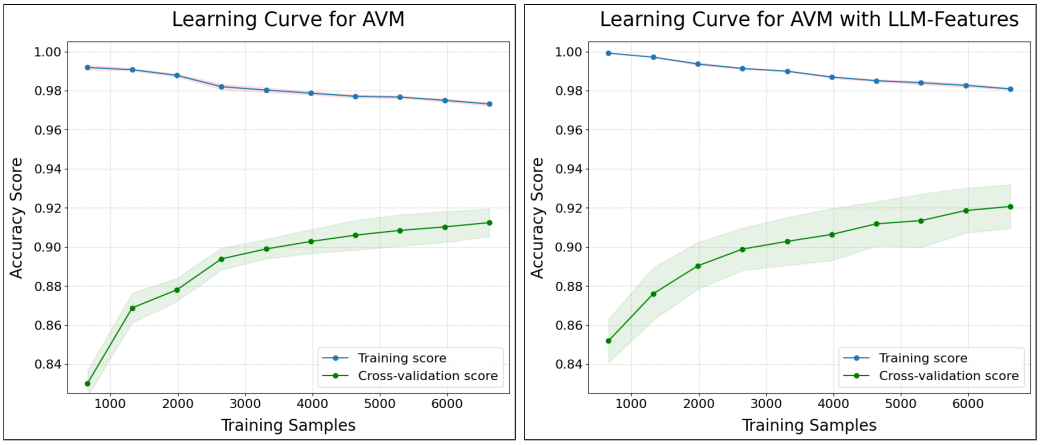
(a) (b)

Figure 7.8: Residual plot and its distribution.



(a) (b)

Figure 7.9: Prediction versus actual values.



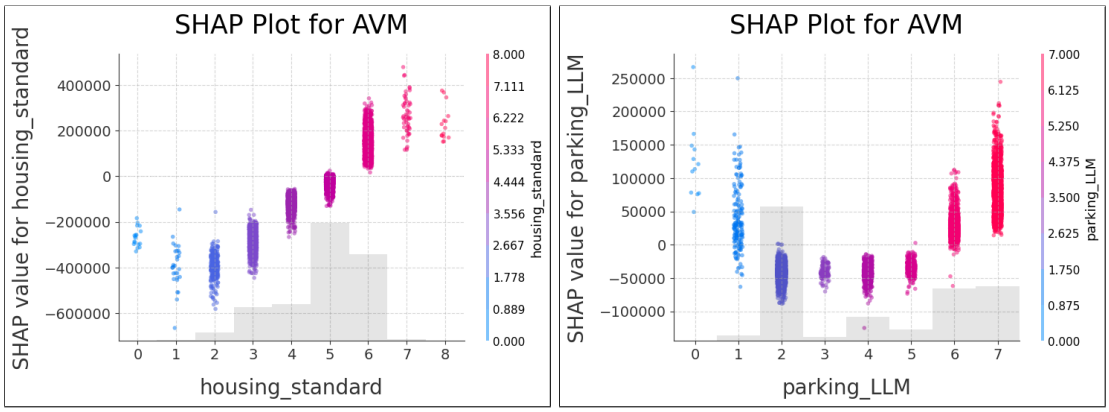
(a) (b)

Figure 7.10: Learning curves

Figure 7.8 and Figure 7.9 compare the results from the base model in Test 1, and the improved model after eliminating weaker LLM-features. Once again the **(b)** plots show an improvement over the base model, also when compared to the AVM with LLM-featurs from Test 1. When comparing both AVMs with LLM-features, the model from Test 2 has a 13.40 % improvement for RMSE%, while the MAPE is slightly worse at a -4.49 % change. Despite these results, Figure 7.10 **(b)** shows a smaller gap between the learning and training curves, indicating a better fit and total improvement of the model.

7.2.3 LLM-Feature Impact

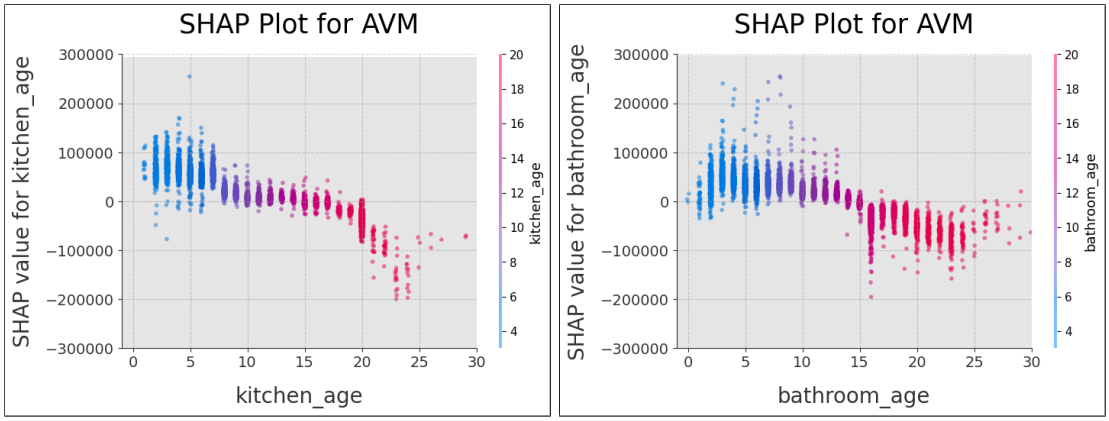
To investigate which LLM-features are most important for the base model, their performances are plotted as SHAP Dependence Plots compared to the actual housing price. Figure 7.11 **(a)** display performance of `housing_standard` which indicates a clear positive relationship compared to actual housing price. In **(b)** the `parking_llm` is displayed and shows a slightly different, but also positive, relationship. The plot starts more flat before increasing significantly with a higher value, which corresponds to a "better" parking option. This indicates that parking isn't as impactful for the lower feature values.



(a) **(b)**

Figure 7.11: SHAP Dependence Plot.

The plot in Figure 7.12 **(a)** and **(b)** show `kitchen_age` and `bathroom_age`. Both of these plots are relatively flat but have a slight negative trend. This reveals a negative correlation between each feature and the actual housing price. Both plots are relatively flat until the age of 15 years, before trending downwards. As both features start with a relatively flat trend, they are not as impactful and important to the model for housings younger than 15 years.

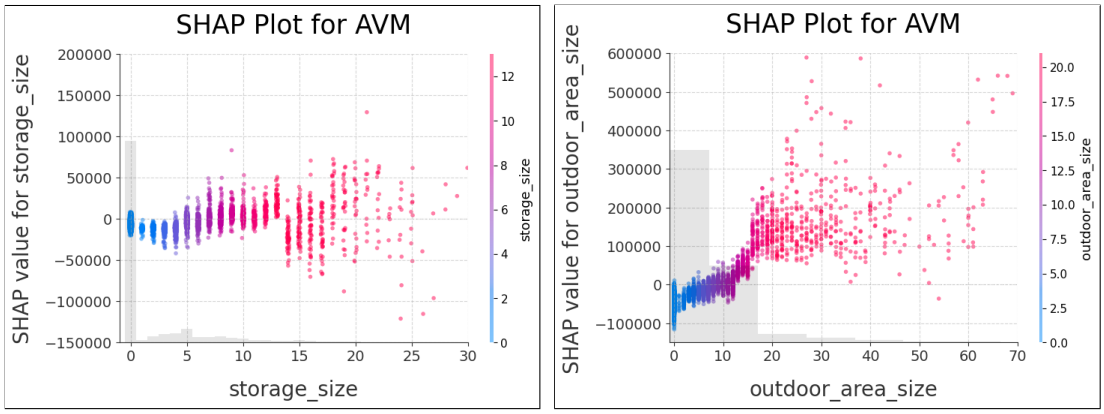


(a) **(b)**

Figure 7.12: SHAP Dependence Plot.

The plots in Figure 7.13 are the SHAP plots for `storage_size` and `outdoor_area_size`. Both have noisy results, especially for higher feature values, which impacts the model negatively as it isn't consistent with its values for the same preconditions. Plot **(a)** are generally flat indicating a weak impact on the model's output, while **(b)** has a slight positive trend, but a lot of noise. The irregularities in both plots make it hard for the features to contribute positively to the model.

Further observations for the remaining LLM-features show a relatively strong performance by the feature `fireplace` with a clear positive correlation, while `outdoor_area` and `storage` were weaker only contributing slightly. Common to the last two was a relatively flat plot with many data points on a few values.



(a) **(b)**

Figure 7.13: SHAP Dependence Plot.

7.2.4 Test 3 - Subset of Initial and LLM-features

Based on the observations it is a clear indication that some of the LLM-features have positive effects on an AVM. After testing each individual LLM-feature with the base model from Test 1, it shows that all of them have a clear positive impact on the base AVM on their own, except `storeage_size` and `outdoor_area` which didn't significantly affect the model.

A *feature selection* method has been used to find a subset of all available features, including LLM-features, for the model to enhance its performance. This process removed the housing standards `_detached` and `_other` as well as `ownership_type`. Additionally, four[2] LLM-features, `bathroom_age`, `kitchen_age`, `storage_size`, and `outdoor_area_size`, were also filtered out.

Without the LLM-features, this model returned an accuracy of 9.98 % in terms or RMSE% and an MAPE of 6.53 %. The feature combination when introducing the LLM-fetrues resulted in a strong performance with an RMSE% of 7.97 % and MAPE of 5.39 %. This is an accuracy improvement of 20.14 % for RMSE% and 17.46 % for MAPE. This is the strongest model presented and has the smallest gap between training and test curves, both increasing the accuracy and minimizing signs of overfitting. Compared to the base model in Test 1, by introducing LLM-features and undergoing feature selection the AVM showed an accuracy improvement of 24.31 % for RMSE% and 15.25 % for MAPE. The results are summarized in Table 7.3 below.

Table 7.3: AVM results compared to base model for Test 1, Test 2 and Test 3. Additionally the model in Test 3 is also compared to the subset AVM.

	Base AVM			**Difference**	**Improvement**
Test 1	RMSE%	10.53 %	9.40 %	-1.13	10.73 %
	MAPE	6.36 %	5.35 %	-1.01	15.88 %
Test 2	RMSE%	10.53 %	8.14 %	-2.39	22.70 %
	MAPE	6.36 %	5.59 %	-0.77	12.11 %
Test 3	RMSE%	10.53 %	7.97 %	-2.56	24.31 %
	MAPE	6.36 %	5.39 %	-0.97	15.25 %
	Subset AVM				
Test 3	RMSE%	9.98 %	7.97 %	-2.01	20.14 %
	MAPE	6.53 %	5.39 %	-1.14	17.46 %

[2]The four features removed are the same four that were pointed out as the weakest contributors.

Chapter 8

Discussion

In this chapter, all the results from Phase Two are discussed and put in context to the outcome presented in Phase One. This will highlight the correlations between the findings in both phases, in addition to critically evaluating the use of AI in the real estate market.

8.1 Feature Impact on LLM and AVM

The analysis and results done throughout this thesis demonstrate the large impact and potential AI has on price estimation in the real estate market. By introducing property-specific features whose values are extracted by a GPT-model, the accuracy generally has increased the performance significantly. Even with the overall performance gain when including LLM-features in an AVM, there is a distinction between the contributing features.

There is a link between the results in Phase One and the results in Phase Two, but there are several factors that play a huge role in a feature's ability to contribute positively to an AVM. The information provided in a real estate advertisement, the formulation of the listing texts, the LLM's ability to understand the goal, and its ability to extract the correct information are all important to have the best foundation for the AVM.

As described in Chapter 5 the majority of features extracted showed a high accuracy, and all had a relatively strong correlation coefficient. Even with these results, the output of the AVM had varying outputs indicating that it isn't necessarily a direct link between the performance in Phase One and Phase Two.

8.1.1 Output Comparison of Phase One and Phase Two

The feature `housing_standard` led to significant performance gain and was the LLM-feature with the highest impact. Its impact is in line with results from a newer study by Oust et al., concluding that the condition of a housing gave consistent performance gain across all models they tested [19]. From Phase One, the feature showed a relatively strong performance in terms of an average correlation of 0.7146, indicating that the LLM hits close to the score given in the manually labeled data. Since the labeled data is based on impressions from the listing text that can be different for each reader, the correlation is a strong performance as it follows the same impression to a large extent. The same goes for `parking_llm`, also with a strong average correlation of 0.7920, the feature increased the AVMs performance as the second most important LLM-feature. On the other hand, even with the high correlation, the LLM's ability to accurately extract values was the worst of all features, which suggests a bigger potential for the feature.

Among the features with the highest accuracy from Phase One had `fireplace` an accuracy of 95.5 %, and a high combined precision-recall score (F-score) of 0.9072. This was also the only feature that could be seen as binary, as only 0.60 % of the samples had more than one fireplace, meaning the remaining 99.4 % of the housing advertisements either had or didn't have a fireplace. By putting these results in context with the impact of the AVM, one can argue that a high F-score for features with binary values seems to have a positive effect on the AVMs output. On the other hand, as this is the only LLM-feature where this is the case it cant be determined with certainty, but by looking at other non-LLM-features with binary values used to train the AVM they also have a positive impact on the model.

When it comes to the other high accuracy features `kitchen`, `bathroom`, `storage_size`, and `outdoor_area_size`, even with high correlation coefficients between 0.8282 and 0.9801, are the results worse than other features. All four of the features had a clear negative impact on the model's output. As the LLM ability to extract the correct information, and other features with the same accuracy and correlation scored way better, there are other factors that play a role in the weak impact on the AVM. During that analysis of the LLM performance in Chapter 5 it was discovered that a good portion of the listing texts lacked the information the LLM was meant to collect, meaning the listing texts themselves seem to have an impact. The missing values made the LLM return blank values ("0") for these features. Further inspection found that the features with the most amount of missing values due to lack of information in the listing text were the same four features that had the most

missing values and were mentioned in Table 5.6. The poor performance of these features when used in the AVM might be explained by the shortage of variation in the data, due to the missing values. Totally, in nearly 80 % of the listing texts, the size of the storage unit wasn't mentioned, while on the other hand, only 26 % of the listing texts didn't mention any recent upgrades of the bathroom, which also had a significant negative impact. Furthermore, the kitchen which wasn't mentioned in 50 % of the data, had nearly the same negative impact on the model, as the bathroom feature had. Even though the AVM program handles these values (NaN values), are the portions with missing values were too big to positively affect the AVM.

The remaining features `storage` and `outdoor_area` had both around 70 % accuracy in Phase One, and a positive impact on the AVM in Phase Two. Their correlation was 0.6080 and 0.7032, and even though the number of storage units had a lower correction, the AVMs performance was greater with `storage` than `outdoor_area`. This could be because the latter is given as numerical values for several categories, while the other is the number of storage units in a housing. As the number of storage units in an apartment is often limited to a maximum of two or three, it's possibly easier for the AVM to find the patterns in this data, rather than in the data of the outdoor area. Another reason might be how the LLM interprets the listing texts if several outdoor areas are mentioned. Language models do not necessarily return the same value for the same input, making it harder for the AVM to find a clear path during training.

8.1.2 Other Factors

The last section implies that both the information and quality of listing texts and the model's ability to extract the correct values are important to have a positive impactful feature for an AVM. During the analysis done in this thesis, several hundred housing advertisements have been read on *finn.no*, and there are several observations that stand out.

The housing advertisements on *finn.no* don't have a fixed format and the wording and formulation are therefore different from advertisement to advertisement. This is because they are written by different real estate agents and brokerage firms, which all have their style of writing. Common to all, and as mentioned earlier, is that the housing advertisements provided seem to under-communicate the condition of the housing unit if it is considered worse. If the housing condition is considered better, the listing text focuses more on the details. This is in line with the observation from Phase Two, where housing standards with lower scores impact the AVM less,

and higher scores more. All of the above influences the LLM and its ability to extract values, in addition to the feature quality of the AVM. The listing texts themself therefore have an important and moderate impact on certain features such as `kitchen` and `bathroom` where details are necessary. The same goes for other features such as `parking_llm`, where better parking options usually were clearly mentioned and explained, but worse or no parking options usually were under-communicated or not mentioned at all.

During the testing process of the LLM, it was discovered that a weaker performance for `parking_llm` also gave a weaker performance for `housing_standard` and vice versa. This can indicate that if the LLM is struggling to extract the correct value for one feature, it does the same for the other. On the other hand, as both features have complex feature rules, they are both dependent on information-rich listing text. These listing texts usually describe all aspects of a housing unit including parking in detail. The lack of information for both features in a listing text seems like a more viable option as the listing text has been a reason for weaker performance for other features as well.

The LLM performs well overall with both high accuracy and correlation compared to the manually labeled data, but when conducting several tests one can experience that the data extracted does vary to a certain degree for each test. By running through the same data several times, and averaging out the values, the noise and inconsistency in the output are filtered out and will be more suited for an AVM. Even though this has proven to increase the correlation for every feature compared to labeled data in this thesis, it will become costly as OpenAIs API is expensive to use. By averaging out five tests, the cost will be five times as expensive. Furthermore to collect enough data to train an even more accurate and viable model, and to prevent overfitting, this cost would be even greater. As *solgt.no* envision using the model on several hundred thousand to millions of housing advertisements in the next years, the cost of running the advertisements several times would be unacceptable.

A solution to this could be hosting a similar open-source model in-house, but the availability of relevant models as well as computational power could be an obstacle. As this thesis utilizes the GPT-model *gpt-3.5-turbo-1106*, other model's performances and ability to extract the correct values can't be determined. Another option could be to rent computational power through the web, but this also comes at a cost based on the amount needed to extract the data with an LLM and train an AVM. Also here could the available open-source language models be a hindrance.

It's obvious from the results in this thesis that the quality of the values extracted

by the LLM comes down to two things, prompt engineering and the listing texts themselves. The prompt engineering done in Chapter 5 shows strong results with high accuracy and correlation towards the labeled data. The quality of the listing texts is an important aspect as described in this chapter, where some features might have high accuracy and correlation, but might as well have little variation in the dataset. There is a clear relationship between the features with great variety in the data and their impact on the AVMs output. In the last test form Chapter 7 was it proven that by introducing the features extracted by the LLM the price estimation was 20.14 % more accurate in terms of RMSE and 17.46 % more accurate for MAPE, than without these features. The deviation was reduced to 7.97 % for RMSE% and to 5.39 % for MAPE.

8.2 Further Work

This thesis has had some limitations in some areas of the research. There have been few studies in the same field, and of the studies available most have been from 2023 or 2024 making it hard to validate their quality. AI is a field that is under continuous development and during the work in this project, several new models have been presented by OpenAI.

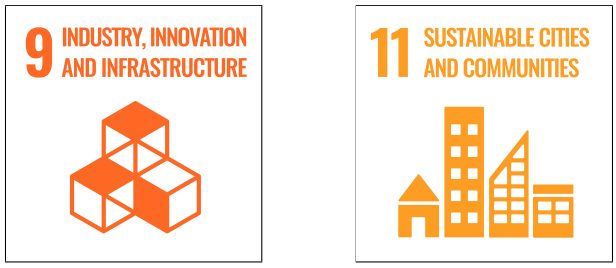
The suggested further work would be to examine other options of accessing models, that potentially could be cheaper to run and operate. There are several opportunities for utilizing LLMs both locally and through online resources. Furthermore would be to inspect other features that could be extracted from the listing texts, and further analyze how the contents in the listing texts could affect an AVM. Another goal should be to gather a greater set of data points than used in this thesis, to train an AVM. This would give a better understanding of how each LLM-feature influences the model's output.

As AI and LLMs get more powerful it's possible to utilize even more information from the housing advertisements. In addition to the textual description in the housing advertisements, it also contains several images that new AI technology could utilize to extract additional information such as conditions and floor plans.

8.3 Sustainable Development Goals

Due to their variety and information rich text are housing advertisements a great resource of finding trends in society. The following sections will explain the project's relevance to the United Nations Sustainable Development Goals.

Both Goal 9 "Industry, Innovation, and Infrastructure" and Goal 11 "Sustainable Cities and Communities" are relevant to this thesis. Using AI and LLMs to extract information can give an innovative insight into infrastructure, housing trends, developments, and the environmental impact of communities. This directly follows the target of the goals mentioned.



(a) **(b)**

Figure 8.1: United Nations Sustainable Development Goal number 9 and number 11 [49][1].

[1] "The content of this publication has not been approved by the United Nations and does not reflect the views of the United Nations or its officials or Member States" [49].

Chapter 9

Conclusion

In this thesis, the goal was to develop an AVM that utilizes an LLM to capture and return quantitative values from the unstructured data in housing advertisement texts for real estate price prediction in Oslo, Norway.

Phase One of this project was to develop a program that accesses an LLM, takes a housing advertisement as input, and returns quantitative values from the unstructured data in the listing text. These values formed LLM-features that should be validated to ensure the LLMs reliability. After going through extensive prompt engineering, the model's overall performance was found to be good and within reasonable limits compared to manually labeled data. The majority of the features show an accuracy of over 90 %, while the remaining features have a relatively high correlation coefficient. Even though there are some differences for each feature, overall results indicate that the model handles the task and input in a good way for all features tested in this thesis.

Phase Two of the project was to develop an AVM that used both existing features and the new LLM-features extracted in Phase One during training and validate the effectiveness and reliability of the new features. After going through data prepossessing and feature engineering, the trained AVM increased the overall performance when using the LLM-featrues. Further testing discovered that using a subset of the LLM-fetures increased the AVM performance by 20.14 % for RMSE% to 7.97 %, and 17.46 % for MAPE to 5.39 %. The features with the most positive impact were `housing_standard` and `parking_llm`.

By comparing the results from Phase One and Phase Two it was discovered that a good feature performance in Phase One didn't necessarily mean a good feature performance in Phase Two. The features `kitchen`, `bathroom`, `storage_size` and

`outdoor_area_size` all had strong performance in Phase One, but were omitted in the AVM due to negative impact on the model. Further investigation revealed that large portions of the listing texts didn't include necessary information for the features, resulting in little variation in the data due to the "missing" values, even when these values are handled.

As the LLM's ability to extract values is high, the information available in the listing texts has proven to be a critical part of the AVM's performance. Despite this, the introduction of highly impactful LLM-features does increase the AVM's performance significantly and has a great potential for price estimation in the real estate industry.

Bibliography

[1] H. Faxvaag Johnsen, 'Llm-based analysis of real estate advertisements', Unpublished., NTNU, Trondheim, Norway, 2023.

[2] J. Weizenbaum, *Computer Power and Human Reason: From Judgment to Calculation*. W. H. Freeman, 1976, ISBN: 978-0-7167-0463-8. [Online]. Available: https://books.google.no/books?id=1jB8QgAACAAJ.

[3] J. Weizenbaum, 'ELIZA—a computer program for the study of natural language communication between man and machine', en, *Communications of the ACM*, vol. 9, no. 1, pp. 36–45, Jan. 1966, ISSN: 0001-0782, 1557-7317. DOI: 10.1145/365153.365168. [Online]. Available: https://dl.acm.org/doi/10.1145/365153.365168 (visited on 7th Mar. 2024).

[4] T. Brown, B. Mann, N. Ryder *et al.*, 'Language Models are Few-Shot Learners', in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 1877–1901. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html (visited on 6th Mar. 2024).

[5] B. Li, G. Fang, Y. Yang *et al.*, *Evaluating ChatGPT's Information Extraction Capabilities: An Assessment of Performance, Explainability, Calibration, and Faithfulness*, arXiv:2304.11633 [cs], Apr. 2023. [Online]. Available: http://arxiv.org/abs/2304.11633 (visited on 6th Feb. 2024).

[6] S. J. Han, K. J. Ransom, A. Perfors and C. Kemp, 'Inductive reasoning in humans and large language models', *Cognitive Systems Research*, vol. 83, p. 101 155, Jan. 2024, ISSN: 1389-0417. DOI: 10.1016/j.cogsys.2023.101155. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389041723000839 (visited on 7th Mar. 2024).

[7] J. Dodge, G. Ilharco, R. Schwartz, A. Farhadi, H. Hajishirzi and N. Smith, *Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping*, arXiv:2002.06305 [cs], Feb. 2020. [Online]. Available: http://arxiv.org/abs/2002.06305 (visited on 8th Mar. 2024).

[8] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi and G. Neubig, 'Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing', *ACM Computing Surveys*, vol. 55, no. 9, 195:1–195:35, Jan. 2023, ISSN: 0360-0300. DOI: 10.1145/3560815. [Online]. Available: https://dl.acm.org/doi/10.1145/3560815 (visited on 6th Feb. 2024).

[9] OpenAI, *OpenAI Platform - Prompt Engineering*, en. [Online]. Available: https://platform.openai.com/docs/guides/prompt-engineering/prompt-engineering (visited on 8th Mar. 2024).

[10] D. P. Foster, 'Featurizing Text: Converting Text into Predictors for Regression Analysis', en, 2013.

[11] E. Cetinoglu, 'Text-Based Prediction of Dwelling Condition', eng, Accepted: 2023-09-21T22:00:48Z, M.S. thesis, 2023. [Online]. Available: https://www.duo.uio.no/handle/10852/105216 (visited on 1st Jun. 2024).

[12] A. Galtier, *Fine-tuning BERT for a regression task: Is a description enough to predict a property's list price?*, en, Dec. 2022. [Online]. Available: https://medium.com/ilb-labs-publications/fine-tuning-bert-for-a-regression-task-is-a-description-enough-to-predict-a-propertys-list-price-cf97cd7cb98a (visited on 1st Jun. 2024).

[13] E. Ghysels, A. Plazzi, R. Valkanov and W. Torous, 'Chapter 9 - Forecasting Real Estate Prices', in *Handbook of Economic Forecasting*, ser. Handbook of Economic Forecasting, G. Elliott and A. Timmermann, Eds., vol. 2, Elsevier, Jan. 2013, pp. 509–580. DOI: 10.1016/B978-0-444-53683-9.00009-8. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780444536839000098 (visited on 12th Mar. 2024).

[14] V. Limsombunchai, C. Gan and M. Lee, 'House Price Prediction: Hedonic Price Model vs. Artificial Neural Network', *American Journal of Applied Sciences*, vol. 1, Mar. 2004. DOI: 10.3844/ajassp.2004.193.201.

[15] K. Birkeland and A. D. D'Silva, 'Developing and Evaluating an Automated Valuation Model for Residential Real Estate in Oslo', en, 2018.

[16] M. M. Lenk, E. M. Worzala and A. Silva, 'High-tech valuation: Should artificial neural networks bypass the human valuer?', *Journal of Property Valuation and Investment*, vol. 15, no. 1, pp. 8–26, Jan. 1997, Publisher: MCB UP Ltd, ISSN: 0960-2712. DOI: 10.1108/14635789710163775. [Online]. Available: https://doi.org/10.1108/14635789710163775 (visited on 8th Mar. 2024).

[17] Q. Truong, M. Nguyen, H. Dang and B. Mei, 'Housing Price Prediction via Improved Machine Learning Techniques', *Procedia Computer Science*, 2019 International Conference on Identification, Information and Knowledge in the Internet of Things, vol. 174, pp. 433–442, Jan. 2020, ISSN: 1877-0509. DOI: 10.1016/j.procs.2020.06.111. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050920316318 (visited on 16th Mar. 2024).

[18] K. Birkeland, A. D. D'Silva, R. A. Füss and A. Oust, 'The predictability of house prices: "human against machine"', eng, *139-183*, 2021, Accepted: 2022-12-05T12:56:33Z Publisher: Asian Real Estate Society, ISSN: 2154-8919. [Online]. Available: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/3035876 (visited on 3rd Jun. 2024).

[19] A. Oust, S. Westgaard, J. E. Waage and N. K. Yemane, 'Assessing the Explanatory Power of Dwelling Condition in Automated Valuation Models', *Journal of Real Estate Research*, vol. 0, no. 0, pp. 1–27, 2023, Publisher: Routledge _eprint: https://doi.org/10.1080/08965803.2023.2280280, ISSN: 0896-5803. DOI: 10.1080/08965803.2023.2280280. [Online]. Available: https://doi.org/10.1080/08965803.2023.2280280 (visited on 27th May 2024).

[20] J. O. Olaussen, A. Oust and J. T. Solstad, 'Real Estate Price Formation: Energy Performance Certificates and the Role of Real Estate Agents', *Journal of Sustainable Real Estate*, vol. 13, no. 1, pp. 1–11, Jan. 2021, Publisher: Routledge _eprint: https://doi.org/10.1080/19498276.2021.2006875, ISSN: 1949-8276. DOI: 10.1080/19498276.2021.2006875. [Online]. Available: https://doi.org/10.1080/19498276.2021.2006875 (visited on 3rd Jun. 2024).

[21] A. Vaswani, N. Shazeer, N. Parmar *et al.*, 'Attention is All you Need', in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html (visited on 15th Mar. 2024).

[22] OpenAI, *New embedding models and API updates*, en-US. [Online]. Available: https://openai.com/blog/new-embedding-models-and-api-updates (visited on 18th Mar. 2024).

[23] OpenAI, *OpenAI Platform - API*, en. [Online]. Available: https://platform.openai.com/docs/guides/text-generation/chat-completions-api (visited on 18th Mar. 2024).

[24] J. J. Webster and C. Kit, 'Tokenization as the initial phase in NLP', en, in *Proceedings of the 14th conference on Computational linguistics* -, vol. 4, Nantes, France: Association for Computational Linguistics, 1992, p. 1106. DOI: 10.3115/992424.992434. [Online]. Available: http://portal.acm.org/citation.cfm?doid=992424.992434 (visited on 18th Mar. 2024).

[25] *Pricing*, en-US. [Online]. Available: https://openai.com/pricing (visited on 19th Mar. 2024).

[26] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016, pp. 95–159, ISBN: 978-0-262-03561-3. [Online]. Available: https://www.deeplearningbook.org/.

[27] G. James, D. Witten, T. Hastie and R. Tibshirani, *An Introduction to Statistical Learning*, en-US. Springer, 2013, vol. 103, ISBN: 978-1-4614-7137-0. [Online]. Available: https://www.statlearning.com (visited on 6th Apr. 2024).

[28] M. S. Mahmud, J. Z. Huang, S. Salloum, T. Z. Emara and K. Sadatdiynov, 'A survey of data partitioning and sampling methods to support big data analysis', *Big Data Mining and Analytics*, vol. 3, no. 2, pp. 85–101, Jun. 2020, Conference Name: Big Data Mining and Analytics, ISSN: 2097-406X. DOI: 10.26599/BDMA.2019.9020015. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9007871 (visited on 5th Apr. 2024).

[29] Y. J. Soofi, Y. Gu and J. Liu, 'An adaptive Physics-based feature engineering approach for Machine Learning-assisted alloy discovery', *Computational Materials Science*, vol. 226, p. 112 248, Jun. 2023, ISSN: 0927-0256. DOI: 10.1016/j.commatsci.2023.112248. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0927025623002422 (visited on 11th Apr. 2024).

[30] F. Nargesian, H. Samulowitz, U. Khurana, E. B. Khalil and D. Turaga, 'Learning Feature Engineering for Classification', en, in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, Melbourne, Australia: International Joint Conferences on Artificial Intelligence Organization, Aug. 2017, pp. 2529–2535, ISBN: 978-0-9992411-0-3. DOI: 10.24963/ijcai.2017/352. [Online]. Available: https://www.ijcai.org/proceedings/2017/352 (visited on 11th Apr. 2024).

[31] A. Zheng and A. Casari, *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. O'Reilly Media, 2018, ISBN: 978-1-4919-5319-8. [Online]. Available: https://books.google.no/books?id=sthSDwAAQBAJ.

[32] 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*, arXiv:1603.02754 [cs], Aug. 2016, pp. 785–794. DOI: 10.1145/2939672.2939785. [Online]. Available: http://arxiv.org/abs/1603.02754 (visited on 15th Apr. 2024).

[33] J. H. Friedman, 'Greedy Function Approximation: A Gradient Boosting Machine', *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 1999, Publisher: Institute of Mathematical Statistics, ISSN: 0090-5364. [Online]. Available: https://www.jstor.org/stable/2699986 (visited on 15th Apr. 2024).

[34] A. K. Bhoi, *5G IoT and edge computing for smart healthcare* (Intelligent data centric systems), English. [S.l.]: Academic Press, 2022, ch. 8.3.7, ISBN: 9780323906647. [Online]. Available: https://www.sciencedirect.com/science/book/9780323905480.

[35] J. Bergstra and Y. Bengio, 'Random Search for Hyper-Parameter Optimization', en, 2012.

[36] A. Hjort, J. Pensar, I. Scheel and D. E. Sommervoll, 'House price prediction with gradient boosted trees under different loss functions', en, *Journal of Property Research*, vol. 39, no. 4, pp. 338–364, Oct. 2022, ISSN: 0959-9916, 1466-4453. DOI: 10.1080/09599916.2022.2070525. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/09599916.2022.2070525 (visited on 11th Mar. 2024).

[37] *Standard for bygningstype / Matrikkelen*. [Online]. Available: https://www.ssb.no/klass/klassifikasjoner/31 (visited on 20th Apr. 2024).

[38] openai, *OpenAI Platform*, en. [Online]. Available: https://platform.openai.com/docs/libraries/python-library (visited on 2nd Jun. 2024).

[39] *Scikit-learn: Machine learning in Python — scikit-learn 1.5.0 documentation*. [Online]. Available: https://scikit-learn.org/stable/index.html (visited on 2nd Jun. 2024).

[40] *XGBoost Documentation — xgboost 2.0.3 documentation*. [Online]. Available: https://xgboost.readthedocs.io/en/stable/index.html (visited on 2nd Jun. 2024).

[41] P. Fränti and R. Mariescu-Istodor, 'Soft precision and recall', *Pattern Recognition Letters*, vol. 167, pp. 115–121, Mar. 2023, ISSN: 0167-8655. DOI: 10.1016/j.patrec.2023.02.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865523000296 (visited on 11th May 2024).

[42] F. T. Liu, K. Ting and Z.-H. Zhou, 'Isolation Forest', Jan. 2009, pp. 413–422. DOI: 10.1109/ICDM.2008.17.

[43] H. I. W. Wolstad and D. Dewan, 'Machine learning as a tool for improved housing price prediction : The applicability of machine learning in housing price prediction and the economic implications of improvement to prediction accuracy', eng, Accepted: 2021-04-27T07:51:16Z, M.S. thesis, 2020. [Online]. Available: https://openaccess.nhh.no/nhh-xmlui/handle/11250/2739783 (visited on 3rd Jun. 2024).

[44] X. Wang, S. Zhai and J.-L. Chen, 'Research on House Price Forecast Based on Hyper Parameter Optimization Gradient Boosting Regression Model', in *2020 8th International Conference on Orange Technology (ICOT)*, Dec. 2020, pp. 1–6. DOI: 10.1109/ICOT51877.2020.9468726. [Online]. Available: https://ieeexplore.ieee.org/document/9468726/authors#authors (visited on 3rd Jun. 2024).

[45] *XGBoost Parameters — xgboost 2.0.3 documentation*. [Online]. Available: https://xgboost.readthedocs.io/en/stable/parameter.html (visited on 15th May 2024).

[46] *3.4. Metrics and scoring: Quantifying the quality of predictions*, en. [Online]. Available: https://scikit-learn.org/stable/modules/model_evaluation.html#mean-squared-error (visited on 2nd Jun. 2024).

[47] *Metrics and scoring: Quantifying the quality of predictions*, en. [Online]. Available: https://scikit-learn.org/stable/modules/model_evaluation.html#mean-absolute-percentage-error (visited on 2nd Jun. 2024).

[48] *Bydelsfakta*, nb. [Online]. Available: https://bydelsfakta.oslo.kommune.no (visited on 22nd May 2024).

[49] R. Neshovski, *UN, Sustainable Development Goals*, en-US. [Online]. Available: https://www.un.org/sustainabledevelopment/ (visited on 1st Jun. 2024).

Appendix

A Model Behavior for LLM

The following is the default model behavior for the LLM. It is constructed as one string, including a set of rules for each feature the model extracts. For simplicity it is given as sections here. The *model_behavior* a continuation from the preliminary project, and the formulation in this thesis are therefore similar to the prompts from the project [1].

Format

Din oppgave er å analysere innholdet i boligannonsen. Du skal returnere følgende informasjon om boligen på følgende format: 'parking': 'p_score', 'kitchen': 'k_score', 'bathroom': 'b_score', 'fireplace': 'f_score', 'storage': 's_score', 'storagesize': 'ss_score', 'outdoorarea': 'o_score', 'outdoorareasize': 'os_score', 'dwellingstandard': 'd_score'. Verdiene til 'x_score' er alltid et tall som velges på følgende måter:

Parking

'p_score' er det alternativet mellom 1 og 7 som passer best fra følgende liste, svar kun med tallet. Gjelder kun parkering til bil:

1 Ingen parkering eller parkering ikke nevnt.

2 Det er parkering eller gateparkering i området, men ikke eksklusivt for boligen.

3 Borettslaget/sameiet har flere parkeringsplasser på deling, borettslaget disponerer flere parkeringsplasser.

4 Boligen har muligheten til å leie en parkeringsplass.

5 Borettslaget/sameiet har parkeringsplasser som er tildelt basert på venteliste/ansiennitet,

6 Boligen har egen utendørs parkering eller biloppstillingsplass eller carport, hver seksjonseier har parkering, leiligheten har medfølgende parkeringsplass.

7 Boligen har egen parkering i en garasje.

Kitchen

'k_score' er det fulle årstallet kjøkkenet er fra eller sist det ble renovert, pusset opp eller oppgradert, eventuelt boligens byggeår. Hvis årstall ikke er nevnt, returner 0.

Bathroom

'b_score' er det fulle årstallet badet er fra eller sist det ble renovert, pusset opp eller oppgradert, eventuelt boligens byggeår. Hvis årstall ikke er nevnt, returner 0.

Fireplaces

'f_score' er antall peiser/ildsteder/vedovner. Om det ikke er nevnt er f_score=0. Er peis nevnt, men antallet ikke kommer tydelig frem, anta at antallet er 1.

Storage Unit

's_score' er antall boder boligen disponerer. Om bod er nevnt, men antallet ikke kommer frem er s_score=1.

Storage Unit Size

'ss_score' er størrelsen i kvadratmeter på boden. Er det flere boder er det totalt kombinert areal av bodene. Kommer ikke størrelsen tydelig fram er ss_score=0.

Outdoor Area

'o_score' er det alternativet mellom 0 og 10 som passer best fra følgende liste, svar kun med tallet:

0 Uteområde ikke nevnt.

1 Privat bakgård.

2 Felles bakgård.

3 Privat hage.

4 Felles hage.

5 Privat terrasse.

6 Felles terrasse.

7 Privat balkong.

8 Felles balkong.

9 Privat takterrasse.

10 Felles takterrasse.

Outdoor Area Size

'os_score' er størrelsen i kvadratmeter på balkongen, terrassen eller takterrassen har. Har boligen flere er det kombinert areal av balkongene, terrassene eller takterrassene. Er det hage, bakgård eller liknende, returner 0.

Housing Standard

'd_score' der score er det alternativet mellom 0 og 9 som passer best fra følgende liste, svar kun med tallet:

0 Ikke mulig å fastslå standard.

1 Boligen krever totalrenovering, den har alvorlige problemer og mangler som krever omfattende arbeid.

2 Boligen har lav standard som trenger betydelige oppgraderinger og renovering for å bli beboelig.

3 Enkel standard der boligen er funksjonell, men har behov for modernisering og mindre reparasjoner.

4 Moderat standard der boligen er i grei tilstand, men har behov for estetiske oppdateringer og enkelte reparasjoner.

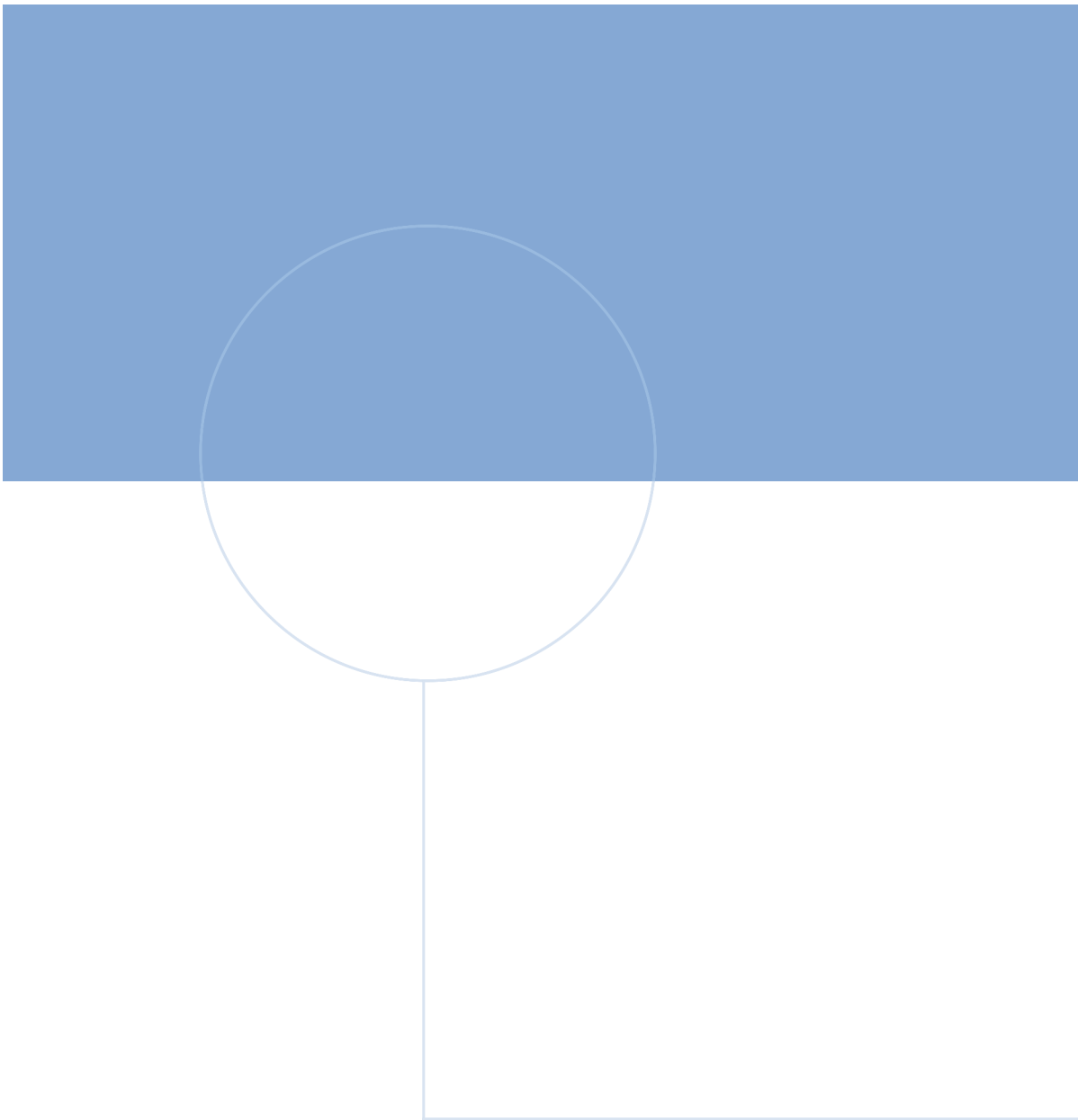
5 God standard der boligen er velholdt med noen moderne fasiliteter, men kan ha behov for mindre oppgraderinger.

6 Meget god standard der boligen har flere oppdaterte funksjoner, god vedlikeholdsstatus og kun mindre behov for oppgraderinger.

7 Høy standard der boligen er nylig oppusset eller bygget med kvalitetsmaterialer, moderne fasiliteter og lite til ingen behov for oppgraderinger.

8 Svært høy standard der boligen fremstår som luksuriøs med toppmoderne fasiliteter, eksklusive materialer og detaljer av høy kvalitet.

9 Eksepsjonell standard der boligen har unike kvaliteter, skreddersydde løsninger, høy teknologi og bruk av de mest eksklusive materialene og fasilitetene tilgjengelig.



NTNU

Norwegian University of
Science and Technology