UiO : Department of Informatics
University of Oslo

Text-Based Prediction of Dwelling Condition

Thesis submitted for the degree of
Master in Data Science: Language technology
60 credits

Ece Cetinoglu
Master's Thesis, Spring 2023

The cover page was designed by Martin Helsø.

Abstract

The exploration of regression analysis based on text is understudied compared to other tasks, and there are limited literature on this topic, with very few studies delving into this specific task. This thesis aims to contribute to this topic while solving a real-life problem, i.e., not by experimenting on a benchmark. Our objective was to predict the condition score, a value between 0 and 3, of dwellings in the real estate market in Norway based on the features extracted from the textual content of their respective listing advertisements. Usually, the condition of a dwelling is described in a publicly available condition report written by a certified assessor. We aspired to obtain a benchmark method that can be utilized to predict the score in case of missing condition reports. We approached the regression task by creating progressively more complex models. We experimented with these models to improve the accuracy, for instance by hyperparameter tuning and oversampling. The results have shown that while the BERT-based regression models demonstrated superior performance, simpler regression methods trained on features extracted from text using Bag of Words (BoW) approaches produced comparable results. Among the models explored in this thesis, the gradient boosting regression model trained on bag-of-words features, and the unfrozen NB-BERT$_{\text{BASE}}$ model both trained on the oversampled data set, stood out with noteworthy results, yielding mean absolute errors of 0.1835 and 0.1578 respectively. The results obtained in this thesis present convincing evidence that text-based regression analysis with BoW-based and BERT-based approaches is a viable and promising downstream task. This thesis can potentially contribute to the advancement of knowledge in the real estate market and introduces a novel application of Natural Language Processing, a field that traditionally emphasizes classification tasks rather than prediction of continuous variables.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to Aslak Wigdahl Bergersen and Andrey Kutuzov for being great supervisors and being patient with me. Your invaluable guidance and exceptional pedagogical skills have been instrumental in shaping my thesis.

I would also like to like to thank deeply to my colleague, Sebastian Mitusch for his insightful tips and constructive critique of my writing, and my former colleague, Herman T. Holmøy for developing the web crawler that I utilized to gather essential data for this thesis. I am grateful to Boligmappa AS for generously funding the computational resources essential for conducting the experiments and analyses in this thesis.

Finally, I would like to thank my family for always believing in me, especially during those times when I did not. Living in a different country than you while writing a thesis has been a challenge, but your unwavering support has transcended geographical boundaries. Through hours of FaceTime, you have been a constant source of encouragement. Your unconditional love and understanding have been crucial in keeping me motivated and focused on this journey.

As my time as a student at the University of Oslo comes to an end, I take a look back and see five great years with excessive amounts of joy and excitement, but also self-growth and friendships that I hope will last life-long. The years spent here and the people I have spent them with have shaped me into the woman that I am today, and I will not take that for granted.

Thank you.

Ece Cetinoglu

Contents

1 Introduction

Having a secure roof over our heads, is one of our most fundamental needs. On an individual level, it is a necessity that provides us with physical shelter, as well as a sense of safety and privacy. In Norway, homeownership is given great importance not just for individuals, but for the entire society. After the Second World War, and especially after the deregulation of the housing market in the 1980s, the Norwegian government has aimed to ensure that most of the Norwegian population is able to purchase and retain their homes through the Norwegian housing policy (Nordic Journal of Housing Research, 2021). The Norwegian government encourages home ownership through various incentives, such as low-interest loans and tax deductions. As a consequence of this policy, Norway is one of the countries with the highest homeownership rate, with 81.9% of the population fully or partially owning the dwelling they live in (Statistics Norway, 2023a). For comparison, the homeownership rate in some of the other Scandinavian countries, Finland, Sweden and Denmark, were 70.3%, 64.9%, and 59.2%, respectively (Eurostat, 2023).

Regardless of the homeownership rate, real estate is one of the most important categories of assets in any economy. Under normal circumstances, real estate prices in Norway generally followed an increasing pattern. However, it is important to note that there have been exceptional events that have affected the prices negatively, such as the Norwegian Banking Crisis in 1988-1992, which had a significantly negative effect on the house price indices (Eitrheim & Erlandsen, 2004). Other notable events were the Financial crises that caused the real estate prices in Norway to go down by 11 percent (Dreyer, 2018) in the summer of 2008, and the COVID-19 global pandemic which caused the average office rental prices in Oslo to go down by 4 percent (Hangaard, 2020). Notwithstanding the effects of these extraordinary events on the real estate market, real estate prices have an increasing trend (Eiendom Norge, 2023) since the 70s up to today, with the aggregate house price index rising by almost 1300% between the 70s to 2003 (Eitrheim & Erlandsen, 2004). The inflation in the same period, however, was only 859,31% (Statistics Norway, 2023b).

On an individual level, purchasing a house is often the most significant investment a person makes in their lifetime. Individuals in Norway tend to allocate a substantial part of their savings on purchasing a house, resulting in high housing debts. According to Statistics Norway, the average household debt in Norway in 2012 amounted to NOK 1.1 million (Statistics Norway, 2012). Despite the high values of debt, purchasing a house has generally proven to be rewarding in the long term due to the consistent increase in housing prices relative to inflation. In addition to the economic advantages that come with owning a home, many people might also argue that having your own space and being able to personalize it without any (or, with very few) restrictions gives a feeling of comfort and freedom.

Yet, while making a crucial decision like purchasing a home, many individuals have limited access to the information real estate agents and financial institutions sit on. The only free and open platform for insight into the real estate market in Norway is hjemla.no. Hjemla provides its users with valuable information to guide them to make well-informed decisions during the processes of purchasing, selling, or renovating their homes. Some of the offerings on the

platform are insight into previous sold prices and public data for all homes, an automated valuation, and renovation calculators. One missing part of the current offering is the lack of information on dwelling quality. This could both be of direct value to users, but also improve existing offerings like the automated valuation models. An important question is, how do we measure a qualitative variable like the level of maintenance in a dwelling?

During the process of selling a dwelling, the seller hires an assessor who writes a detailed appraiser report describing the dwelling's condition prior to sale. The contents of this report can be considered as industry standard for the objective measure of the quality of the dwelling. However, the condition reports are not available for every dwelling. We might have to consider other sources to estimate the dwelling condition, such as online pictures in the listing, or the judgment of a potential buyer after visiting a house. Another source of information to consider is the advertisement description in the listings. While this text also may be biased depending on by whom it was written, it may still provide the reader with more detailed information. In this thesis, we will explore how we can extract new features from the advertisement text regarding the condition of a house, and whether this text can be used to successfully predict the condition of homes that we do not have an appraisal report for.

Previous studies, (Ooi, Le, & Lee, 2014), (Mamre & Sommervoll, 2022) have mostly focused on the correlation between the level of maintenance on the value of a dwelling, and demonstrated that these two aspects are correlated. Nevertheless, the objective of this thesis differs as we would like to estimate the dwelling condition itself, rather than to examine its impact on the value. Furthermore, to the best of our knowledge, there have been no studies conducted with the goal of predicting dwelling conditions. Even if our objective is understudied, previous studies have suggested some useful manual methodologies for extracting the condition of a dwelling from the sources and data at our disposal (Oust & Yemane, 2021). This thesis aims to fill in some of the gaps in the literature by incorporating the advertisement texts to a greater extent in these model, to build a more robust, and objective algorithm to predicting the condition of dwellings. Moreover, we approach our objective as a regression task and conduct regression analysis based on text with common techniques within the Natural Language Processing (NLP), a field that traditionally emphasizes classification tasks rather than prediction of continuous variable.

In Chapter 2, we present the previous work done on assessing or utilizing the dwelling quality in literature and discuss studies where text-based regression models were conducted. We also shortly explain the real estate market in Norway in order to provide more context about the data sets described in Chapter 3. In Chapter 4, we present and explain the models and NLP techniques employed to accomplish our objective of estimating the condition score of dwellings based on advertisement text. The results obtained by our models are presented in Chapter 5, and further discussed in Chapter 6.

2 Background

2.1 Previous work on dwelling quality

There has been a lot of work on trying to connect the condition or the rate of depreciation of a dwelling to its sale or rental price. Many of these studies discussed in this chapter have argued for and shown that the level of maintenance and the price of the house is positively correlated. However, most of the previous work has focused on the impact of the dwelling condition on its value in terms of the sale price or rental price. Moreover, when assessing the value of a dwelling, researchers have often included other attributes, such as structural features of the dwelling and spatial attributes. To what extent the condition of a dwelling contributes to the price has also been shown to depend on its other features, such as the size of the dwelling and the number of rooms (Mathur, 2019).

(Wilhelmsson, 2008) is one of the few works in the literature that examines the dwelling condition in a more granular way. The study uses 640 observations, all of which are detached houses in Stockholm, Sweden, and investigates the depreciation rate of the houses depending on their level of maintenance. Sources of the information in the data set are the tenants who were surveyed. In contrast to other studies, this paper takes into account the condition of some areas and aspects of a dwelling as separate features. There are, in addition to the attribute that measures the overall quality of the house, binary attributes representing the quality of the exterior and the interior. The overall quality is the value that is appraised for tax assessment purposes. The attributes representing the indoor condition include the condition of the kitchen, electricity facilities, and the laundry room. These attributes were acquired from the surveys, in which the house owner states whether the kitchen, laundry, and electricity facility need repairs. Similarly, the attribute representing the outdoor condition informs whether the outdoor facilities need improving. This study has shown that the need of renovating the indoors and outdoors affects the price negatively, where the need of rehabilitating the kitchen had a greater impact than the other attributes. One shortcoming of this study is that the binary attributes on the indoor and outdoor conditions are hard to interpret and they do not provide details about the degree of maintenance that is needed. For instance, it is not specified whether the kitchen only needs to get a new coat of paint, or it needs a complete renovation. Hence, the need of repairing the interior or exterior can have very different meanings for two different houses.

Another study that has examined the condition of different aspects of a house and their impact on the price separately is (Ooi et al., 2014). This study investigates over a hundred thousand sale transactions of apartment units in Singapore and the impact of construction quality on the sale prices. The quality of construction, measured by a construction scoring system introduced in Singapore in 1989 (*CONQUAS, Singapore Building and Construction Authority*, n.d.), is an aggregated numerical value from 1 to 100, which is assessed to the units according to national standards. The final score is an aggregation of three components: 1) the structural works, i.e the strength of the concrete, 2) architectural works, i.e internal finishes, external works, and 3) mechanical & electrical (M&E) works, i.e air conditioning, sanitary and plumbing. The study shows that the construction quality score alone and the components separately impact the unit price positively and are of high significance. One weakness of

this study is that the construction quality score is only assessed to new homes. Thus, the impact of the construction quality was only measured on units that are brand new and do not need any maintenance. Although the conclusions drawn from this paper may not be applicable to data set used in this thesis, it is clear that high quality houses is something demanded and appreciated by the buyers.

A more recent study (Oust & Yemane, 2021) aimed to investigate whether the inclusion of dwelling condition attributes would improve the performance of an Automated Valuation Model (AVM) that estimates the price of a dwelling based on its features. Their data and the source of information for determining the dwelling condition are very similar to ours, including over 11 thousand observations from Oslo, the capital of Norway, and the second and fourth biggest cities in Norway, Bergen, and Trondheim, respectively. Most of their data, however, belonged to houses in these cities due to the fact that they struggled with merging the dwelling condition data to the dwelling data confidently, dooming them to exclude most of the observations which were apartments from their experiments. The condition score of a single dwelling is calculated from its accompanying appraisal report, which contains information about a collection of checkpoints evaluated. The number of checkpoints included in the report can vary from unit to unit, hence the condition scores are combined into several categories, with the average score of checkpoints in a category being the final score for that category. Some of the categories were the condition of the kitchen, bathroom, windows, doors, and roof. This approach provided information on the condition of separate parts of the unit and allowed them to examine how the condition of each minor aspect impacts the big-picture. Some of the important findings of this study were, the inclusion of dwelling condition features has improved all of their models in all three cities to a similar extent, and the information about the bathroom condition had the highest impact on the price across the models. Similarly, the condition of doors, roofs, and exterior extensions such as balconies and terraces, were found to have a low impact on the price and were statistically insignificant. While these findings are important and insightful for this thesis, we must not forget the exclusion of the data on apartments and the lack of data from other places in Norway. What is more desirable for us is to build a model that is applicable to all kinds of dwellings anywhere in the country.

Finally, another study from Norway that has investigated the impact of the level of maintenance of a dwelling on the dwelling price in terms of renovation premiums was (Mamre & Sommervoll, 2022). They have divided over 10 thousand dwellings into 4 categories: 1) fully renovated, 2) partially renovated, 3) neutral (neither renovated nor unmaintained), and 4) unmaintained. 85.5% of the observations in their data were apartments, and 65.1% of data were classified as "neutral". All of the observations were from Oslo, Norway. They have estimated that the maintained dwellings value 5-7% more than the others, and the unmaintained dwellings cost 9-10% less. They have also found that the amount of which the level of maintenance contributes to the total price depends on the time of the sale. The renovation premium and the contribution of the renovation was seemingly low when the real estate market was "hot". Similarly, the negative premium for unmaintained dwellings is reduced in a more heated market. One limitation of this study is that the dwelling condition was computed based on reading searching of the description texts in the listings. This classification

method might not be reliable, as an advertisement text is often shallow and does not focus on the smaller details to the same degree like an appraisal report written by a professional. Moreover, the feature extracted from the listing that represents the level of maintenance disregards information about which parts of the dwelling need maintenance or which parts are well-maintained.

2.2 Previous work on using NLP for regression

Natural Language Processing (NLP) is a branch of Artifical Intelligence (AI) that focuses on creating methods and algorithms that uses natural language data as input that aim to solve a machine learning task. (Goldberg, 2017). Some tasks that can be solved by using NLP are document classification, where the input data is the text from the documents and the output is some discrete information about the document (e.g who it is written by or what source it is from), and sentiment analysis, where the input data is a collection of documents or sentences and the output is whether the data is written from a positive, negative or neutral point of view. While NLP is widely used for classification tasks where labels assigned to the data are discrete values that represents a class, there are less examples of regression tasks and predicting continuous variables with NLP.

Age prediction, a commonly studied task in NLP, is a regression task aiming to predict a continuous variable, the age of the author or writer, based on the texts thee authors have written. An intermediate study that approaches to age prediction by utilizing text was conducted by (Nguyen, Smith, & Rose, 2011). Their data set consisted of three different corpora, each corpus containing text mainly written by people from different age groups. They extracted unigrams and bigrams from their corpus and fed the features into their linear regression models. They have created three individual models trained on the corpora individually, as well as joint models where all three corpora were taken into account. Their experiments have resulted in high correlations between the predicted and actual values for age, as well as mean absolute errors between 4.1 and 6.8 years, showing that there are signals in the corpora indicating the author's age. This paper has also showed that even the linear regression models fed with unigrams, relatively simple method for extracting features from text, has satisfactory performance.

One of the first studies that has used NLP for making predictions on continuous variables and introducing the text regression was (Kogan, Levin, Routledge, Sagi, & Smith, 2009). Their goal was to forecast financial volatility of companies from the text in their their annual reports. They have used support vector regression (SVR) models, and the features extracted from the text and fed into the model were 1) term frequency (TF), 2) term frequency-inverse document frequency (TF-IDF) and 3) log-normalization of TF. They have compared different models, some of which estimating the volatility based on the historical volatility values without textual features, estimating with textual features only, and combining historical features with textual features. Model performances were measured in terms of mean squared error. They have demonstrated that the models with only textual features have performed almost as good as the model without any textual features, where the best text-only model yielded an MSE of 0.1667 and the best historical model achieved an MSE of 0.1576. It was also demonstrated that the combined models outperforms the others, where the

best-performing combined model yielded an error of 0.1538. They have shown that textual features add a lot of information to the model in addition to the historical features, and a continuous variable such as volatility can successfully be predicted from text using uncomplicated models.

A more recent study that is similar to Kogan et.al.'s was conducted by (Dereli & Saraclar, 2019). They also aimed to predict volatility, based on a set of input features with and without information extracted from the annual reports. They have used relatively more complicated models, where they fed static and non-static word embeddings into convolutional neural network models with a convolution layer that consisted of tri-grams, four-grams and five-grams. Finally, they measured the model performances in terms of mean squared error (MSE) and demonstrated that their models gained performance in the presence of textual features.

While these studies mentioned above assess the impact of textual features in linear regression models predicting continuous variables, most of them are older than 10 years old, and do not cover state-of-the-art methods in NLP field, such as the Large Language Models (LLMs). To the best of our knowledge, regression analysis with LLMs, such as Generative Pre-trained Transformers (GPT) (Radford, Narasimhan, Salimans, & Sutskever, 2018) and Bidirectional Encoder Representations from Transformers (BERT) (Devlin, Chang, Lee, & Toutanova, 2018) is rarely performed. In the forthcoming chapters, we explore and discuss the predictive power of BERT-based approaches in estimating the condition score, as well as more traditional approaches like the bag-of-words methods.

2.3 Real estate in Norway

Before diving into data and models, we shortly explain the real estate market, the cadastral register system and the process of purchasing real estate in Norway in order to provide context about the datasets utilized in the upcoming chapters.

2.3.1 Matrikkelen

Like most countries in the world, Norway has its own cadastral register system for to keep record of real estates and properties in the country, Matrikkelen. Matrikkelen contain information about properties, such as property borders, buildings and adresses registered on the property, etc. Matrikkelen consists of several "matrikkel units", each of them denoted by a unique number, called "matrikkelnumer" (cadastral key). The registration number is composed of five numbers, community identification number, cadastral unit number, property unit number, lease number, and unit number.

Real estate in Norway are identified by unique matrikkel numbers, and this number is used later in this thesis to identifiy and merge data acquired from three different sources, explained in Chapter 3.

2.3.2 Purchasing real estate

Purchasing real estate in Norway is a highly regulated process. A normal sale will follow these steps: Firstly, the seller(s) hire a real estate agent and an assessor. It is constitutionally mandated that there is an accompanying appraiser

report for all homes that are listed as for sale, thus the assessor writes a detailed condition report and evaluates the status of the unit that is listed as for sale. Secondly, the real estate agent creates a detailed property description which contains all information available for the unit, including the condition report. After these steps, the property can be advertised for sale. Finally, the real estate agent arranges an open house for potential buyers, who get the opportunity to view the unit.

2.3.3 Condition reports

Prior to the sale of a real estate property, the sellers hire an assessor, who writes a detailed appraiser report describing the dwelling's condition. This appraiser report consists of a number of checkpoints that are relevant to the unit, and each checkpoint is scored from 0 to 3. The scores can be described as the following:

- Score 0: Perfect condition and no symptoms of deterioration of condition
- Score 1: Mild symptoms of deterioration of condition
- Score 2: Moderate symptoms of deterioration of condition
- Score 3: Strong symptoms of deterioration of condition, needs to be fixed immediately

3 Data

The data utilized in this thesis has been provided or collected from 4 primary sources: hjemla.no, Vendu, Unbolt AS, and boligmappa.no. This chapter aims to provide an explanation of the data acquisition and the extraction of features from each dataset, along with details on data cleaning and merging all four data sets to a single comprehensive data set that includes the essential information. Additionally, we provide a brief overview of data contents and characteristics, and relevant descriptive statistics.

In Section 3.1, we provide a detailed description of the data set containing linguistic features extracted from advertisement texts crawled from Finn.no. Then, three separate data sets containing condition reports with elaborate information on dwellings, as well as the process of extracting information and cleaning these data sets prior to conducting our experiments were outlined in Section 3.3. Later, we explain the methodology used to compute a singular condition score for each dwelling in Section 3.4. Finally, we define the merging algorithm we developed to combine all four data sets in Section 3.5. Figure 3.1 and present the final data set that was utilized for feeding our models in the subsequent sections.

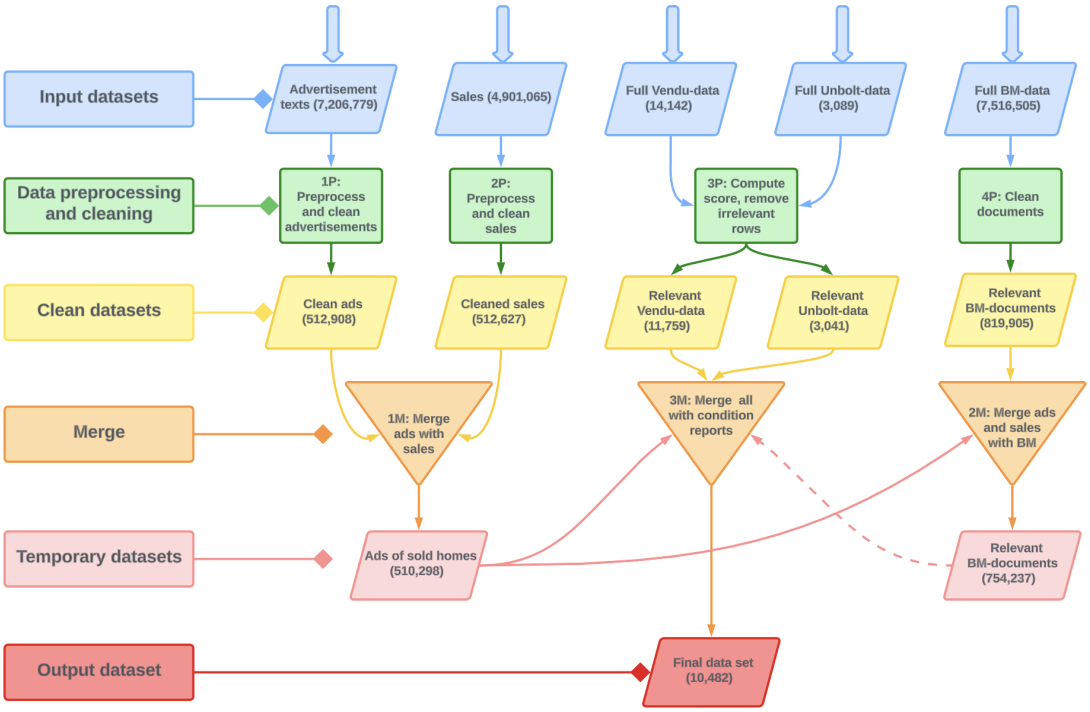

Figure 3.1: Flowchart explaining the order of data cleaning and pre-processing, as well as how all datasets were merged to achieve the final data set we have used in our models. The rectangles represent a procedure the datasets undergo. The parallelograms represent the input and output datasets. A triangle indicates a merging procedure performed on two or more datasets.

3.1 Advertisement texts

A part of the service on Hjemla is to show previous housing advertisements on Finn.no, one of the most used advertisement websites in Norway (Wikipedia, 2023). We were granted access to Hjemla's database which has stored information about 7.2 million advertisements from Finn, containing the advertisement texts and advertisement titles, as well as the date of creation and when the advertisements were edited. They also had a status indicator for each advertisement, stating whether the advertisement is still active (e.g. the dwelling has not been sold) or inactive (e.g. the dwelling was sold).

In Norway, there are two official written standards of the Norwegian language: Bokmål and Nynorsk. Both of them are recognized by the Norwegian government. While there is no statistics stating the total number of Nynorsk users in Norway, it is estimated that between 10-15% of the Norwegian population utilizes Nynorsk as their written language (Vikør, Jahr, Berg-Nordlie, & Thorvaldsen, n.d.). Therefore, it is safe to assume that there are some advertisement texts written in Nynorsk present in the data set. After reviewing the texts briefly, we have also observed that some most commonly used languages on Finn were other Scandinavian languages such as Danish or Swedish. As a result of these observations, we have decided to focus on the advertisements written in Bokmål only, and data cleaning was necessary to filter out advertisements that are not written in this standard. We performed language detection on the advertisement texts using `polyglot`[1].

To verify the language detector, we identify a subset of 10,000 advertisement texts and assess the performance of the detector on this subset. Some of the findings was that the detector often misclassified the language of texts that are too short (e.g. the advertisement text only contains the address or the cadastral registry number). We also observed that the language detector struggled with correctly identifying the language in advertisement texts that contained special characters (e.g. "é" or "è") that does not exist in the Norwegian alphabet. These were labeled as "unknown language". Around 200 of 10 thousand advertisements were annotated as Norwegian Nynorsk. A subset of these texts was manually reviewed to verify the accuracy of the labeling, revealing that the language detector yielded satisfactory performance.

The language detector was then applied on the full Finn-data. Around 97.24% of advertisements were identified as Norwegian Bokmål. We eliminated advertisement texts that were written in languages other than Norwegian Bokmål from the dataset to ensure that the data used for analysis and processing specifically pertained to Norwegian text, thus maintaining the integrity and consistency of the dataset.

The process of cleaning and pre-processing the advertisement data set, displayed in Figure 3.1(1P), contained of the following steps:

1. Remove irrelevant rows:
    (a) Remove ads that does not belong to a real estate property
    (b) Remove rows with no advertisement text.
    (c) Apply language detection and only keep Norwegian bokmål.

[1]Language detection with polyglot: https://polyglot.readthedocs.io/en/latest/Detection.html

2. For the regression models described in the next chapter, preprocess the text:

   (a) Convert all text to lowercase.

   (b) Remove numerical digits from the text.

   (c) Remove punctuation marks from the text.

   (d) Trim unnecessary whitespace from the text.

   (e) Exclude stop-words from the text.

   (f) Tokenize the text using whitespace as a delimiter.

We made sure to keep both processed and raw advertisement texts, as the regression models trained in the next chapter need different types of inputs features. After performing data cleaning and preprocessing, we obtained a total of 512,908 relevant rows from the Finn-dataset.

After reviewing many real estate listings on Finn.no, some similarities were observed among the advertisement texts. Brand-new, almost-perfect homes also have descriptions indicating how perfect they are. The advertisement texts for less maintained or highly damaged homes are written in a positive manner as well, highlighting the dwelling's good sides, but also endorsing its lacks and flaws to suggest that there is a "potential" in that home and potential for increasing the value of it if the necessary renovations are done. Another claim that is often made about the homes with bad condition is that when someone buys an old dwelling, they get the opportunity to renovate it and personalize it in their own way. Below, we present two advertisement texts from Finn, the former belonging to a house in bad condition, whereas the latter is of a modern apartment from 2021. These texts are not direct quotes from Finn, but text translated from Norwegian to English. Parts of the advertisement text are left out.

> (1) "Detached house with workshop in picturesque surroundings. (...) The house **requires care** and **renovation** but has great **potential** to become a beautiful home or vacation property (requires rezoning). (...)"

> (2) "Delightful 2-bedroom freehold apartment **from 2021** with a garage space. (...) a tasteful 2-bedroom freehold apartment **from 2021**, located on the ground floor with an end position. (...) Inside, there is a **refined style**, **smart solutions**, and **modern features** such as balanced ventilation and underfloor heating. (...) The bathroom is **timeless** with tiled walls (...) Welcome!"

The first advertisement states that the house requires care, but it lies in a nice area. The second advertisement focuses on the build year of the apartment, as well as its modern features such as the bathroom being timeless and the apartment being equipped with smart solutions.

3.2 Sold dwellings

Another service Hjemla provides its users with is to give them information about sales that has happened in Norway. Users can access details such as the dates

of the sales and the corresponding sale prices of the dwellings. For this thesis, we were granted access to dataset with over 4.9 million rows where each row contains features about the sale process of a dwelling. Some of the features present in the dataset are the price the dwelling was sold for and the date of sale, as well as dwelling-specific features such as the unique matrikkel number. Unfortunately, not all sales are connected to a corresponding Finn-advertisement in the advertisement data set we have described in Section 3.1. As shown in 3.1(2P), the sales without an advertisement code were removed before going forward with the merging process, yielding the final sales dataset with 512,627 observations.

3.3 Condition reports

We have compiled the condition reports in this thesis from three distinct sources and employed different methods to gather and extract the data from each of them. Our data acquisition involved crawling data from Vendu, downloading data through API created by Unbolt AS, and retrieving the documents uploaded to Boligmappa. The data from the two former sources was in JSON format, whereas the latter was in PDF format where we needed to utilize PDF parsers in order to extract the information they contained.

Even if the JSON files containing the data from these sources were built in different formats, there were similarities in terms of what kind of features they stored. Both data sources contained condition scores for different rooms and different aspects of a home. In order to reduce the number of data features included, these checkpoints were merged into 6 main categories: Building framework, roof, bathroom, kitchen, surfaces, and other categories. Table 3.1 shows what checkpoints each category consists of.

Category	**Checkpoints included in the category**
Building framework	Foundation, retaining walls, building plot, walls, windows, doors, balconies, terraces, drainage, water pipes, stairs, fireplace, etc.
Roof	Roofing, drains and fittings, attic
Bathroom	Bathrooms, washrooms, WC
Kitchen	Living room and kitchen, dining room
Surfaces	Inner walls, wallpapers, floors, cabinets, etc.
Other	Ventilation, water lines, electrical installations other installations, other rooms with drain, other moist-exposed room, laundry room

Table 3.1: Contents of the condition reports were merged together into 6 main categories.

Details about the matrikkel information of units, unit size, and the full address were extracted in order to achieve a healthy combination of data from different data sources. The algorithm for combining the datasets consisted of

several steps and manual inspection of samples to determine whether there has been a successful merging process.

3.3.1 Vendu

We have gathered 14142 condition reports, dated between December 2021 and August 2022, from Vendu. The data was collected using a scraper tool developed by a colleague for the purpose of this thesis, and the data was collected using this tool. We have removed 16.85% of the condition reports as these reports belonged to other types of buildings that are left out of the scope of this thesis, leaving us with 6207 condition reports on apartment units and 5552 on houses. Table 3.2 shows the distribution of condition reports across unit types.

Unit type	**Number of units**	**Percentage (%)**
Apartment	**6207**	**43.89**
House	**5552**	**39.26**
Holiday home	1170	8.27
Other	1192	8.43
Commercial buildings	21	0.15
Total	14142	100

Table 3.2: Number of homes of different unit types in Vendu-data. The data that is included in this thesis is marked in bold font.

The dataset includes condition reports from each county in Norway, and the counties with more sales and a greater number of homes are seemed to be more frequently represented in the data. Table 3.3 shows the distribution of condition reports across counties.

County nr	**County**	**Number of units**	**Percentage (%)**
3	Oslo	2109	14.01
11	Rogaland	1092	7.72
15	Møre og Romsdal	607	4.29
18	Nordland	711	5.03
30	Viken	2923	20.67
34	Innlandet	1107	7.83
38	Vestfold og Telemark	1210	8.56
42	Agder	818	5.78
46	Vestland	1565	11.07
50	Trøndelag	1387	9.81
54	Troms og Finnmark	591	4.18
N/A	Unknown	22	0.16
-	All	14142	100

Table 3.3: Distribution of units in the Vendu data set according to the counties of Norway.

Table 3.4 shows the average number of checkpoints per home for each category, as well as the average condition score per home for each category. We also present the total number of units in the Vendu dataset without any checkpoints from different categories.

Category	Average number of checkpoints per home	Average condition score per home	Number of units without checkpoints (%)
Building framework	9.72	1.4298	45 (00.32%)
Roof	1.76	1.5754	5351 (37.84%)
Bathroom	5.83	1.4243	93 (00.66%)
Kitchen	0.97	1.2453	1887 (13.34%)
Basement	0.47	1.8037	8361 (59.12%)
Surfaces	3.86	1.4282	89 (00.63%)
Other	5.86	1.3408	61 (00.43%)

Table 3.4: Statistics deduced from the Vendu data set.

3.3.2 Unbolt AS

The condition reports provided by Unbolt AS were constricted to the Oslo area, and contained 3089 reports in the period from November 2022 to mid-Februrary 2023. The data was downloaded in JSON format through their API. Most of the reports belonged to apartment units, whereas 10.8% of the data were condition reports were on houses, and 1.5% on other building types. This distribution aligns with Statistics Norway's data, which reports that approximately 71% of all dwellings in Oslo are apartment units. Therefore, the skewness in the distribution of different unit types present in the data set reflects the actual distribution to some extent.

Condition reports show some variation between houses and apartments. For instance, conditions reports of apartents do not generally assess the exterior of the building. This is not the case with the houses. The data collected were in the form of a list of nested dictionaries, where each dictionary represents the condition report of a single home. Each dictionary has sub-levels: 1) each "building" on the plot, 2) each floor in a building, 3) each room on a floor, 4) each checkpoint of the room. We first extracted the condition score for each checkpoint, then aggregated the checkpoints into six categories as defined in Table 3.1. Table 3.5 shows the statistics deduced from the Unbolt data set.

While the data from Unbolt has information on homes in Oslo only, the data from Vendu covers a much broader area. Both data sets contained condition scores that belong to other types of buildings, such as commercial buildings or holiday homes. The buildings that are not classified as apartments or houses are not considered in the scope of this thesis, and these were removed at the end of the process of exploring the data sets.

In addition to the condition scores of the checkpoints in the dataset, the matrikkel information, size of the unit, address, and report conduction date were also extracted. For the apartments, the floor number was also deduced from the data. These procedures were performed both on the Vendu dataset and Unbolt dataset. The extracted features are later used for successfully merging the new

Category	Average number of checkpoints per home	Average condition score per home	Number of units without checkpoints (%)
Building framework	6.81	1.4751	18 (%0.58)
Roof	0.35	1.4955	2641 (85.5%)
Bathroom	6.10	1.4970	30 (0.97%)
Kitchen	2.07	1.3920	31 (1.00%)
Basement	0.10	1.5367	2789 (90.3%)
Surfaces	1.14	1.4765	27 (0.87%)
Other	3.42	1.4798	25 (0.81%)

Table 3.5: Statistics deduced from the Unbolt data set

homes with the homes in market transactions data and the advertisement data.

3.3.3 Documents from Boligmappa

The third source of condition reports is Boligmappa (BM). Boligmappa is a company that provides digital solutions to collect and secure documentation about every building and home in Norway. Boligmappa offers a platform that allows homeowners to upload and store documents related to their homes, as well as certified craftworkers to upload documentation about the jobs that they have fulfilled in specific houses and apartments.

We have been granted access to 7.5 millions of documents in PDF from Boligmappa's database. However, the contents of the majority of these documents are unknown, meaning that not all the documents may contain relevant information for this paper. These documents are first needed to be converted into a data format that can be processed with available Python tools. In addition, the majority of the documents were older than the timespan of the advertisements in the Finn-data was removed, which yielded us 819,605 documents, as shown in Figure 3.1(4P). Then, the documents were merged with the datasets containing the advertisements and sold dwellings to achieve the relevant documents. As shown in Figure 3.1(2M), the resulting BM-dataset contained 754,237 observations.

3.3.3.1 Testing different PDF Parsers in Python

The documents from Boligmappa are in PDF format and they must be converted into plain text. There are several libraries that can parse PDF data into text or other formats. In this part, four different libraries were tested on a subset of 998 documents and compared against each other in terms of performance and error handling. The number of files that were successfully opened, read, and extracted data from by the PDF parser is considered a success, while the files that could not be opened by the PDF parser were considered an error.

For the extraction of text from PDF files, we tested 4 open-source parsers. These parsers were tested and the results are represented in Table 3.6. All PDF parsers, except `PyPDF`, performed similarly in terms of error handling, handling 99.8% (996/998) of the documents and extracting their contents. However, a large variation in performance was observed. The two extrema in terms of

processing time were 17 seconds by PyMuPDF, and approximately 20 minutes by pdfminer. Overall, PyMuPDF has performed best and was therefore chosen to be the main PDF parser for the rest of this thesis.

Library	Total time (s)	Time per document (s)	Success	Error
PyMuPDF	17.01	0.017	996	2
PyPDF	289.93	0.291	987	11
pdfplumber	678.85	0.680	996	2
pdfminer	1170.61	1.173	996	2

Table 3.6: Comparison of online PDF parsing libraries in Python

3.3.3.2 Downloading, parsing, and storing files from Boligmappa

Acquiring and saving the documents, as well as parsing, extracting and storing the desired information was done on a personal computer. Storing all 2.1 million PDF files on a computer at the same time would require a large amount of storage. Instead, the documents were downloaded, parsed with PyMuPDF, and then deleted in batches.

The initial experiments, done with a sample of 1500 files, using a batch size of 250 documents per batch yielded the best numbers in terms of effectivity, resulting in downloading, parsing, and deleting all documents in 10 minutes and 33 seconds. This gives a processing time of 0.422 seconds per document. If we had used this algorithm on all $\sim 750,000$ documents, the program would take almost 4 days to finish. Instead, we parallelized the functions by distributing the function's input values across child processes. The functions for downloading and parsing the documents were parallelized, and a grid search was executed to find the optimal batch size along with the optimal number of processors. We decided to parse the documents by iterating through batches of 3000 documents and used 10 processors both for downloading and parsing. This has resulted in a processing time of ~ 0.107 seconds per document, reducing our total estimated process time for $\sim 750,000$ documents to less than a day. The new features extracted from the documents are described in Table 3.7.

Feature	Data type	Acquired from	Mean value	# None
Text	str	PyMuPDF	-	0
Page count	int	PyMuPDF	4.25	0
Word count	int	PyMuPDF	750.29	0
Image count	int	PyMuPDF	15.38	0

Table 3.7: Features extracted from Boligmappa's documents

Based on an initial test, we expected there to be thousands of condition reports in Boligmappa. However, after processing all the documents we found less than 100. We, therefore, chose to not spend time extracting the condition reports from the text, as the added value would be minimal.

3.4 Condition score

The condition score of a home is determined from the accompanying appraisal report for that home. However, these reports do not contain a single conclusive value that describes the condition. Instead, many different aspects of homes were investigated by the appraiser and given a score between 0 and 3. The condition reports were often conducted in different formats, e.g they had different numbers of checkpoints for each room in a unit, and not all units had the same rooms or facilities. The main goal in this thesis is to create a simple regression model for predicting the condition score of a home, hence the number of features describing the condition should be reduced to one.

The most intuitive approach for calculating a single value for the condition score of a home would be computing the average score of all checkpoints in that home and assigning this value as the true condition score of that home. However, from a business perspective, not all rooms in a unit have the same importance for homeowners nor contribute to the housing price equally. In Norway, bathrooms and kitchens are some of the most expensive interior parts of a home to upgrade. Apart from the price of renovation, building framework or roofing of the building an apartment unit lies in may not be considered as the most important aspects when buying a home. Thus, these aspects that are not the owner's responsibility are usually not evaluated in the condition reports. Therefore, we have to take into account the importance of different aspects of a home according to the unit type of that home, and weight the aggregated condition score of each category in a logical way such that the magnitude of these weights corresponds to the importance of the category. Table 3.8 shows the initial weights used for each category when computing the aggregated average condition score. During the computations, the weights are adjusted such that if a unit does not have any information on a category, the weight of this category is distributed evenly across the remaining categories. The resulting aggregated condition score was used as the target variable to be predicted. Figure 3.2 shows the distribution of the aggregated condition scores across unit types.

	Weights	
Category	Houses	Apartments
Building framework	0.16	0.06
Roof	0.16	0.06
Bathroom	0.16	0.35
Kitchen	0.16	0.30
Basement	0.16	0.06
Surfaces	0.19	0.10
Other	0.01	0.01

Table 3.8: The weights of each category in apartments and houses.

Table 3.9 shows the 25^{th}, 50^{th}, 75^{th} percentiles and the average values of the condition score according to the unit type.

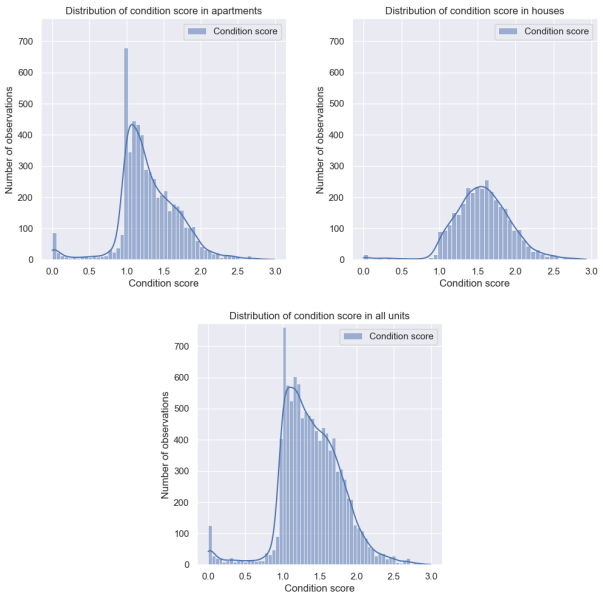

Figure 3.2: Distribution of aggregated, weighted condition scores across unit types.

Unit type	25th	50th	75th	Mean
All units	1.12	1.36	1.65	1.39
Apartments	1.06	1.23	1.54	1.30
Houses	1.34	1.56	1.79	1.56

Table 3.9: 25th, 50th and 75th percentiles as well as the average condition score in all units, apartments and houses.

3.5 Merging datasets

Having 4 different data sources is one of the obstacles of this thesis, as the data sets are of different formats and contain different information to some extent. The Matrikkel information (Norwegian real estate registry number) is one of the tools used in this thesis to identify unique homes. However, the matrikkel information is incomplete or absent for some units in the data sets. For example, many of the units were found to have missing lease numbers (festenr) and unit number "seksjonnr". However, we have other types of information, such as unit size or floor number, which can help identify the units uniquely and mitigate this problem. In order overcome this problem, other features, such as the unit size and the floor number of the unit, were also incorporated into the merging process.

Merging houses together was more straightforward than merging the apartment. Houses can be uniquely identified even without a unit number. However, this is not the case for the apartments, making the merging process a more challenging task. Unfortunately, the majority of the data belonged to apartments,

therefore removing apartments from the final data set was not feasible. The merging algorithm follows the following steps:

1. Prepare data features that were used in merging. This preparation includes:
   - Making sure the common columns in the data have the same data type
   - Editing the columns containing the full address of homes such that the matching address strings can easily be found, i.e removing blank spaces, converting the string to lowercase, replacing special characters.
   - Find the number of days since conducting a condition report on the home and the home is registered as for sale.
2. For all units, identify units with a unit number.
   (a) Perform a left join on units with unit number and Virdi units with unit number using the matrikkel information and the full address.
   (b) Sort the rows by the register date, and keep the row with the newest register date.
   (c) Keep the matched homes in a new data frame, and remove these from the initial datasets.
3. For apartments:
   (a) Identify apartments with a missing unit number, perform the left join on the columns ["kommunenr", "gardsnr", "bruksnr", "adresse", "bruksareal", "etasje"]. Repeat steps 2(a)-(b).
   (b) Then, perform one more left join on the columns ["kommunenr", "gardsnr", "bruksnr", "adresse", "bruksareal"] and repeat steps 2(a)-(b).
4. For houses, perform step 3(b) only.

3.6 Oversampling

Upon analyzing Figure 3.2 where the distribution of the condition score of the observations in the final data set is visualized, we observe that the target variable has a skewed distribution with many observations around the median value of the variable, and very few observations towards the minimum and maximum values of the score. Dealing with a dataset where the target variable has a skewed distribution is not favorable in any kind of ML task. In an attempt to address this limitation of our dataset and overcome the problem of imbalanced data, we applied an oversampling method and acquired a balanced dataset.

An oversampling technique that is commonly used in machine learning is SMOTE (Synthetic Minority Oversampling Technique) (Chawla, Bowyer, Hall, & Kegelmeyer, 2002). This algorithm is based on identifying minority classes and performing resampling by generating synthetic observations from the minority classes with replacements until the desired number of observations from

each minority class is achieved. While this method is commonly used in classification tasks with imbalanced data, it is not directly applicable to regression tasks where the target is a continuous variable. In order to create an oversampled dataset, we first need to define boundaries for the target variable, treating it like a discrete class during the oversampling.

For oversampling purposes, we created an encoded version of the condition score for each observation in the data set, using the encoding function shown in Equation 1, where x is the aggregated condition score computed in Section 3.4, and y is the encoded version of x. The encoded version of the condition score is referred to as the *condition score class* for the rest of this thesis. Then, the data set was oversampled according to the encoded condition score before splitting into training, validation, and test sets. For reproducibility, this split was kept constant across experiments.

$$encoder(x) = y = \begin{cases} 0, & \text{if } x < 0.98 \\ 1, & \text{if } 0.98 \leq x < 1.5 \\ 2, & \text{if } 1.5 \leq x < 2 \\ 3, & \text{otherwise} \end{cases} \tag{1}$$

Table 3.10 shows the distribution of the condition score classes in the training, validation, and test sets acquired from the original data set obtained after the merging process. Table 3.10 shows the distribution of the same variable in the training, validation, and test sets acquired after the data was oversampled.

Encoded score	All data	Training data	Validation data	Test data
0	700	430	141	129
1	5861	3491	1147	1223
2	3266	1980	665	621
3	655	388	144	123
Total	10482	6289	2097	2096

Table 3.10: The distribution of the encoded condition score in the original dataset.

Encoded score	All data	Training data	Validation data	Test data
0	5861	3568	1150	1143
1	5861	3540	1162	1159
2	5861	3485	1173	1203
3	5861	3473	1204	1184
Total	23444	14066	4 689	4689

Table 3.11: The distribution of the encoded condition score in oversampled datasets.

For the rest of this thesis, the data set obtained as a result of the merging process is referred to as the *imbalanced data*, and the data set obtained by the oversampling is referred to as the *oversampled data*.

4 Text-Based prediction of the dwelling condition

In this chapter, we present and explain the models and NLP techniques employed in order to accomplish our objective of estimating the condition score of dwellings based on advertisement text. The models are arranged in ascending order of complexity, ranging from simpler models to more complex ones. First, we start by creating two baseline models, approaching the regression task in a very simple and naive manner without involving any information about the dwellings. To improve the results acquired by the baseline models, we first introduce the methods for extracting features from the advertisement texts, then test the most commonly used ML models for regression problems. Finally, we fine-tune 3 pre-trained large language models for Norwegian.

The data was split into training, test, and validation sets. For reproducibility and comparability the same split was used across all models. The training set, which accounted for 60% of the data, was used to train the models. The validation set, comprising 20% of the data, was utilized for hyperparameter tuning and early stopping. Lastly, the test set, also representing 20% of the data, was reserved for final model evaluation to assess its performance and ability to generalize the results.

4.1 Model evaluation

The metric chosen for the evaluation of the models was the mean absolute error (MAE). The expression for MAE is shown in Equation 2, where y_i is the actual condition score of the i-th dwelling, $\hat{y}_i$ is the predicted condition score of the same dwelling and N is the number of observations.

$$\text{MAE} = \frac{\sum_{i=1}^{N} |y_i - \hat{y}_i|}{N} \tag{2}$$

4.2 Naive baseline models

As a reference point for comparing the forthcoming experiments in this chapter, we first approach the regression task with two baseline models that aim to establish naive, but simple and fast solutions for the task. Subsequently, we seek to improve the results achieved by the baseline models. The baseline models did not utilize any linguistic or numerical features, and were completely based on randomness or the distribution of the target variable.

The first baseline model, $\text{Baseline}_{\text{random}}$, assigns a random value between 0 and 3 to each data point as their predicted value, whereas the second baseline model, $\text{Baseline}_{\text{hard}}$, assigns a pre-defined value to all data points. This pre-defined value was determined by computing the average condition score of the units in the full data set.

4.3 Bag of Words-based approaches

A natural next step to improve the naive baselines is to use a Bag of Words-based approach. Firstly, two different methods for extracting features from text are described. Then, we explain different regression models trained on imbalanced

and oversampled datasets with two different input features: The Bag of Words (BoW) matrix and the Term frequency-inverse document frequency (TF-IDF) matrix extracted from the advertisement texts. We also discuss the advantages and disadvantages of different models, give a concise comparison and explain how the models were tuned and evaluated.

4.3.1 Document representations

In this part, we present two different methods that are commonly used in the NLP field for representing documents as numerical features. We present the Bag of Words (BoW) model, and Term frequency-invert document frequency (TF-IDF), both of which rely on counting the frequency of words in the document.

4.3.1.1 Bag of Words

The Bag of Words model is one of the most commonly used, and simplest methods used for feature extraction from text. The method consists of extracting the vocabulary containing all words from the pre-processed texts in the corpus.

The BoW model yields a $N \times V$ feature matrix, where N is the number of documents and V is the size of the vocabulary extracted from these documents. The value in i-th row and j-th column represents the number of times term j occurs in document i. If all words were taken into consideration for computing this matrix, we would end up with a vocabulary containing 21,086 words, which would result in very sparse matrices as input features to our regression models. Instead, in order to reduce the number of features fed into the regression models, we eliminate some features conditionally. We remove words that have a document frequency strictly greater than a parameter `max`, and strictly less than `min`. With this method, the words that are too common or too rare are eliminated from the feature matrices. For the rest of this thesis, we refer to the parameters (`min`, `max`) as the term filtering threshold. Table 4.1 shows the size of the vocabularies achieved with different combinations of (`min`, `max`). As expected, the number of words included in the vocabulary decreases as the threshold covers a smaller percentage of the documents.

4.3.1.2 Term frequency-inverse document frequency

Term frequency (TF), given in Equation 3, is defined as the relative frequency of term t within document d, where $f_{t,d}$ is the number of times a term t occurs in document d, and the denominator is the total number of terms in document d. Invert document frequency (IDF), given in Equation 4, is the logarithm of the total number of documents N divided by the number of documents containing the term t. Finally, TF-IDF is shown in 5.

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{3}$$

$$idf(t,D) = log\frac{N}{|\{d \in D : t \in d\}|} \tag{4}$$

$$tfidf(t,d,D) = tf(t,d) \cdot idf(t,D). \tag{5}$$

	Split		
	min	max	**Vocabulary size**
Imbalanced	0.01	0.99	832
	0.05	0.95	221
	0.10	0.90	109
	0.15	0.85	69
	0.20	0.80	41
	0.25	0.75	28
Oversampled	0.01	0.99	816
	0.05	0.95	219
	0.10	0.90	106
	0.15	0.85	65
	0.20	0.80	41
	0.25	0.75	27

Table 4.1: Vocabulary size of the input matrices based on different document frequency splits.

Once again, in order to reduce the number of features fed into the regression models, we eliminate some features with the same method used while computing the BoW-matrix in Section 4.3.1.1. Since the same documents were used to compute these two matrices, they contain the same number of words in their vocabularies for the same term filtering threshold.

4.3.2 Regression models

There are four different classes of regression models used in this thesis:

- Simple linear regression model (LinReg)
- Linear regression model with L1 regularization (LASSO)
- Stochastic Gradient Descent regression model (SGD) and
- Gradient boosting regression model (CatBoost)

All four models were trained on imbalanced and oversampled datasets with two different document representations as input features. In this section, we explain the different models and present the tuned parameters for each model.

4.3.2.1 Linear regression

Linear regression is often considered as one of the simplest methods for predicting a continuous variable in ML. A linear regression model assumes that the relationship between the target variable and the input vectors is linear (Hastie, Tibshirani, & Friedman, 2001). It is denoted as shown in Equation 6, where X is the input vector, p is the number of input features, and β_j's and β_0 are unknown coefficients.

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j \tag{6}$$

The linear regression models in this thesis aim to achieve the coefficient vector $\hat{\beta} = (\beta_0, \ldots, \beta_p)$ that minimizes the residual sum of squares (RSS), computed as shown in Equation 7, where N is the number of observations, and y_i and $f(x_i)$ for $i \in \{1, \ldots, N\}$ are the corresponding target and estimated values, respectively.

$$RSS(\beta) = \sum_{i=1}^{N} (y_i - f(x_i))^2 = \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} X_j \beta_j \right)^2 \tag{7}$$

Linear regression analysis is often easy to conduct and interpret. It does not require parameter tuning and is not computationally demanding. However, a linear regression model might lead to two unsatisfactory outcomes. Firstly, ordinary least square estimates often produce results with low bias and high variance, yielding poor prediction accuracy. Secondly, it is difficult to interpret the resulting feature coefficients when dealing with data with a high number of predictors (Tibshirani, 1996).

Table 5.2 and Table 5.3 show the MAE acquired by the linear regression model with BoW and TF-IDF matrices as input features, respectively.

4.3.2.2 Lasso regression

To address the drawbacks of the linear regression model, (Tibshirani, 1996) proposed a shrinkage method, namely Lasso (Least absolute shrinkage and selection operator) regression, that can automatically set some of the feature coefficients to zero, which may help to increase the interpretability of the model while reducing the variance. Lasso regression, similarly to the linear regression, aims to minimize the RSS with the additional Lasso penalty $\sum_1^p |\beta_j|$, where the amount of penalization is controlled by a tuning parameter. The optimal Lasso regression coefficients $\hat{\beta}$ that minimizes the penalized RSS are computed by solving Equation 8,

$$\hat{\beta}^{lasso} = \arg\max_{\beta} \left\{ \frac{1}{2} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} X_j \beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\} \tag{8}$$

where λ is the tuning parameter that controls the amount of shrinkage applied to the coefficient estimates $\hat{\beta}$ (Hastie et al., 2001). Lasso can shrink the coefficients until the coefficients are exactly 0, which provides a natural subset selection and feature elimination which can address the limitations of the normal linear regression (Tibshirani, 1996). The optimal value of the tuning parameter often depends on the dataset and the objectives of the analysis. While larger values of the tuning parameter result in setting most of the features coefficients to zero, providing sparsity and achieving higher interpretability. Conversely, setting smaller values for λ has minimal effect on the model, resulting in fewer coefficients being set to zero.

The tuning parameter λ chosen for this task was 0.01, which is small enough to keep most of the input features while still eliminating the least needed ones, providing us with a natural subset selection. The resulting MAEs by the Lasso regression models are presented in Table 5.2 and Table 5.3.

4.3.2.3 Stochastic Gradient Descent Regressor

Gradient Descent, (GD) (Cauchy, 1847) and Stochastic Gradient Descent, (SGD) (Robbins & Monro, 1951) are iterative optimization methods that are often used for speeding up the process of determining the optimal regression coefficients, β_j's, which minimizes the loss function in a regression task.

The GD algorithm starts with guessing initial values for the β's, and iteratively updating the coefficients in the direction that minimizes the loss. The iterations stop when the gradient of the loss function is close to zero. The magnitude of the updates of the coefficients is regularized by a learning rate. During this process, all observations in the data set are taken into consideration, which might slow down the process when there is a large number of observations to consider.

The SGD is a faster variant of the GD, where we start with randomly determined initial coefficients and a randomly selected subset of observations from the full data set. Then, the coefficients are updated according to the subset. Iteratively, new subsets are selected and the coefficients are updated until the gradient of the loss function is sufficiently close to 0. This procedure is advantageous when solving regression problems with larger datasets, as the subset selection eases the computations needed to find the optimal coefficients. It is often found that the SGD is more accurate than a linear regression model, which motivates us to train the SGD regression model.

We have trained SGD regression models with a learning rate of 0.0001 on BoW and TF-IDF matrices, in combination with six different term filtering thresholds. The results were presented in Table 5.2 and Table 5.3.

4.3.2.4 Gradient boosting

Gradient boosting is a more recently developed machine learning technique that can be used in regression and classification tasks (Friedman, 2002). The gradient boosting regression model is of the stronger regression models in ML, where the combination of weak learners results in a powerful learner. The model is built as an ensemble of many predictive models that are in fact weak learners, which are often decision trees. The training process consists of training the decision trees one by one, where each decision tree corrects the error made by the previous tree. The gradient boosting model has the following additive form;

$$F(\mathbf{x}) = \mathbf{y} = \sum_{m=0}^{M} \beta_m h(\mathbf{x}; \mathbf{a}_m) \tag{9}$$

where M is the total number of iterations, $\mathbf{x}$ are the explanatory variables, $\mathbf{y}$ are the predicted values, the functions $h(\mathbf{x}; \mathbf{a})$ are the weak learners, the β_m's are the coefficients minimizing the previously determined loss function. One starts with an initial guess $F_0(\mathbf{x})$, and continues to update the model coefficients for $m = 1, \ldots, M$ steps (Friedman, 2002).

A gradient boosting model has many hyperparameters that may have significant effects on the model performance. For example, the contribution of each new tree to the prediction of the older trees is regularized by a learning rate. The number of levels allowed in a decision tree is also an important aspect of the model. Gradient boosting trained with shallower trees may improve the interpretability of the model while avoiding overfitting, whereas deeper trees may capture more complex signals from the data.

Compared to simpler regression models, gradient boosting models often yield better prediction accuracy. Another advantage of the gradient boosting model is that it can handle various types of feature variables, including numerical, categorical, and binary variables. Because of their predictive power and flexibility with input data, gradient boosting models are favorable ML models for regression. However, because of the complexity of the models, training gradient boosting models is often time-consuming and the results might be difficult to interpret in contrast to other regression models, for instance, linear regression model. Additionally, there are many hyperparameters that require parameter tuning in order to achieve optimal results with a gradient boosting model.

There are many online tools used for building gradient boosting methods. The tool used in this thesis was `CatBoost` (Dorogush, Ershov, & Gulin, 2018), an open-source Python library. We have trained a CatBoost Regressor with 1000 decision trees of depth 5, and a learning rate of 0.1. We also chose to make use of the built-in early stopping function that comes with this library and made the training procedure halt if the evaluation loss increases in more than 100 iterations in a row. The CatBoost Regressor models with the parameter values mentioned were trained on BoW and TF-IDF matrices, in combination with six different term filtering thresholds. The results were presented in Table 5.2 and Table 5.3. Similar to the linear, Lasso, and SGD regression models, the model on the BoW input features and term filtering threshold of (0.01, 0.99) yielded the lowest MAE among the models trained on the same input features. For both types of input features, CatBoost has given the best performances with (0.01, 0.99) threshold, resulting in MAEs of 0.2429 and 0.2437 for BoW and TF-IDF, respectively.

4.3.3 Grid-search for tuning the gradient boosting model

Since we have observed that the CatBoost regressor was the best-performing model amongst the regressors with BoW-based input features discussed in this thesis, we concluded to conduct a grid-search to find the optimal values of some of the model parameters. The parameters tuned in this part were the learning rate, which corresponds to the rate at which the contribution of each tree is shrunk, the depth indicating how deep the decision trees will be, and the number of iterations which corresponds to the maximum number of trees in the model. We conducted a grid-search testing the following values for these parameters:

- Learning rate $\in \{0.1, 0.05, 0.01, 0.005, 0.001\}$
- Depth $\in \{4, 5, 6, 7, 8\}$
- Iterations $\in \{2000, 3000, 4000\}$

Encoded score	All data	Training data	Validation data	Test data
0	5861	3568	1150	1143
1	5861	3540	1162	1159
2	5861	3485	1173	1203
3	5861	3473	1204	1184
Total	23444	14066	4 689	4689

Table 4.2: The distribution of the encoded condition score in oversampled datasets.

The parameter search was conducted on models trained on both input matrices and the term filtering threshold was kept constant at (0.01, 0.99). Tables 5.5 and 5.6 show the results obtained by the models which were trained on BoW and TF-IDF input matrices, respectively. Unfortunately, parameter tuning had minimal effects on the model performance of the CatBoost regressors. The most optimal model found through the grid search was the CatBoost regressor trained on the BoW input matrix with 4000 decision trees with a depth of 8 and a learning rate of 0.01, which yielded an MAE of 0.2396, only slightly better than the former model with an MAE 0.2429. In addition, the MAE was improved by 0.0033, whereas the time taken to train the parameter-tuned model was approximately 4.5 times slower than the former model. While the results are discussed in detail in the next chapter, we concluded that there is not a need for additional parameter tuning for the CatBoost models.

4.3.3.1 Regression models with oversampled data

The majority of the dataset (55,9%) belonged to class 1, meaning that most of the observations in our data have a condition score that is more than or equal to 0.98 and less than 1.5. Observations from classes 0, 2, and 3 were oversampled from the final dataset in order to achieve the same amount of data that belongs to class 1, resulting in 5861 observations from each class and a total number of 23,444 observations. Table 4.2 shows the distribution of the encoded condition score after oversampling across the training, validation, and test datasets.

After achieving a balanced data set, the BoW and TF-IDF matrices were extracted from the documents in the oversampled data. Table 4.1 shows the sizes of the vocabularies in the matrices computed with different term filtering thresholds. Then, the same regressors were trained on the oversampled dataset with BoW and TF-IDF as input matrices, where the model parameters were chosen to be those which yielded the best model performances for each combination of regressor and input matrix. The results are presented in the tables 5.7 and 5.8, and are discussed further in the next chapter.

4.4 Pre-trained BERT-based approaches

Large language models (LLMs) are currently considered one of the most advanced NLP models available. These models employ state-of-the-art neural network architectures and incorporate sophisticated training techniques, enabling them to achieve exceptional performance across a broad spectrum of NLP tasks.

These models have numerous practical applications and can be adapted to many downstream tasks with ease, including the applications providing chatbots or virtual assistants, language translation, sentiment analysis, and text summarization, among others.

Most of the pre-trained large language models (LLMs) currently available are built by utilizing the transformer architecture. In recent years, there has been some revolutionary achievement in the field of NLP due to the still ongoing developments of transformer-based language models, such as the Generative Pre-trained Transformers (GPT) (Radford et al., 2018) and Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018). Both models have been pre-trained on an extensive size of unlabeled text dataset in an unsupervised manner, then fine-tuned on labeled data from downstream tasks. In this chapter, we will focus on BERT-based models to address our objective.

The BERT model was pre-trained on a corpus containing approximately 3,300 million tokens using two unsupervised tasks, masked language modeling (MLM), and next sentence prediction (NSP). In the MLM task, some of the tokens in the input sentence were randomly masked where the objective of the model was to correctly predict the original token based on the context of the sentence. This gives BERT the ability to understand the relationship between words. In the NSP task, the model's objective is to determine the predicted sentence is the actual consecutive sentence or a random one. With the NSP task, BERT understands sentence relationships and text flow. The fine-tuning of BERT involves initializing a pre-trained BERT model with the initial parameters and adapting it to specific tasks.

The model architecture of BERT consists of several elements. These are model parameters, transformer layers, hidden layers, and attention heads. The model parameters are the weights that can be learned by the model, the transformer layers are transformer blocks, the hidden layers are the layers with mathematical functions applied on model weights, and the attention heads are the size of the transformer blocks in the transformer layers. The $\text{BERT}_{\text{BASE}}$ model consists of 110 million model parameters, 12 layers, 768 hidden layers, and 12 attention heads per transformer block, whereas the $\text{BERT}_{\text{LARGE}}$ model has 340 million parameters, 24 transformer layers, 1024 hidden layers, and 16 attention heads per transformer block.

The original BERT model was pre-trained on an English-only corpus, which restricts its applicability to the corpus in this thesis. Luckily, there are many other multilingual or Norwegian-specific BERT-based models available.

4.4.1 Available Pre-trained BERT-based models for Norwegian

For some time, Google's multilingual BERT model (mBERT) was the only model that can be used for various NLP tasks on Norwegian text. This model was pre-trained on a large corpus that contains text from Wikipedia in around 100 languages as an unsupervised MLM task. At first, 21% of the data were in English[2]. The less frequently represented languages were oversampled before the training procedure, whereas frequently represented languages were undersampled. Even though they have demonstrated that mBERT had a satisfying performance for many languages, it performed somewhat worse for the languages

[2]The distribution of languages in the corpus used to pre-train mBERT is described in this GitHub repository: https://github.com/google-research/bert/blob/master/multilingual.md)

that are frequently represented in their corpus. This indicated that language models specific to a language might perform better than multilingual ones.

Instead of the multilingual models, 3 pre-trained BERT models, that are specifically trained on a Norwegian corpus, were tested in this thesis. These models were NorBERT and NorBERT$_2$ (Kutuzov, Barnes, Velldal, Øvrelid, & Oepen, 2021), models developed by the Language Technology Group (LTG) at the University of Oslo, and NB-BERT$_{\text{BASE}}$ (Kummervold, De la Rosa, Wetjen, & Brygfjeld, 2021), developed by the National Library of Norway. All three models were trained by feeding them raw Norwegian text, meaning that pre-processing of text was not necessary for these models.

NorBERT is one of the first monolingual LLMs for Norwegian and was released by the LTG in 2021. The model was pre-trained on a corpus that consisted of around 2 billion word tokens and 203 million Norwegian sentences gathered from the Norwegian Newspaper Corpus and Norwegian Wikipedia. A second version, NorBERT$_2$, was released in February 2022. It was trained from scratch on a completely new, larger corpus that contained around 15 billion word tokens and 1 billion sentences. The NB-BERT$_{\text{BASE}}$ model by the National Library of Norway was pre-trained on a corpus built from text from many different sources. Some of these resources, among many others, were the Norwegian Wikipedia, Norwegian books, and government reports. All three models have shown that monolingual models tend to perform better than multilingual models for many downstream tasks. Therefore, we have decided to not include multilingual models in this thesis.

To the best of our knowledge, there is no academic research that has been conducted using pre-trained BERT-based models for regression analysis. This gap in the literature highlights an opportunity for further investigation and experimentation, particularly given the potential benefits of the BERT-based models' powerful natural language processing capabilities for other downstream tasks. In the following sections, we will fine-tune the models by incorporating a regression layer on top of the pre-trained layers. Furthermore, we will investigate the effect of combining different model parameters, such as the learning rate and batch size, and examine the adaptability of BERT-based models to a regression task.

4.4.2 Finding the optimal parameters

Some aspects of the model, like the model architecture and the early stopping criteria, and some training parameters, such as the loss function and the optimizer, were kept constant across all models in order to simplify comparing the results. Model architecture and the early stopping criteria are described later in this section. If the early stopping criteria is not met, the model was allowed to be trained for 200 epochs at most. The loss criterion chosen for this task was mean squared error loss (MSE), one of the most commonly used loss functions in regression tasks. The computation of the MSE loss is given in Equation 10, where N is the batch size, $\mathbf{y}$ denotes the target and $\hat{\mathbf{y}}$ denotes the predictions in each batch. AdamW optimizer (Loshchilov & Hutter, 2019) was chosen as the main optimizer. Keeping the loss function and the optimizer constant, a range of learning rates and batch sizes were examined for each model. We started first by testing 5 different batch sizes for finding the optimal one that yields the minimum mean absolute error for each model while keeping the learning rate

constant at 0.001. After determining the best batch size for each model, we kept this parameter constant and ran the models with different learning rates.

$$L = mean(\{l_1, \ldots, l_N\}^\top), \quad l_n = (y_n - \hat{y}_n)^2 \tag{10}$$

Model architecture

The pre-trained BERT-based models that were fine-tuned in this thesis were utilized both in their frozen and unfrozen states. The BERT-based models consist of a single linear regression layer on top of the pre-trained BERT model. First, the inputs from the test set is fed into the BERT model in batches of size B to obtain contextualized representations. Then, the output from BERT was fed into a linear layer, where a linear transformation was applied to the contextualized representations in order to achieve a $(B \times 1)$ vector which represents the predictions for the batch. Then, these predictions were sewed together to obtain the predictions for the full test set.

In a frozen model, the model parameters are kept constant, and the contextualized word embeddings obtained by the pre-training process are directly fed into the regression layer, where a simple linear regression model is trained. In the unfrozen model, however, the model parameters in all layers are updated during the fine-tuning, and the updated version of the word embeddings is fed into the regression layer. The most important part of this method is that the model is allowed to learn more context from the input text (e.g. the advertisement texts advertisements) instead of predicting values based on the final version of the pre-trained model, and the model parameters are adjusted specifically for our task. Fine-tuning an unfrozen model is a more computationally expensive method, and it is easier for an unfrozen model to start overfitting as the model might start memorizing the characteristics of the training set when the model parameters are updated according to this set.

Early stopping criteria

When training machine learning models, we usually aim to create a model that performs well on new, unseen data. That is, the model is expected to have the ability to generalize well on unseen data. The opposite of this phenomenon is over-fitting, where the model starts to memorize the properties of the data we use during training instead of learning new information from it. Large language models with a very high number of parameters like BERT are especially prone to overfitting when the training data is not sufficiently large (Lee, Lee, & Kang, 2021). Early stopping is one of the methods that can be used to prevent this undesirable behavior of the ML model. While there is no universally optimal early-stopping method for all ML models, there are several approaches in the literature that have been demonstrated to work well.

One may choose to halt the training process as soon as the validation error in the current epoch is higher than the validation error in the previous epoch. This method may result in stopping the training too early, as we may have reached a local minimum instead of the global minimum. Instead, (Prechelt, 2002) suggested different early-stopping techniques for training multi-layer perceptrons: 1) Stopping the training as soon as the validation loss exceeds some pre-set value, 2) stopping if the validation error increases in s successive epochs.

Figure 4.1: Mean training and validation errors for each epoch during training of NorBERT$_2$ with and without early stopping. The learning rate in these experiments was 0.001 and the batch size chosen was 256. The test error yielded with and without early stopping were 0.2764 and 0.2737 respectively.

Then, these techniques are evaluated in terms of training time, efficiency, effectiveness, robustness, and the trade-off between test error and training time across 12 different problems and 24 different model architectures. The third method has been demonstrated to have the best trade-off between test error and the time taken to train the models. Prechelt has also concluded that models with slower early stopping criteria, e.g. a criterion that makes it harder for the model to stop the training process, tend to gain more generalization. Based on the findings of this paper, a method similar to Prechelt's third suggestion was adapted to the training process in this thesis.

For the experiments in this thesis, an early stopping function was implemented in order to avoid over-fitting and accelerate the experimenting process. Across all models, the fine-tuning is stopped if the validation error does not improve for more than s, a pre-set number, epochs in a row. That is, if a new lowest error is not found for a defined number of epochs, the fine-tuning will halt. The pre-set value chosen in this thesis was 3. In order to determine whether the adoption of $s = 3$ demonstrates favorable performance in terms of training time and testing error, NorBERT$_2$ was fine-tuned with and without early stopping. The batch size chosen for this experiment was 256, whereas the learning rate was set to 0.001.

Table 4.3 shows the number of epochs and time taken to train these two models, as well as the minimum validation error and the testing error acquired. NorBERT$_2$ without early stopping was ~ 6.4 times slower than NorBERT$_2$ with early stopping, only to perform 0.9768% better in terms of testing error. The validation and training errors through epochs for both models are plotted in

	# Epochs	Min. val. error	Test error	Time (s)
$\text{NorBERT}_{2,\text{ early}}$	**34**	0.2928	0.2764	**7387**
$\text{NorBERT}_{2,\text{ 200}}$	200	**0.2885**	**0.2737**	47429

Table 4.3: Comparison of NorBERT_2 with and without early stopping. The number of epochs for the training, minimum validation error acquired during training, the MAE on the test set and the total training time for each model is presented. The best result for each metric is written in bold font.

Figure 4.1. It is observed that the validation error stops improving around the 30th epoch and converges after this point. These results indicate that with the chosen early stopping criteria and $s = 3$, we achieve an optimal trade-off between the time taken to train a model and the test error, using much less time to train the model while preserving the model performance. For the rest of this thesis, the early stopping criterion has been kept constant, with $s = 3$.

Batch size and maximum sentence length

The batch size (`batch_size`) and maximum sentence length (`max_len`) allowed while fine-tuning BERT-models are parameters that can impact both the speed of the fine-tuning process, and the performance of the models. Intuitively, the greater the batch size and maximum sentence length are, the more representative each batch is of the training data. Increasing the maximum sentence length allows us to keep the sentences fed into the models as long as possible, keeping much of the information contained preserved. With large batch sizes, the models are expected to be fine-tuned much quicker. However, combining large batch sizes with large maximum sentence lengths would also require more powerful memory resources. Therefore, we chose to test these two hyperparameters simultaneously and aim to strike a balanced trade-off between model performances and resources needed to fine-tune.

In order to determine the optimal pairing of batch size and the maximum number of tokens for each pre-trained model, we have trained the models with various values for the hyperparameters in question, while keeping the learning rate and the early stopping criterion constant. The learning rate chosen for these experiments were 0.001, as this is the default learning rate of the `AdamW` optimizer. The early stopping criterion was set to halt the fine-tuning process if the validation error worsens for more than 3 epochs in a row. The batch sizes tested were 64, 128, 256, 320, and 512, and the maximum sentence lengths were {128, 256, 512}. All experiments were conducted on the Nvidia T4, with the exception of combining batch size of 320 together with maximum sentence length of 256. For this case, we needed to increase the available memory resources, switched to the Nvidia A10G. The only experiment we decided to not go forward with was combining a batch size of 512 with maximum sentence length of 512 due to memory limitations.

The results are shown in Table 5.10. While we were not able to fine-tune models with a batch size of 512 and maximum sentence length of 512 with the Nvidia T4 or A10G, we did also observe that while fine-tuning with a maximum sentence length of 512, the batch size has little to no effect on the model per-

formances. Therefore, we decided that upgrading to a bigger GPU to test this combination of hyperparameters is not needed in this case.

Based on this mini-experiment, the (`batch_size`, `max_len`) combinations that achieved the best performance for NorBERT, NorBERT$_2$, and NB-BERT$_{\text{BASE}}$ were (128, 512), (64, 128) and (256, 215), respectively, yielding MAE's of 0.2858, 0.2737 and 0.2689. The optimal values for these hyperparameters determined in this experiment were also reused in the next part, where we determine the optimal learning rate for fine-tuning the pre-trained models.

Learning rate

After determining the optimal batch size for the pre-trained models, a range of learning rates was evaluated through experimentation. Learning rates chosen for this task were $\{0.0001, 0.0005, 0.001, 0.005, 0.01\}$. Batch sizes were set to be the optimal ones acquired from earlier experiments, while the loss function, optimizer and the early stopping criterion were kept constant. The results obtained by the models are presented in Table 5.11.

4.4.3 Improving the model performance: Oversampling and unfreezing

While parameter optimization has a notable impact on the performance of the models, the effect is not significant as the resulting performance gains are often modest. Furthermore, the MAE's from each model are quite close in terms of magnitude. Overall, the BERT models performed slightly better than the Baseline$_{\text{hard}}$ model, and was outperformed by the best-performing linear regression model. This suggests that, while parameter search can play a role in optimizing the model performance, we must consider different approaches in order to improve the models.

The presence of highly imbalanced data is an additional concern that has been acknowledged but not yet resolved. Our target variable, the home standard takes values between 0 and 3. However, as discussed in the previous chapters and shown in, 3.2, the distribution of this variable is very skewed. Specifically, most of the observations of the target variable lie between 1 to 2, with limited observations falling outside of this range. Figure 5.5 shows the distribution of the predicted condition score within the test set only for our best-resulting frozen models, together with the actual values. We observe that even the most optimal models we have fine-tuned struggle to predict values outside a certain range. To address this problem, we first fine-tune the models on the oversampled data with the previously determined optimal hyperparameters. Then, we unfreeze the models and fine-tune them on imbalanced and oversampled data.

Oversampling

The oversampling technique employed on the dataset for training regressors on balanced data was utilized once more to fine-tune the BERT-based models using oversampled data. The continous condition scores in the dataset were encoded to discrete classes according to the encoding function given in Equation 1, and the minority classes were resampled with replacement until a sufficient amout of data from the minority classes were achieved.

For this task, we have fine-tuned our three frozen models on the balanced dataset using the optimal hyperparameters determined in the previous section. We have fine-tuned NorBERT$_2$ (batch size 128, maximum length 512, learning rate 0.0005), NorBERT$_2$ (batch size 64, maximum length 128, learning rate 0.001), and NB-BERT$_{\text{BASE}}$ (batch size 256, maximum length 512, learning rate 0.005) on the oversampled data set. The results are presented in Table 5.13.

Unfreezing the models

Until now, all models were fine-tuned in their frozen state, where the learnable model parameters were fixed and not updated during fine-tuning. Utilizing frozen models with constant parameters limits the extent to which a model can learn. The results obtained align with this statement, as we observed that the frozen pre-trained models have only demonstrated slightly improved results than our baseline models. We have also observed that imbalanced data was not the cause of poor outcomes of the frozen models, as the oversampled frozen models performed worse than the models fine-tuned on imbalanced data. Surprisingly, despite the complexity of BERT, the frozen models did not seem to capture any additional signals in the data set that is different than the signals captured by simpler linear regression models.

We fine-tuned the BERT-based models in their unfrozen states and allowed the models to adjust their learnable model parameters according to the training set. The models were fine-tuned with two different (`batch_size`, `max_len`) combinations, as well as various values of the learning rate. The models are fine-tuned with the imbalanced and oversampled data sets, and the results are presented in Tables 5.14, 5.15, 5.17 and 5.18.

5 Results and analysis

The results acquired from our models in Chapter 4 are presented in this chapter. We first discuss the results from our baseline models as well as the differences between those. Then, we present the mean absolute errors obtained by 4 different regression models, trained on two different feature representations and different term filtering thresholds. Later, we discuss how these models behave under parameter tuning and with oversampled data sets. We explore the predicting abilities of our models, both in terms of their predictive power on the full data set and on dwellings with a condition score from specific ranges. We investigate whether some models are better at learning signals from advertisement texts of dwellings with a condition score that can be considered as an outlier, or whether some models perform best with dwellings with condition scores that are approximately equal to the average. We continue with similar methods to analyze the results acquired by three pre-trained BERT models for Norwegian. Finally, we compare the BoW-based approaches and BERT-based approaches.

5.1 Naive baseline models

We have created two baseline models as reference points for the rest of the models. These models were the $\text{Baseline}_{\text{random}}$ which assigns a randomized value between 0 and 3 to the observations in the test set as their predicted condition score, and $\text{Baseline}_{\text{hard}}$ which computes the average condition score of all units in the dataset and assigns this value to all observations in the test set. The MAE acquired by $\text{Baseline}_{\text{random}}$ and $\text{Baseline}_{\text{hard}}$ were 0.8308 and 0.3161, respectively and the results are shown in Table 5.1.

Model	**MAE**
$\text{Baseline}_{\text{random}}$	0.8308
$\text{Baseline}_{\text{hard}}$	0.3161

Table 5.1: MAEs acquired by the baseline models. We observe that an assisted baseline performs much better than the baseline that randomly guesses the condition score.

Randomly guessing the score in the $\text{Baseline}_{\text{random}}$ model yielded the worst result among the baseline models. With this method, we picked a random value between 0 and 3 while predicting a rather imbalanced target which mostly lies between 1.12 and 1.65, the 25^{th} and 75^{th} percentiles of the target variable in the full dataset. However, due to the randomness in this model, the probability of the model predicting a condition score outside of the range $[1.12, 1.65]$ is the same as the probability of predicting a score that lies in that range. For this reason, the model might predict values outside of the range more often than the actual frequency of the occurrence of the outlier values. This model might have performed better on a balanced dataset where the distribution of the target variable is not skewed and indicates that the skewness of the target variable is an important characteristic of the data.

Assigning the average condition score of the full dataset as the predicted score for the observations in the test set in the $\text{Baseline}_{\text{hard}}$ model resulted in

a much lower MAE than the first baseline. The condition score assigned to the observations in the test set was 1.39, and the MAE achieved with this model was 0.3161, which has created a better reference point for the forthcoming models in this thesis. Due to the skewness of the distribution of the target variable and the values of the variable being concentrated around the mean value, the probability of an observation having a condition score close to the average is greater than it having a score further from the average. Hence, this baseline model is evidently good at predicting most of the dwellings in this set, which aligns with the model's performance being better than the first baseline model.

5.2 Regression models

We have trained four different regression models on the imbalanced and over-sampled dataset, where the input features were the BoW and TF-IDF matrices computed with 6 distinct term filtering threshold. The regression models trained were a simple linear regression model, a Lasso regression model, a Stochastic Gradient Descent (SGD) regression model and a gradient boosting regression model from the `CatBoost` online Python library. The tuning parameter, α, in the Lasso regression models was set to 0.01 across all models. The SGD regression models were trained with a learning rate of 0.0001, and the initial gradient boosting models were trained with a tree depth of 5, maximum number of decision trees of 1000 and learning rate of 0.1.

In this section, we outline the models' performances in terms of MAE, present the distribution graphs of the actual and predicted values of the target variable for the most optimal version of each regression model, and explore their predictive power for dwellings with a condition score that is a member of 4 discrete classes that were computed with the encoding function denoted in Equation 1. The latter was done by computing the Pearson correlation coefficient between the actual values and the predicted values of the target variable for all observations in the test set. Additionally, we divide the actual values into 4 discrete classes using the encoding function given in Equation 1, and present the correlation coefficients between the predicted and actual values of the target variable for dwellings with condition scores from different condition score classes.

5.2.1 Regression on imbalanced data

When trained on the imbalanced data set, all four regression models had similar performances in terms of the prediction error and the correlation between the predicted and actual values of all dwellings in the test set.

Table 5.2 and Table 5.3 show the MAE acquired by the models trained on the imbalanced data set with the BoW and TF-IDF input matrices, respectively. Additionally, the tables show the impact of the term filtering threshold on the performance regression models. In general, choosing a less strict term filtering threshold has resulted in higher model performance and lower MAE across all regression models. The inclusion of a higher number of words in the computation of the input matrices might have protected most of the information contained in the text, which allows the models to have more to learn from the input features. This is the case for most of the regression models, with the exclusion of Lasso which has performed better with a term filtering threshold of (0.2, 0.8) on the TF-IDF matrix.

	Term filtering threshold (min, max)					
Models	(0.01, 0.99)	(0.05, 0.95)	(0.1, 0.9)	(0.15, 0.85)	(0.2, 0.8)	(0.25, 0.75)
LinReg	**0.2553**	0.2597	0.2661	0.2732	0.2802	0.2943
LASSO	**0.2703**	0.2730	0.2761	0.2811	0.2845	0.2971
SGD	**0.2500**	0.2599	0.2657	0.2725	0.2796	0.2931
CatBoost	**0.2429**	0.2578	0.2606	0.2699	0.2780	0.2929

Table 5.2: MAE acquired by different regression models trained on imbalanced dataset with Bag of Words as input features. The best result for each model is written in bold font.

	Term filtering threshold (min, max)					
Models	(0.01, 0.99)	(0.05, 0.95)	(0.1, 0.9)	(0.15, 0.85)	(0.2, 0.8)	(0.25, 0.75)
LinReg	**0.2527**	0.2551	0.2632	0.2703	0.2777	0.2926
LASSO	0.3076	0.3076	0.3045	0.3017	**0.2999**	0.3056
SGD	**0.2699**	0.2708	0.2749	0.2807	0.2854	0.2969
CatBoost	**0.2437**	0.2538	0.2665	0.2740	0.2878	0.2975

Table 5.3: MAE acquired by different regression models trained on imbalanced dataset with TF-IDF as input features. The best result for each model is written in bold font.

We have also observed that the models have behaved differently based on the input matrix fed into them. For the simple linear regression and Lasso models, feeding the model with the BoW input matrix has yielded better results, whereas the SGD regression model had better performance with the TF-IDF matrix regardless of the term filtering threshold chosen. The `CatBoost` regression model achieved lower MAE's with the TF-IDF input matrix for most of the term filtering thresholds, except when using the least strict term filtering threshold, namely (0.01, 0.99).

Overall, the best-performing models trained on imbalanced datasets with two different input matrices were the same. `CatBoost` regression model has achieved an MAE of 0.2437 with the term filtering threshold of (0.01, 0.99) when trained on the TF-IDF input matrix, and an MAE of 0.2429 when trained on the BoW input matrix with the same threshold. The latter was the best-performing among the models trained on the imbalanced data set.

In order to examine the predictive power of the models for the test set, we have visualized the distributions of the predictions of the models against the distribution of the actual target values. The visualization of the distributions was carried out using Kernel density estimation (KDE) plots. Figures 5.1 and 5.2 show the distribution of the actual values and predictions of the best-performing model from each regression model type, where the features fed into the models are the BoW input matrix and TF-IDF matrix, respectively. We observe that all four models trained on both input feature matrices struggled to estimate the condition of dwellings with very high or very low scores. Most predictions lay around 1.3-1.4. Considering that the median condition score for the full data set was 1.36, most of the models appear to have predicted values around the

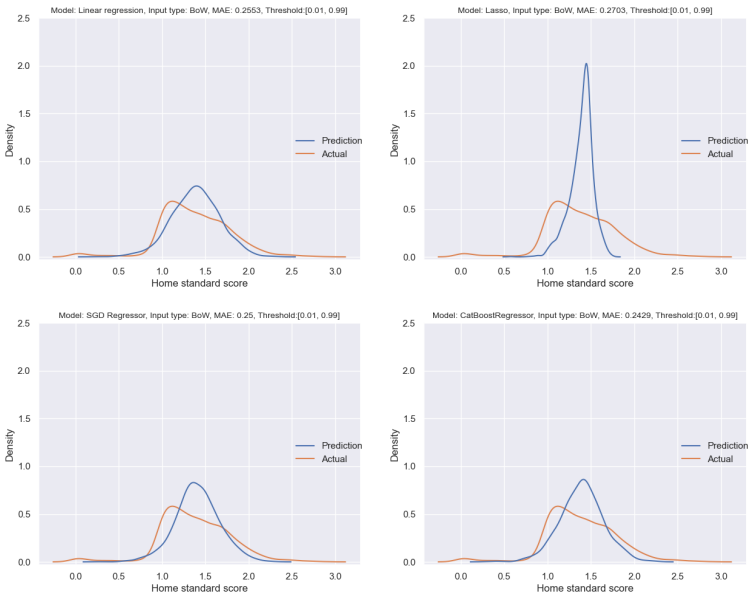

Figure 5.1: Kernel density estimation plots for the distribution of the actual target values versus the predictions made by the regression models trained on the imbalanced dataset using the BoW input matrix.

median. With the exception of the Lasso regression model, most of the models were able to have somewhat small and large predictions. The Lasso model, however, had the worst performance across all linear regression models, both in terms of MAE and also in terms of the variation between its predictions. Due to very little variation in their predictions, the regression models have performed only slightly better than the $\text{Baseline}_{\text{hard}}$ model, which yielded an MAE of 0.3161, while the best-performing regression model (`CatBoost` regression model trained on BoW-matrix with the least strict term filtering threshold) performed only slightly better than this baseline, indicating a performance improvement of 0.073.

For further examination of the predictive power of the models, we conduct a correlation analysis and investigate the relationship between the predictions made by models and the actual values of the target variable. Table 5.4 shows the Pearson correlation coefficients between the predicted and actual values of the target variable for the imbalanced models. In addition to the correlation coefficients between actual and predicted values of the target variable for all observations in the test set, as well as the correlation between the actual values from different condition score classes. The correlation coefficients are computed from the predictions of the best-performing models for each combination of regression model type and input matrix type.

We observe that the correlation coefficients acquired for the observations in the full test set across the models are very similar in terms of the magnitude and direction of the correlation. The gradient boosting model had the highest correlation coefficient between the values it predicted and the actual values of the target variable for both input matrix types, with correlation coefficients

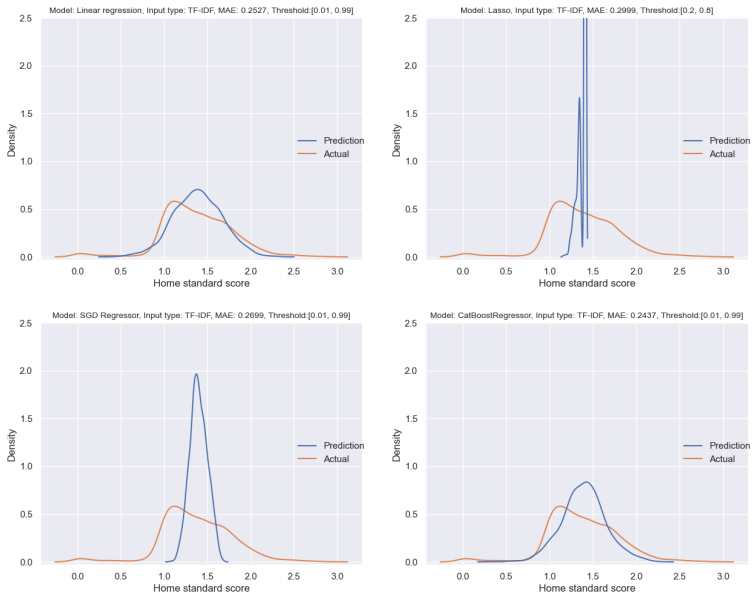

Figure 5.2: Kernel density estimation plots for the distribution of the actual target values versus the predictions made by the regression models trained on the imbalanced dataset using the TF-IDF input matrix.

of 0.5937 and 0.5840 for the BoW matrix and TF-IDF matrix, respectively. The Lasso regression model, on the other hand, had the smallest correlation coefficient among all models, indicating that the predictions made by the Lasso model are less related to the actual values than the rest of the models. This finding aligns with the MAEs obtained with the models, where the gradient boosting model yielded the smallest error and Lasso had the greatest.

The correlation coefficients between the actual values that belong to the four discrete classes and their predictions separately, however, are much lower than the correlation coefficients between the actual and predicted values for the full test set. All models, regardless of the input matrix, had the highest correlation between the predicted and actual values of observations with a condition score that belongs to the condition score class 1, i.e. the condition score lies between 0.98 and 1.5. Considering that the models mostly predict around the median condition score of the total data set, and the median value lies in the interval defining the condition score class, this finding is only logical and consistent.

All models, regardless of the input matrix, had the least correlation coefficient for the observations with a condition score from class 3. This indicates, once again, that the models struggled with predicting the condition score of the badly maintained homes based on their advertisement text. Two extremes in predicting the condition score of these homes were the Lasso regression model (trained on the BoW matrix), and the SGD regression model (trained on the TF-IDF matrix), both of which yielded a negative correlation coefficient for this condition score class. The negative correlation coefficient means that the predictions and actual values are negatively proportional to each other, and these two models performed extremely poor on these dwellings.

Overall, the magnitude of the correlation coefficients is considered to indicate a moderate to weak correlation between the predictions and the actual values and suggests that the predictive power of the linear regression models is insufficient for this task.

	Input type									
	BoW					TF-IDF				
CS Class	0	1	2	3	All	0	1	2	3	All
LinReg	0.2816	0.3593	0.2192	0.1324	0.5643	0.2730	0.3687	0.2250	0.1132	0.5688
LASSO	0.1166	0.3367	0.1415	-0.0602	0.4535	0.1056	0.1883	0.0797	0.0153	0.2772
SGD	0.2879	0.3723	0.2319	0.1530	0.5706	0.0934	0.4075	0.1449	-0.0181	0.5157
CatBoost	0.2552	0.3784	0.2651	0.1867	0.5937	0.2799	0.3682	0.2589	0.1056	0.5840

Table 5.4: Correlation coefficients indicating the correlation beween the actual and predicted values of the target variable. The predictions were acquired by the regression models trained on the BoW and TF-IDF input matrices extracted from the imbalanced data set.

5.2.2 Grid-search for the gradient boosting model

Since we have acquired the lowest MAEs with the `CatBoost` model, it was decided to perform a grid search on this model to investigate the impact of the model parameters on the MAE obtained by the model and to determine whether a better-performing gradient boosting model can be trained on the imbalanced data set. The hyperparameters considered in this grid-search were the learning rate of the model with values of 0.1, 0.05, 0.01, 0.005, 0.001, the depth of the decision trees with values of 4, 5, 6, 7, and 8, and the number of trees in the model with the values of 2000, 3000, and 4000. The grid-search was conducted on models utilizing the BoW input matrix as well as the TF-IDF input matrix, and the term filtering threshold was kept constant. Overall, 150 different `CatBoost` regression models were trained. The results of the grid-search experiment are presented in Table 5.5 and 5.6.

We have observed for larger learning rates, the impact of the depth and number of iterations were minimal. For learning rates of 0.1 and 0.05, the worst-performing gradient-boosting model was the one trained on TF-IDF input matrix with parameter values 4, 4000 and 0.1 for the depth, number of iterations, and learning rate, whereas the best-performing model was the one trained on the BoW input matrix with parameter values 8, 2000 and 0.05. The difference between the MAEs obtained by these models were only 0.0087, indicating that the parameter selection has very little effect of the performance when the learning rate is large. For learning rates that are smaller than 0.01, it was observed that the models performed worse as the learning rate decreased. With some exceptions, the learning rate of 0.01 was observed to be the optimal learning rate for most of the models.

For the models trained with the same (iterations, learning rate) combinations, the depth of the model was found to have a positive impact on the model. In general, the models where building a deeper tree is possible had better model performances. For the models that were trained with a learning rate of 0.01, the

		Learning rate				
Depth	**Iterations**	0.1	0.05	0.01	0.005	0.001
4	2000	0.2463	**0.2438**	0.2460	0.2489	0.2678
	3000	0.2487	0.2444	0.2446	0.2468	0.2614
	4000	0.2499	0.2455	0.2438	0.2458	0.2575
5	2000	0.2453	**0.2422**	0.2449	0.2475	0.2656
	3000	0.2470	0.2431	0.2434	0.2457	0.2594
	4000	0.2483	0.2443	0.2426	0.2448	0.2555
6	2000	0.2443	0.2423	0.2440	0.2467	0.2642
	3000	0.2456	0.2433	0.2425	0.2449	0.2579
	4000	0.2460	0.2436	**0.2418**	0.2542	0.2439
7	2000	0.2438	0.2415	0.2431	0.2459	0.2631
	3000	0.2444	0.2419	0.2411	0.2441	0.2569
	4000	0.2450	0.2426	**0.2402**	0.2428	0.2533
8	2000	0.2445	0.2408	0.2417	0.2448	0.2623
	3000	0.2446	0.2407	0.2402	0.2429	0.2561
	4000	0.2450	0.2411	**0.2396**	0.2416	0.2524

Table 5.5: Results of grid-search experiment conducted on the CatBoost regression model where the model was fed with BoW input matrix.

		Learning rate				
Depth	**Iterations**	0.1	0.05	0.01	0.005	0.001
4	2000	0.2474	0.2442	0.2452	0.2485	0.2680
	3000	0.2487	0.2455	0.2444	0.2462	0.2615
	4000	0.2504	0.2465	**0.2440**	0.2450	0.2576
5	2000	0.2459	0.2427	0.2442	0.2469	0.2659
	3000	0.2475	0.2440	0.2432	0.2448	0.2594
	4000	0.2485	0.2455	**0.2428**	0.2437	0.2556
6	2000	0.2458	0.2431	0.2435	0.2460	0.2460
	3000	0.2476	0.2446	0.2422	0.2442	0.2580
	4000	0.2485	0.2457	**0.2417**	0.2430	0.2543
7	2000	0.2474	0.2419	0.2426	0.2450	0.2632
	3000	0.2485	0.2435	0.2413	0.2434	0.2569
	4000	0.2492	0.2445	**0.2410**	0.2423	0.2532
8	2000	0.2447	0.2417	0.2420	0.2446	0.2624
	3000	0.2451	0.2429	0.2409	0.2427	0.2561
	4000	0.2451	0.2434	**0.2401**	0.2416	0.2523

Table 5.6: Results of grid-search experiment conducted on the CatBoost regression model where the model was fed with TF-IDF input matrix.

most optimal results were always obtained by the models with the deepest trees. While the decrease in terms of MAE is still very small, it can be concluded that the error made by the model is negatively proportional to the depth of the trees in the model.

For most of the models trained with the same (depth, learning rate) combinations, the performance of the models was often proportional to the total number of decision trees present in the model. This indicates that as the number of weak learners in a model increases, a performance gain was achieved by the ensemble model built on the weak learners.

The best-performing gradient boosting models in the parameter search experiment for the different input feature matrices had the same model parameters. For both input matrices, we achieved the smallest MAE when the parameters were set to 8, 4000, and 0.01 for the tree depth, number of trees, and learning rate. With these parameters, the MAEs acquired were 0.2396 and 0.2401 for the models trained on the BoW input matrix and the TF-IDF input matrix, respectively. The former was the best-performing model throughout the grid-search experiment we conducted. It is noteworthy that the gradient boosting models trained on the imbalanced dataset with the BoW input matrix have most of the time better performance than the models trained with the TF-IDF input matrix, regardless of the model parameters.

The grid-search experiment conducted has shown that the performance of the gradient boosting regression models is directly proportional to the tree depth and the number of trees present in the model. However, the relationship between the model performance and the learning rate is somewhat unclear. We have observed that the model performance worsens for any learning rate that is larger and smaller than 0.01, and this finding is consistent for most of the models. We have demonstrated that while parameter tuning is an important part of the gradient boosting model which is a highly parameter-sensitive model, the impacts of the parameter choices have been minimal for our case.

5.2.3 Regression on balanced data

Table 5.7 and Table 5.8 show the MAE acquired by the models trained on the oversampled data set with the BoW and TF-IDF input matrices, respectively. The effect of the term filtering threshold was similar to the results achieved by the models trained on the imbalanced data set, where the least strict threshold yielded the best results in general. The overall observation in terms of the model performances was that most of the regression models have responded negatively to the oversampling of the dataset, and performed worse when the dataset was balanced. Only the `CatBoost` regression model has achieved better results when trained on the oversampled data set, improving the MAEs from 0.2429 to 0.2041 for the best-resulting BoW model, and from 0.2437 to 0.1835 for the best-resulting TF-IDF model. The latter was the best-performing among the models trained on the balanced data set.

	Split (min, max)					
Models	(0.01, 0.99)	(0.05, 0.95)	(0.1, 0.9)	(0.15, 0.85)	(0.2, 0.8)	(0.25, 0.75)
LinReg	0.3225	0.3664	0.4118	0.4361	0.4538	0.4860
LASSO	0.3850	0.3948	0.4247	0.4415	0.4572	0.4884
SGD	0.3208	0.3663	0.4126	0.4385	0.4560	0.4870
CatBoost	**0.2041**	0.2161	0.2458	0.2684	0.2886	0.3261

Table 5.7: MAE acquired by different regression models trained on the oversampled dataset with Bag of Words as input features for different term filtering thresholds. The only significant result was acquired by the CatBoost regression model with the threshold (0.01, 0.99)

	Split (min, max)					
Models	(0.01, 0.99)	(0.05, 0.95)	(0.1, 0.9)	(0.15, 0.85)	(0.2, 0.8)	(0.25, 0.75)
LinReg	0.3166	0.3586	0.4045	0.4297	0.4455	0.4790
LASSO	0.5082	0.4925	0.4745	0.4702	0.4743	0.5023
SGD	0.3433	0.3690	0.4081	0.4313	0.4478	0.4799
CatBoost	**0.1835**	0.1871	0.1991	0.2093	0.2315	0.2535

Table 5.8: MAE acquired by different regression models trained on oversampled dataset with TF-IDF as input features for different term filtering thresholds. The only significant result was acquired by the CatBoost regression model with the threshold (0.01, 0.99).

The visualization of the distribution of predictions made by the models trained on the oversampled dataset versus the distribution of actual values is presented in Figures 5.3 and 5.4 using the Kernel density estimation method. Compared to the previous figures presenting the distribution of the predictions of the models trained on the imbalanced data set, the distributions of the predictions acquired by the models trained on the oversampled data appear to fit the distribution of the actual values much better in this case. We also observe that for most of the models, with the exception of the Lasso model, the models were able to make predictions that are closer to 0 and to 3. Overall, the models demonstrate an increased variability in their predicted outcomes.

The results of the correlation analysis for the regression models trained on the oversampled data using the BoW and TF-IDF input matrices are presented in Table 5.9. Despite the decrease in model performances for most of the regression models, the correlation coefficients of the majority of the models for most of the condition score classes have increased, indicating a stronger correlation between the predictions and the actual values for the oversampled dataset.

The foremost finding in the result is that the correlation coefficients for the gradient boosting model have increased, both for the coefficient representing the relation between the predictions and actual values of the target variable for all observations in the dataset and for the condition score classes separately. The correlation coefficients of 0.9144 (BoW) and 0.9223 (TF-IDF) between the overall predictions and actual values reveal a strong correlation and reflect the

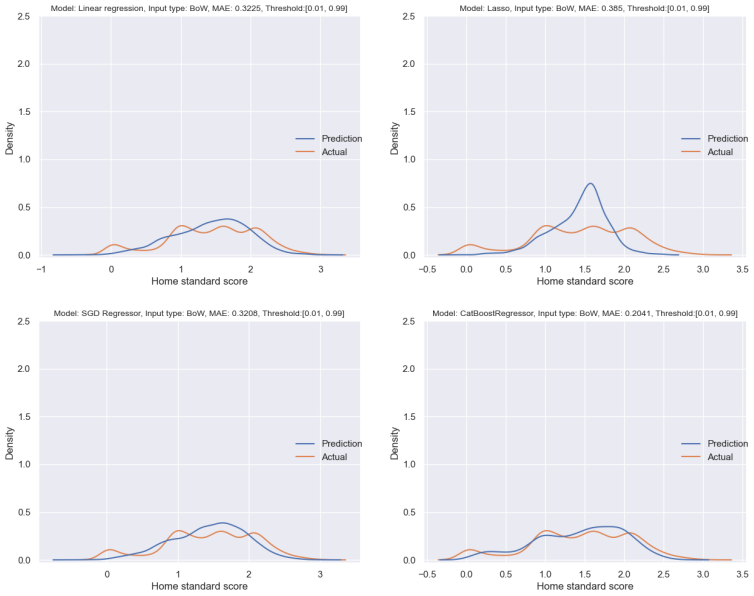

Figure 5.3: Kernel density estimation plots for the distribution of the actual target values versus the predictions made by the regression models trained on the oversampled dataset using the BoW input matrix.

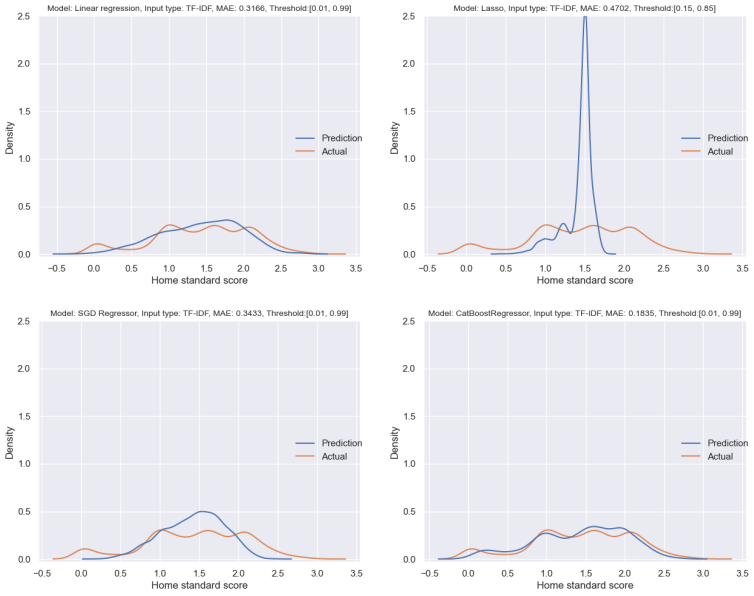

Figure 5.4: Kernel density estimation plots for the distribution of the actual target values versus the predictions made by the regression models trained on the oversampled dataset using the TF-IDF input matrix.

increase in the model performances for the gradient boosting models when the data is oversampled. The results show that gradient boosting has strong predictive capabilities for the extreme values of the condition score, as the condition score classes 0 and 3 have higher correlation coefficients than the other classes. While the new correlation coefficients for the gradient boosting models are promising, there still is a moderate to small correlation between the actual and predicted values of the observations with a condition score that belongs to the condition score class 1, and 2. This indicates that when trained on the oversampled data set, the gradient boosting models gain performance and can predict more accurately for the well-maintained and badly maintained dwellings, but still struggle to identify moderately maintained dwellings and separate them from the rest.

Another important aspect of the results of the correlation analysis is that almost all models, except for the Lasso regression model when trained with the TF-IDF input matrix, have acquired higher correlation coefficients for the observations from condition score 0, which implies that the models trained on oversampled data are better at predicting the condition score of the well-maintained dwellings based on their advertisement text. The models that have gotten especially stronger at predicting the condition score of well-maintained dwellings as a result of the oversampling are the linear regression model and the SGD regression model. The correlation coefficients for the condition score class 0 have increased from 0.2816 to 0.5006 for the linear regression model, and from 0.2879 to 0.4861 for the SGD regression model when the models were trained with the BoW input matrix.

Overall, compared to the results of the correlation analysis on the models trained with an imbalanced dataset, most of the models demonstrated a better performance at predicting the condition score of the well-maintained homes from the condition score class 0. For the rest of the condition score classes, there is little to no increase in the correlation coefficients between the predictions and the target for the majority of the models, indicating that encounting challenges when predicting for moderately maintained and badly maintained homes is a commonality across the models.

	Input type									
	BoW					**TF-IDF**				
CS Class	0	1	2	3	All	0	1	2	3	All
LinReg	0.5006	0.3381	0.2281	0.2218	0.7792	0.5048	0.3521	0.2340	0.2588	0.7910
LASSO	0.2185	0.3645	0.1599	0.0399	0.6555	0.0841	0.2877	0.0682	-0.0228	0.4551
SGD	0.4861	0.3508	0.2330	0.2099	0.7791	0.3029	0.4271	0.2182	0.0826	0.7451
CatBoost	0.8801	0.3929	0.2938	0.6800	0.9144	0.9480	0.3739	0.2921	0.7709	0.9283

Table 5.9: Correlation coefficients indicating the correlation beween the actual and predicted values of the target variable. The predictions were acquired by the regression models trained on the BoW and TF-IDF input matrices extracted from the oversampled data set.

5.3 BERT-based models

We have fine-tuned three pre-trained BERT-based models that are specifically trained on Norwegian text. First, we have fine-tuned the models at their frozen state and investigated the impact of different hyperparameters on the model performance. These hyperparameters were the maximum sentence length allowed during the fine-tuning, batch size, and learning rate. Then, we proceeded to further fine-tune the models on the oversampled data using the optimal parameters determined. Finally, we have fine-tuned the same models in their unfrozen states, with and without oversampling.

In this section, we present results obtained from fine-tuning the BERT-based models in terms of MAE, show distribution plots comparing the distribution of the actual values of the target variable againtst the predictions made by the models, and show the correlation between the predictions and actual values for dwellings with condition scores that lies in the different condition score classes.

5.3.1 Frozen pre-trained BERT-based models on imbalanced data

Table 5.10 show the MAE's produced by the three pre-trained BERT-based models fine-tuned in their frozen states with 14 different combinations of (batch size, maximum sentence length) pairs.

Our findings indicate that there is very little correlation between the hyperparameter choice and the error made by the model, as most of the MAEs lie very close to each other in magnitude, and there is no clear pattern between the error and the variable choice. It was assumed in the previous chapter that the larger the batch size and maximum sentence length are, the more information each batch contains during fine-tuning. Hence, the obtained results deviates from the hypothesized trends.

For different values of the maximum sentence length, we observe that the NB-BERT$_{\text{BASE}}$ model seemed to most of the time yield the least amount of error, whereas NorBERT had the poorest performance. The only exception was NorBERT$_2$, which yielded a lower MAE than the NB-BERT$_{\text{BASE}}$ model when the maximum sentence length was set to 128. Regardless of the maximum sentence length, NorBERT has yielded the best MAE when the batch size was 128. For the other models, the optimal value of the batch size varied with the maximum sentence length. The best-performing model achieved in this part of the thesis was the NB-BERT$_{\text{BASE}}$ model fine-tuned on a batch size of 256 and a maximum sentence length of 512, which yielded an MAE of 0.2689.

	Batch size				
Max length=512	64	128	256	320	512
NorBERT	0.2910	***0.2858***	0.2911	0.2901†	-
NorBERT_2	0.2771	0.2882	0.2824	**0.2815**†	-
$\text{NB-BERT}_{\text{BASE}}$	0.2721	0.2748	***0.2689***	0.2772†	-
	Batch size				
Max length=256	64	128	256	320	512
NorBERT	0.2927	**0.2876**	0.2968	0.2911	0.2946
NorBERT_2	0.2761	0.2879	0.2845	**0.2804**	0.2878
$\text{NB-BERT}_{\text{BASE}}$	0.2756	**0.2700**	0.2715	0.2797	0.2796
	Batch size				
Max length=128	64	128	256	320	512
NorBERT	0.2877	**0.2864**	0.2913	0.2915	0.2961
NorBERT_2	***0.2737***	0.2797	0.2738	0.2763	0.2823
$\text{NB-BERT}_{\text{BASE}}$	0.2767	0.2777	**0.2755**	0.2787	0.2809

Table 5.10: Mean absolute errors acquired by 3 pre-trained BERT models for different batch sizes. The best result for each model is written in bold font.

Keeping the optimal hyperparameter values for the batch size and the maximum sentence length, the models were fine-tuned on the imbalanced data set with different learning rates. The results are presented in Table 5.11. Once again, the results did not establish a distinct pattern explaining the impact of the learning rate on the model performances. Overall, we were unable to detect a relationship between the learning rate and the error made, as all three models had different values as their optimal learning rate, and the MAEs acquired with the other learning rate did not have an increasing nor decreasing trend with the increasing values of the learning rate. The model with the lowest MAE was $\text{NB-BERT}_{\text{BASE}}$, fine-tuned with a batch size of 256, maximum sentence length of 512 and learning rate of 0.005, and yielded an MAE of 0.2643.

In general, the frozen pre-trained BERT-based models fine-tuned on the imbalanced data set performed worse than the linear regression models. The optimal BERT-based model in its frozen state performed apporximately 1.1 times worse than the best-performing regression model, namely the gradient boosting model trained on the imbalanced data set using the BoW input matrix with the least strict term filtering threshold, a learning rate of 0.01, a tree depth of 8 and 4000 trees which yielded an MAE of 0.2396.

			Learning rate				
	BS	ML	0.0001	0.0005	0.001	0.005	0.01
NorBERT	128	512	0.3039	**0.2842**	0.2858	0.3277	0.2986
NorBERT_2	64	128	0.2814	0.2898	**0.2737**	0.3163	1.6189
$\text{NB-BERT}_{\text{BASE}}$	256	512	0.3029	0.2786	0.2689	**0.2643**	0.2662

Table 5.11: Mean absolute errors acquired by 3 frozen pre-trained BERT models for different learning rates. The batch sizes (BS) for *norbert*, *norbert*2, and *nb-bert-base* are 128, 64 and 256, whereas the maximum sentence lengths (ML) are 512, 128 and 512, respectively. The best result for each model is written in bold font.

We have investigated the correlation between the actual values of the target variable and the values predicted by the frozen BERT-based models. The results are presented in 5.12. The correlation coefficients between the actual values of the observations in the test set and the predictions of all three models indicate moderate correlation. The correlation computed according to the condition score class is even weaker, with some of the models yielding negative values for the coefficients. In fact, all three models yielded negative correlation coefficients for observations with a condition score from the condition score class 3. The correlation coefficients align with the poor performances the models have previously demonstrated.

Additionally, we have plotted the distributions of the predicted and actual variables for the optimal models fine-tuned using each model type, as shown in 5.5. In a recurring fashion, the models fine-tuned on imbalanced data have failed to make good predictions on dwellings that are well-maintained or of bad standard.

These findings shows that despite the classification power of the BERT models that has been demonstrated in literature, frozen BERT-based models fine-tuned on imbalanced data fall short in effectively carrying out a regression task.

	Model params			**CS Class**				
Model name	BS	ML	LR	0	1	2	3	All
NorBERT	128	512	0.0005	-0.0299	0.2342	0.0932	-0.2287	0.3247
NorBERT_2	64	128	0.001	0.0104	0.2427	0.1142	-0.0516	0.3587
$\text{NB-BERT}_{\text{BASE}}$	256	512	0.005	0.0317	0.3196	0.1267	-0.0643	0.4562

Table 5.12: Correlation coefficients indicating the correlation beween the actual and predicted values of the target variable. The predictions were acquired by the frozen BERT-based models fine-tuned on the imbalanced data set.

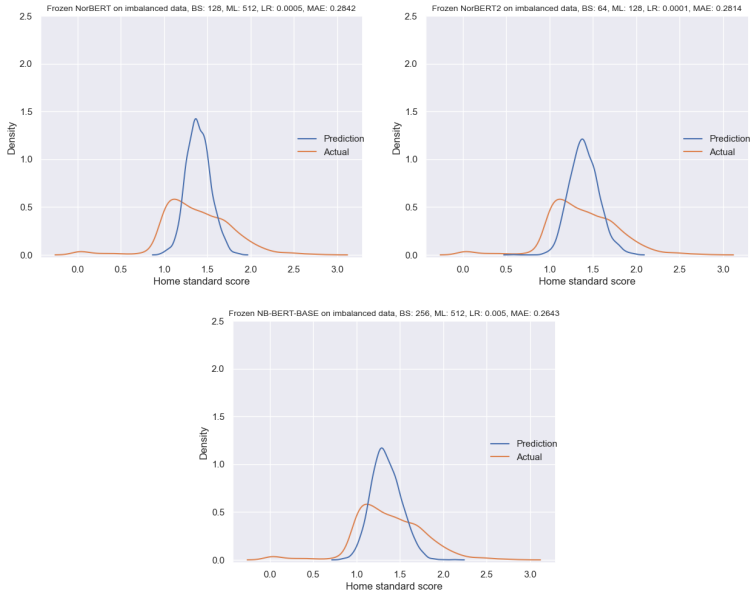

Figure 5.5: Kernel density estimation plots for the distribution of the actual target values versus the predictions made by the frozen BERT-based models fine-tuned on the imbalanced dataset.

5.3.2 Frozen pre-trained BERT-based models on oversampled data

The models were fine-tuned in their frozen state on the oversampled data using the optimal parameters determined in the previous section. The results are presented in 5.13. The results obtained in this part was similar to the results we have acquired from the regression models trained on the oversampled data. All three models have responded negatively to the oversampling, where the MAEs were as high as 0.43. The model that yielded the lowest error in this case was the NorBERT_2 model fine-tuned with a batch size of 64, maximum sentence length of 128 and learning rate of 0.001, and obtained an MAE of 0.4089. Repeatedly, this result was approximately 2.23 times worse than the best-performing regression model trained on the imbalanced data set, namely the gradient boosting regression model fed with the TF-IDF input matrix computed with the least strict term filtering threshold.

Model	BS	ML	LR	MAE
NorBERT	128	512	0.0005	0.4294
NorBERT_2	64	128	0.001	**0.4089**
$\text{NB-BERT}_{\text{BASE}}$	256	512	0.005	0.4192

Table 5.13: Mean absolute errors acquired by frozen pre-trained BERT models fine-tuned on oversampled data set with the optimal values of batch size, maximum sentence length and learning rate found in Section 4.4.2.

In general, we have observed that the pre-trained BERT-based models have

performed much worse than the simpler regression models, and it was challenging to observe a relationship between the varying hypreparameters and the results. This observed phenomenon might be the result of the frozen nature of the models, where all 110 million learnable model parameters are always kept constant during the fine-tuning. This aspect of the frozen models is discussed further in the next chapter.

5.3.3 Unfrozen pre-trained BERT-based models on imbalanced data

The pre-trained BERT-based models were fine-tuned on the imbalanced data in their unfrozen states, and a similar hyperparameter search was conducted on these models. Fine-tuning unfrozen BERT-based models is a resource-demanding process, where several millions model parameters are updated during the fine-tuning which the requires more memory than simply fine-tuning a frozen model. Due to the demanding nature of the unfrozen models compared to the frozen models, and the limitations with the computational resources, the only batch size and maximum sentence length combinations examined were (256, 64) and (128, 128), and the results are shown in Table 5.14 and 5.15. Eight variations of the learning rate was also utilized and the impacts of the hyperparameter choices were investigated simultaneously.

The results have demonstrated that for learning rates higher than 5×10^{-5}, most of the models tend to predict the same value for all observations in the test set. For instance, the models yielding an MAE of 1.6189 have predicted the value 3 for all observations, whereas the models with an MAE of 1.3811 have predicted 1. This may indicate that the learning rate was too large, and that the model stops learning as a local minima is reached. For the majority of the models, a learning rate of 1×10^{-5} yielded the least MAEs.

Batch size=256	Learning rate			
Max length=64	0.005	0.001	0.0005	0.0001
NorBERT	1.3811	1.3811	0.3058	0.2796
NorBERT_2	1.3811	1.6189	1.6189	1.6189
$\text{NB-BERT}_{\text{BASE}}$	1.3811	1.6189	1.6189	0.2616
oversample=`False`	Learning rate			
freeze=`False`	5e-5	1e-5	5e-6	1e-6
NorBERT	**0.2472**	0.2535	0.2577	0.2955
NorBERT_2	1.6172	**0.2468**	0.2498	0.2552
$\text{NB-BERT}_{\text{BASE}}$	0.2513	**0.2419**	0.2445	0.2724

Table 5.14: MAE acquired by unfrozen pre-trained BERT models, fine-tuned with a batch size of 256, maximum sentence length of 64 and various learning rates. The best result for each model is written in bold font.

We have observed that the models fine-tuned with a batch size of 256 and maximum sentence length of 64, regardless of the learning rate, have produced smaller MAEs whenever the learning rate was greater than 0.0001. This conveys that the batch size was more determinative on the model performance than the

maximum sentence length, and choosing larger batch sizes may result in smaller errors. This also indicates that batches containing larger amounts of information tend to yield better results.

Batch size=128	Learning rate			
Max length=128	0.005	0.001	0.0005	0.0001
NorBERT	0.3071	1.3811	1.3811	0.3379
NorBERT_2	1.3811	1.6189	1.6189	1.6189
$\text{NB-BERT}_{\text{BASE}}$	1.3811	1.3811	1.3811	**0.2581**
oversample=`False`	Learning rate			
freeze=`False`	5e-5	1e-5	5e-6	1e-6
NorBERT	0.3091	**0.2538**	0.2497	0.2849
NorBERT_2	1.6189	**0.2796**	0.2455	0.2410
$\text{NB-BERT}_{\text{BASE}}$	0.2875	0.2689	0.2454	0.2517

Table 5.15: MAE acquired by unfrozen pre-trained BERT models, fine-tuned with a batch size of 128, maximum sentence length of 128 and various learning rates. The best result for each model is written in bold font.

A correlation analysis was conducted on the predictions produced by the best-performing unfrozen BERT-based models fine-tuned on the imbalanced data. The results are presented in Table 5.16. Unfreezing the models has resulted in much higher correlation coefficients for observations from all condition score classes, and for the overall correlation between the predictions and the actual values of the condition score of all observations in the test set. The magnitude of the coefficients are similar across all three models for all classes and for the overall results. This demostrates that utilizing the BERT-based models in their unfrozen state is indeed a method that yields higher model performances. All three models had the highest correlation coefficient for the observations from the condition score class 1, which is similar to the results obtained with the simpler regression methods. This finding implicates that the models trained on the imbalanced dataset mostly predict values around the average condition score of the full data set, which lies in the condition score class 1.

	Model params			**CS Class**				
Model name	BS	ML	LR	0	1	2	3	All
NorBERT	256	64	5e-5	0.3105	0.4432	0.2144	0.1227	0.5998
NorBERT_2	256	64	1e-5	0.2993	0.4359	0.2711	0.1671	0.6112
$\text{NB-BERT}_{\text{BASE}}$	256	64	1e-5	0.3720	0.4067	0.2418	0.2273	0.6218

Table 5.16: Correlation coefficients indicating the correlation between the actual and predicted values of the target variable. The predictions were acquired by the unfrozen BERT-based models fine-tuned on the imbalanced data set.

The distribution of the actual values of the target variable against the distribution of the predictions made by the unfrozen models are displayed in Figure

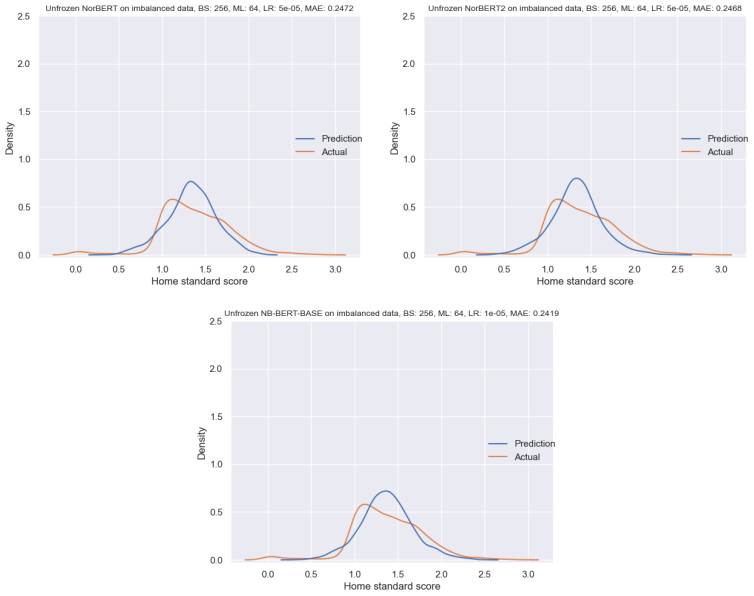

Figure 5.6: Kernel density estimation plots for the distribution of the actual target values versus the predictions made by the unfrozen BERT-based models fine-tuned on the imbalanced dataset.

5.6. The plots reveal that the distributions of the predictions made by the unfrozen models cover a larger range within the range of the condition scores than the frozen models. This phenomenon explains the lower MAEs acquired by the models, indicating that the unfrozen models capable of predicting the extreme values of the condition score, namely values closer to 0 or 3.

5.3.4 Unfrozen pre-trained BERT-based models on oversampled data

The experiments conducted with the unfrozen models fine-tuned on the imbalanced data were replicated on the unfrozen models fine-tuned on the oversampled data. The results are shown in Tables 5.17 and 5.18 for the unfrozen models fine-tuned with batch size/maximum sentence length combinations of (256, 64) and (128, 128).

Surprisingly, the results acquired from these experiments has demonstrated a different impact of the batch size on the MAEs produced by the model. In this case, we observed that the majority of the models fine-tuned with a the smaller batch size, regardless of the learning rate, yielded smaller errors than the models fine-tuned with the larger batch size. While this is unexpected, it might indicate that with a balanced dataset, the amount of information required in each batch during the fine-tuning of the model may be reduced and optimal model performances can still be achieved. For the different choices of the learning rate, the results do not convey a clear pattern indicating the relationship between the learning rate and the MAEs. However, the unfrozen models fine-tuned with smaller learning rates continued to predict the same value for every observation in the test set, which yielded very large MAEs and poor model per-

forances. This phenomenon shows that unfrozen models learn better when they are fine-tuned with smaller learning rates.

Models fine-tuned in this section have also yielded the smallest errors across all models trained or fine-tuned in the scope of this thesis. The unfrozen NorBERT_2 model fine-tuned on the oversampled data with the batch size and maximum sentence length of 128, and a learning rate of 1×10^{-5} resulted in an MAE of 0.1578, making it the strongest regression model we have achieved in this thesis.

Batch size=256	Learning rate			
Max length=64	0.005	0.001	0.0005	0.0001
NorBERT	1.4416	1.4416	1.4416	0.1979
NorBERT_2	1.4416	1.5584	1.5584	1.5584
$\text{NB-BERT}_{\text{BASE}}$	1.4416	1.5584	1.5584	0.1846
oversample=`True`	Learning rate			
freeze=`False`	5e-5	1e-5	5e-6	1e-6
NorBERT	**0.1841**	0.1934	0.2329	0.3470
NorBERT_2	1.5584	**0.1715**	0.1728	0.2599
$\text{NB-BERT}_{\text{BASE}}$	0.1933	0.1723	**0.1712**	0.2956

Table 5.17: MAE acquired by unfrozen pre-trained BERT models, fine-tuned on the oversampled data set with a batch size of 256, maximum sentence length of 64 and various learning rates. The best result for each model is written in bold font.

Batch size=128	Learning rate			
Max length=128	0.005	0.001	0.0005	0.0001
NorBERT	0.5351	0.5293	0.5305	**0.2008**
NorBERT_2	1.4416	1.5584	1.5584	1.5584
$\text{NB-BERT}_{\text{BASE}}$	1.4416	1.4416	1.5584	0.2917
oversample=`True`	Learning rate			
freeze=`False`	5e-5	1e-5	5e-6	1e-6
NorBERT	0.2033	0.2038	0.2167	0.3185
NorBERT_2	1.5584	**0.1578**	0.1590	0.2155
$\text{NB-BERT}_{\text{BASE}}$	0.1689	0.2194	**0.1657**	0.2577

Table 5.18: MAE acquired by unfrozen pre-trained BERT models, fine-tuned on the oversampled data with a batch size of 128, maximum sentence length of 128 and various learning rates. The best result for each model is written in bold font.

Table 5.19 shows the results of the correlation analysis conducted on the best-performing versions of the three pre-trained unfrozen BERT-based models in this thesis. The results of the analysis also suggests that the performance

gain of the unfrozen models due to the oversampling is phenomenal. All three models demonstrated very strong correlations between the predictions and the actual values of the target in all observations, and the observations from the condition score classes 0 and 3. This shows that the predictive power of the BERT-based models are highly dependent on whether the model parameters are updated during the training, and the skewness of the target variable. Similar to the other models in this thesis trained or fine-tuned on the oversampled data set, the correlation coefficients acquired by the predictions on dwellings from condition scores 1 and 2 indicate that the biggest struggle of the oversampled models is to identify between moderately maintained dwellings in the test set and make good predictions for these observations.

	Model params			**CS Class**				
Model name	BS	ML	LR	0	1	2	3	All
NorBERT	256	64	5e-5	0.9708	0.4023	0.3139	0.8726	0.9486
NorBERT_2	128	128	1e-5	0.9607	0.4282	0.3893	0.9224	0.9548
$\text{NB-BERT}_{\text{BASE}}$	128	128	5e-6	0.9586	0.4477	0.3165	0.8751	0.9457

Table 5.19: Correlation coefficients indicating the correlation between the actual and predicted values of the target variable. The predictions were acquired by the frozen BERT-based models fine-tuned on the oversampled data set.

The distribution of the actual values of the target variable against the distribution of the predictions made by the unfrozen models fine-tuned on the oversampled data are shown in Figure 5.7. We observe that the distribution plots of the actual and predicted values align with with each other in harmony for the observations with a condition score values away from the median in both directions, whereas the plots slightly diverge from each other for condition score values around the median. The distribution graphs provide empirical evidence that supports the previous observations deduced from the MAEs yielded by the models and the results of the correlation analysis.

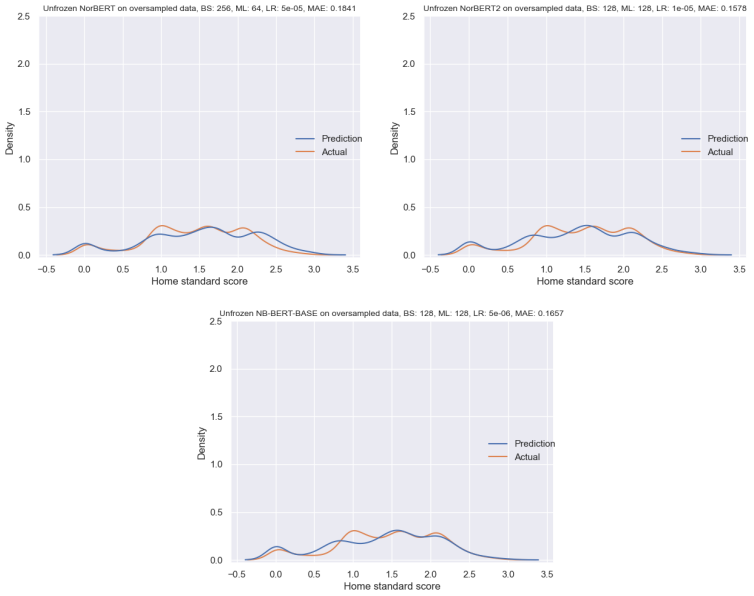

Figure 5.7: Kernel density estimation plots for the distribution of the actual target values versus the predictions made by the unfrozen BERT-based models fine-tuned on the oversampled dataset.

6 Discussion

In this thesis, we have conducted a text-based regression analysis to predict the condition score of dwellings based on the textual features extracted from their advertisement texts. The data used in this thesis was collected from 5 different sources, underwent a cleaning process, and were merged with a complex merging procedure. We have employed BoW-based models trained on BoW and TF-IDF input matrices, and three pre-trained BERT-based models for Norwegian that are publicly available. In this chapter, we discuss the results obtained from these models, evaluate the model behaviors and performances under different conditions, define the advantages and disadvantages of different models, and propose methods to enhance the model performances.

One of the challenges faced in this thesis was combining the datasets from 5 distinct sources, where the data was contained in various formats. Many observations had to be discarded in each data set during the merge process in order to ensure a high quality data set. We did not include the condition report of a dwelling unless we were confident that a corresponding advertisement text for the dwelling exists in our datasets. Moreover, most of the condition reports were dated from the same time period. With more data, we might have obtained a dataset with a less skewed target variable, which could have been helpful for achieving models with better performances.

To address the skewed distribution of the target variable, we created an oversampled version of the dataset and trained the models on this set. The oversampling method used in this thesis to generate synthetic data was an adaptation of the SMOTE method that is commonly used in imbalanced classification tasks. We have adapted this method to our experiments by encoding the continuous condition scores to discrete condition score classes according to a pre-defined encoder function. There are other methods in literature developed for addressing the skewness of the target variable in a regression task that have demonstrated promising results, such as SMOTER (Smote for Regression) (Torgo, Ribeiro, Pfahringer, & Branco, 2013) and SMOTER with Gaussian noise (SMOGN) (Branco, Torgo, & Ribeiro, 2017). The oversampling method chosen for this thesis, however, was created specifically for the aim of this thesis and did not follow an established procedure. Additionally, the intervals defining the range of each condition score class were set manually. It would be advisable to address the problem of having an imbalanced target with methods that has less impact from the potential bias in the intervals.

For the BoW-based models trained on the imbalanced data set, we have investigated the impact of the hyperparameters of the models on the model performance, as well as how the models behaved under oversampling. The findings show that for the majority of the models trained in this exploration, choosing a less strict term filtering threshold and computing the BoW-based matrices on larger vocabularies yielded smaller MAEs and improved the model performances. This indicates that training the models with a larger number of input features may have helped preserve the information contained in the advertisement text and resulted in performance gain, regardless of the type of the input feature matrix.

The results obtained from the BoW-based models we trained on the oversampled data set showed that the oversampling procedure has resulted in perfor-

mance loss for most of the regression model types. The only exception was the gradient boosting regression model, which in fact gained performance due to the oversampling and yielded some of the best results across all models trained on the imbalanced data. This finding implies that the signals hidden in the textual features are not strong enough for the simpler regression models to capture, but are sufficient for the gradient boosting model.

The MAEs exhibited by the BoW-based models trained both on the imbalanced data and oversampled data have shown that in general, the least strict term filtering threshold yields the least error across the models. One may assume that allowing all words in the corpus in the computation of the input matrices and removing the term filtering parameters would result in high performance gains and a decrease in the MAEs obtained. However, owing to memory limitations, training the BoW-based models on input matrices computed from the entire vocabulary was not possible for this thesis.

Another limitation of the BoW-based models was that the input features fed into these models were the BoW and TF-IDF matrices, two feature extraction methods of which computations purely rely on the frequency of word occurrences in the documents, rather than the actual context hidden in the sentences. In addition, we have removed numerical digits from the texts, which may have resulted in information loss as the tokens representing a year (e.g. the build year of a house) were not considered. This decision could potentially account for the BoW-based models' performances.

Based on the analysis of the results obtained from the frozen fine-tuning of the pre-trained BERT-based models on imbalanced data, we have deduced that the choice of the hyperparameters like the batch size and maximum sentence length had little to no effect on the model performance. In fact, the frozen models performed often worse than the BoW-based models trained on imbalanced data. This prompted us to question the fitness of the BERT-based models in the context of regression, as the BERT models, despite being seen as powerful, state-of-the-art models, failed to compete with the results obtained by the BoW-based models trained on the imbalanced data. This was also the case for the fine-tuning results of the BERT-based models on the oversampled data. Even though the frozen model performances did not meet our expectations, the outcomes may be the result of the model parameters not being updated during the fine-tuning.

When the BERT-based models were fine-tuned in their unfrozen states, however, the resulting MAEs were slightly less than the errors made by the BoW-based models. This outcome support that the models are able to learn more from the input data if the model parameters are updated during fine-tuning. As opposed to the regression models, the BERT-based models responded well when fine-tuned on the oversampled data set. NorBERT_2 yielded the best-performing model with an MAE of 0.1578. This value was the lowest error achieved in the coverage of this thesis.

Because of the limited computational resources, the BERT-based models were not fine-tuned with batch sizes and maximum sentence lengths larger than 256 and 128, respectively.

During the time scope of this thesis, the LTG at the University of Oslo has published a new version of their NorBERT models, namely the NorBERT_3 (Samuel et al., 2023). The limited time frame of this thesis and prioritization of other key aspects have prevented us from exploring their new model. The

NorBERT$_3$ model was pre-trained on a corpus yielding 25 billion word tokens, which is 12.5 larger than the number of tokens in the corpus used to train the first *NorBERT* model, and 1.6 times larger than the corpus of NorBERT$_2$. The new model was also shown to outperform the older versions in various downstream tasks. Considering the large size of its training corpus and the promising performance it offers, it would have been beneficial to conduct experiments using the NorBERT$_3$ model and compare the results to the other pre-trained BERT-based models for Norwegian.

Despite the limitations of the BoW-based models, the unexpected behavior of the frozen BERT-based models, and other constraints regarding computational resources, we successfully achieved our objective of predicting the condition score of dwellings based on the advertisement text. Even though there exists scope for refinements in order to further optimize the models, this thesis has shown that text-based regression analysis with BoW-based and BERT-based approaches is a viable and promising downstream task.

7 Conclusion

The aim of this thesis was to predict the condition score of dwellings in the real estate market in Norway. We approached the problem by creating increasingly complex regression models. We conducted hyperparameter tuning and oversampling as an attempt to improve the model performances.

First, we created a labeled training data set by collecting data from five different sources. Three of these sources contained detailed condition reports on dwellings in Norway, conducted by a certified assessor. The reports consisted of a number of checkpoints that are relevant to the units, and each checkpoint was score scores from 0 to 3, 0 indicating perfect condition. We computed an aggregated condition score for each dwelling, and the resulting score was used as the target variable to be predicted by the models. Then, we obtained text features for each dwelling, extracted from the textual content of their respective listing advertisements. Finally, we trained BoW-based models and fine-tuned three BERT-based models that were specifically trained on Norwegian text and are considered one of the state-of-the-art methods for NLP tasks. The models were trained on imbalanced and oversampled versions of the training data.

Surprisingly, the BERT-based models did not perform as well as expected, as these models and the models trained with BoW-approaches on the imbalanced data had similar performances in terms of MAEs obtained. Among the models explored in this thesis, the gradient boosting regression model and the unfrozen NB-BERT$_{\text{BASE}}$ model, both trained on the oversampled data set, stood out with noteworthy results, yielding mean absolute errors of 0.1835 and 0.1578 respectively.

Estimating the real estate condition score was, to the best of our knowledge, was a task that has never been studied before. This thesis can potentially contribute to the advancement of knowledge in the real estate market, and the results can be used in further research when it comes to linking home quality with price.

References

Branco, P., Torgo, L., & Ribeiro, R. P. (2017). SMOGN: a pre-processing approach for imbalanced regression. In *First international workshop on learning with imbalanced domains: Theory and applications* (pp. 36–50).

Cauchy, A. (1847). Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, *25*(1847), 536–538.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, 321–357.

CONQUAS, Singapore Building and Construction Authority. (n.d.). Retrieved from https://www1.bca.gov.sg/buildsg/quality/conquas

Dereli, N., & Saraclar, M. (2019, July). Convolutional Neural Networks for Financial Text Regression. In *Proceedings of the 57th annual meeting of the association for computational linguistics: Student research workshop* (pp. 331–337). Florence, Italy: Association for Computational Linguistics. Retrieved from https://aclanthology.org/P19-2046 doi: 10.18653/v1/P19-2046

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dorogush, A. V., Ershov, V., & Gulin, A. (2018). CatBoost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*.

Dreyer, C. V. (2018, October). *Finanskrisen på norsk.* Retrieved from https://eiendomnorge.no/blogg/finanskrisen-pa-norsk-article599-923.html

Eiendom Norge. (2023). *Boligprisutvikling.* Retrieved from https://eiendomnorge.no/boligprisstatistikk/statistikkbank/

Eitrheim, , & Erlandsen, S. K. (2004). Norges bank occasional papers no. 35. In (chap. Chapter 9 – House price indices for Norway 1819–2003). Addison-Wesley. Retrieved from https://www.norges-bank.no/globalassets/upload/hms/pdf/hmsi_chapter9.pdf?v=03/09/2017122524&ft=.pdf

Eurostat. (2023). *Distribution of population by tenure status, type of household and income group - eu-silc survey.* Retrieved from https://ec.europa.eu/eurostat/web/products-datasets/-/ilc_lvho02

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational statistics & data analysis*, *38*(4), 367–378.

Goldberg, Y. (2017). *Neural Network Methods for Natural Language Processing* (Vol. 37). San Rafael, CA: Morgan & Claypool. doi: 10.2200/S00762ED1V01Y201703HLT037

Hangaard, S. (2020, Oct). *The Norwegian property market in Covid19-Times.* Retrieved from https://svw.no/artikler/the-norwegian-property-market-in-covid19-times

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The Elements of Statistical Learning.* New York, NY, USA: Springer New York Inc.

Kogan, S., Levin, D., Routledge, B. R., Sagi, J. S., & Smith, N. A. (2009). Predicting risk from financial reports with regression. In *Proceedings of human language technologies: the 2009 annual conference of the north*

american chapter of the association for computational linguistics (pp. 272–280).

Kummervold, P. E., De la Rosa, J., Wetjen, F., & Brygfjeld, S. A. (2021). Operationalizing a National Digital Library: The Case for a Norwegian Transformer Model. In *Proceedings of the 23rd nordic conference on computational linguistics (nodalida)* (pp. 20–29). Reykjavik, Iceland (Online): Linköping University Electronic Press, Sweden. Retrieved from https://aclanthology.org/2021.nodalida-main.3

Kutuzov, A., Barnes, J., Velldal, E., Øvrelid, L., & Oepen, S. (2021, May 31–2 June). Large-Scale Contextualised Language Modelling for Norwegian. In *Proceedings of the 23rd nordic conference on computational linguistics (nodalida)* (pp. 30–40). Reykjavik, Iceland (Online): Linköping University Electronic Press, Sweden. Retrieved from https://aclanthology.org/2021.nodalida-main.4

Lee, H. D., Lee, S., & Kang, U. (2021). Auber: automated [bert] regularization. *Plos one*, *16*(6), e0253241.

Loshchilov, I., & Hutter, F. (2019). *Decoupled Weight Decay Regularization.*

Mamre, M. O., & Sommervoll, D. E. (2022). Coming of Age: Renovation Premiums in Housing Markets. *The Journal of Real Estate Finance and Economics*, 1–36.

Mathur, S. (2019). House price impacts of construction quality and level of maintenance on a regional housing market: Evidence from King County, Washington. *Housing and Society*, *46*(2), 57–80.

Nguyen, D., Smith, N. A., & Rose, C. (2011). Author age prediction from text using linear regression. In *Proceedings of the 5th acl-hlt workshop on language technology for cultural heritage, social sciences, and humanities* (pp. 115–123).

Nordic Journal of Housing Research. (2021). Eierlinja og sosialdemokratiske likhetsidealer. *Tidsskrift for boligforskning*, *4*(1), 4-6. Retrieved from https://www.idunn.no/doi/abs/10.18261/issn.2535-5988-2021-01-01 doi: 10.18261/issn.2535-5988-2021-01-01

Ooi, J. T., Le, T. T., & Lee, N.-J. (2014). The impact of construction quality on house prices. *Journal of Housing Economics*, *26*, 126–138.

Oust, S. W. J. E., Are; Westgaard, & Yemane, N. K. (2021). *Assessing the explanatory power of dwelling condition in automated valuation models within real estate* (Unpublished master's thesis). NTNU.

Prechelt, L. (2002). Early stopping-but when? In *Neural networks: Tricks of the trade* (pp. 55–69). Springer.

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). *Improving language understanding by generative pre-training* (Tech. Rep.). OpenAI.

Robbins, H., & Monro, S. (1951). A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, *22*(3), 400 – 407. Retrieved from https://doi.org/10.1214/aoms/1177729586 doi: 10.1214/aoms/1177729586

Samuel, D., Kutuzov, A., Touileb, S., Velldal, E., Øvrelid, L., Rønningstad, E., ... Palatkina, A. S. (2023). NorBench – A Benchmark for Norwegian Language Models. In *The 24rd nordic conference on computational linguistics*. Retrieved from https://openreview.net/forum?id=WgxNONkAbz

Statistics Norway. (2012). *Households' income and wealth, 2012.* Retrieved

from https://www.ssb.no/en/inntekt-og-forbruk/statistikker/ifformue/aar

Statistics Norway. (2023a). *Housing conditions, register-based.* Retrieved from https://www.ssb.no/en/bygg-bolig-og-eiendom/bolig-og-boforhold/statistikk/boforhold-registerbasert

Statistics Norway. (2023b). *Konsumprisindeksen.* Retrieved from https://www.ssb.no/statbank/table/03013

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267–288.

Torgo, L., Ribeiro, R. P., Pfahringer, B., & Branco, P. (2013). Smote for regression. In *Progress in artificial intelligence: 16th portuguese conference on artificial intelligence, epia 2013, angra do heroísmo, azores, portugal, september 9-12, 2013. proceedings 16* (pp. 378–389).

Vikør, L. S., Jahr, E. H., Berg-Nordlie, M., & Thorvaldsen, B. (n.d.). *Språk i Norge.*

Wikipedia. (2023). *Finn.no.* Retrieved from https://no.wikipedia.org/wiki/Finn.no

Wilhelmsson, M. (2008). House price depreciation rates and level of maintenance. *Journal of Housing Economics*, *17*(1), 88–101.

A Code repository

The Python code utilized to conduct the experiments in thesis is available on a GitHub repository[3].

The repository is divided into several folders.

- `dataset/` contains functions related to reading data from different files, dividing the data into training, validation and test sets.
- `process/` contains code for preprocessing, cleaning and merging datasets, as well as computing the condition scores from the reports.
- `models/` contains the code used for training regression models and BERT-based models.
- `parameters/` contains additional files containing preset values, e.g. the weights used for computing the aggregated condition scores, and column names used to standardized the data across four data sets can be found here.

B Computational resources

The computational environment in this thesis was provided by Huggingface Spaces. Most of the experiments conducted in this thesis were performed on a cloud computing environment. The Nvidia T4 medium includes 16 GB of GPU memory, along with 8 virtual CPU and 30 GB of RAM. This environment was chosen for the trade-off it provides between the computational capabilities and cost. By utilizing a standardized and consistent computing platform, variability that can arise from using different computation environments is eliminated. Hence, it is ensured that results obtained in this study are comparable to each other. While the memory provided by Nvidia T4 was sufficient for most of the models trained in this thesis, models that require greater storage space were also trained for testing purposes. Whenever this was needed, we switched to the large Nvidia A10G GPU with 24 GB of GPU memory, 12 virtual CPUs and 142 GB of RAM. The results acquired in the latter have been marked with a dagger† throughout this thesis.

[3] https://github.com/AlvaTechnologies/condition-score-estimator