Leveraging Machine Learning and Geo-Analytics in Automatic Valuation Models to advance Real Estate Valuation

Author: Emil Gravier

Supervisor: Andreas Kamilaris
Critical Observer: Alex Chiumento

Bachelor Thesis
Creative Technology
University of Twente

16th of February 2024

Abstract

This research project, titled "Leveraging Machine Learning and Geo-Analytics in Automatic Valuation Models to advance Real Estate Valuation" has the primary objective of improving existing approaches to property valuation. Traditional Automatic Valuation Model (AVM) methodology often falls short in capturing factors readily accessible to human appraisers. The methodology falls short in considering the properties surrounding environment. The limited incorporation of environmental considerations in current AVM is attributed to a combination of insufficient data availability and incomplete modeling approaches.

To address these limitations, we propose extending the current AVM methodology by utilizing data sourced from a digital twin of the Cyprus real estate market. This digital twin offers a comprehensive representation of both physical and environmental aspects of properties, enabling the incorporation of information about properties' surroundings. By leveraging this extensive dataset, the project aims to develop a machine learning-based AVM that improves upon existing state-of-the-art models.

The key focus of this research is to create a predictive model for property prices that benefits from a more comprehensive set of features, particularly those related to the property's environment. The research aims to bridge part of the gap between traditional AVMs and human appraisal by capturing the environmental factors contributing to property valuation. The model aims to provide a more accurate prediction of property prices, addressing the limitations of existing models and enhancing the quality of decision-making for the stakeholders in real estate, especially individual buyers and sellers.

The methodology involves the integration of machine learning techniques to analyze the rich dataset obtained partially from the digital twin. The model's performance will be evaluated against existing state-of-the-art AVM, through a target performance baseline on the metrics MAE, MAPE, RMSE, R^2, Adjusted R^2, PRD and COD. Through this research, we hope to contribute to the AVM development methodology by showing the potential of incorporating environmental factors. The outcomes of this research may have implications for decision-makers, buyers, sellers, investors, and financial institutions, seeking more robust and precise property valuations in the real estate market.

Acknowledgement

I would like to thank the people involved in this research, making my graduation project possible in the first place.

First and foremost, I would like to thank my supervisor Andreas Kamilaris for his support throughout this research project. His feedback and the discussions in our meetings helped immensely. Furthermore, I was consistently encouraged by him to explore the field of machine learning, which was previous to this project largely unknown to me. Especially the encouragement to follow two online courses on machine learning and python programming related to the field assisted me in the realization phase of the project.
Lastly, his reassurance and positive outlook on my progress helped me feel comfortable with the idea that incompleteness is okay and that there are always more ideas to pursue, but that my results were still meaningful.

Secondly, I would like to thank my critical observer Alex Chiumento for partaking in the project, further supervising it and providing an assessment of my work throughout the evaluation process.

Thirdly, I would like to thank Wishal M Sri Rangan for his previous work on his graduation project titled “Creating accurate valuation models for real estate properties”. His insights, results and methodology inspired me at multiple friction points in my projects process and assisted me with creating my own approach of developing the machine learning models for this project and finally evaluating them.

Lastly, I would like to thank the CYENS SuPerWorld Research Group, for the development of the Gaea tool and the data it provided for my project. Without their data, the development of my project would not have been possible.

Table of Contents

List of Figures

Chapter 1 - Introduction

Introduction to the project

The valuation of real estate is a complex task, requiring appraisers to consider multiple factors such as a property's physical condition, its location, market conditions and previous sales of comparable properties. Appraisal is commonly performed by professionals assisted by computer tools, who use their built knowledge and judgment to weigh these mentioned and other factors to arrive at a final value estimation. There are several factors that pertain to a properties value outside of the boundary of the property, some of which can be observed through the use of satellite imagery.

Accurate real estate valuation is crucial to multiple parties. First and foremost, individual buyers and sellers need precise information about the real estate they are interested in buying or selling, to make informed, fair and financially sound transactions. Real estate investors and businesses are similarly concerned in the value of real estate relevant to their operations. Property financing needs to have accurate valuations to ensure that potential loans are sustainable and are appropriate for the real estate in question. Insurance and risk management firms are interested in the value, to ensure that the real estate is properly compensated in the case of damage. The legal system is further interested in valuations in forms of e.g. legal disputes about property ownership. Lastly, the government is highly interested in accurate valuations, especially to apply appropriate taxation.

As stated, the outlined and further parties have an essential interest in reliable real estate valuation, as it guides their decision making. The more accurate and precise these valuations are, the better their decision making is going to be.

In the context of this project, the primary ethical motivation is to showcase a path towards enabling individual buyers and sellers to access accurate information about their property's value. As real estate valuation and the tools to perform it are currently expensive, this group is financially disadvantaged in their accessibility of accurate valuations. A cost-effective and decently performing automatic valuation model (AVM) could work towards leveling the playing field for this financially disadvantaged group.

Problem statement

For the graduation project “Using Artificial Intelligence in Real Estate: Can we improve automatic valuation models of properties?”, the challenge will be to improve upon the existing automatic valuation models approach towards valuation. Traditional AVM methodology does not fully capture some of the factors that would be accessible to a human appraiser. The surrounding environment of a property has influence on its value, but current AVM methodology largely does not incorporate these environmental factors, likely due to a combination of a lack of available information and incomplete modelling approaches.

By utilizing data retrieved from a digital twin of the Cyprus real estate market, the current methodology could be extended. This digital twin encompasses a detailed representation of the physical and environmental aspects of properties, enabling the incorporation of more information about the surroundings of a property. Through this access to a rich and wide range of data, the task for this project is to create a machine learning AVM predicting prices with more complete information than existing state-of-the-art models, with the goal of making better predictions.

Research questions

The previous section on the problem statement motivates the research objectives of this project. To achieve the stated goal, the following research question has been developed:

How can current automatic valuation models perform by incorporating machine learning methods trained on rich Geo-analytical real estate data?

To answer this main question, multiple sub-questions have been posed to refine the area of research. Answering these sub-questions will be the refined research goal:

What are the significant external features, observable through satellite imagery, that influence property values?

How can these features be effectively incorporated into machine learning models to enhance AVM accuracy?

How does the developed machine learning AVM compare in performance to state-of-the-art AVMs?

Chapter 2 - Background Research

Chapter Introduction - Background reseach

To improve upon the current state-of-the-art automatic valuation models, first one needs to understand how real estate appraisals work and how AVM play part in this appraisal process. Furthermore, the strengths and potential flaws of current AVM need to be investigated to find areas of improvement. In this chapter, through a literature review, the potential of utilizing machine learning and geospatial data in AVM will be explored.

First, the function of AVM is briefly discussed. Secondly, human appraisal, traditional AVM techniques and machine learning AVM techniques are generally compared. Thirdly, the performance achievable by human real estate appraisers is investigated. Next, hedonic price models, a form of traditional AVM methodology, is discussed in greater detail. Afterwards, previous research outcomes on a closely related project will be shortly showcased. Finally, the software projects providing the geospatial data later used in this research project are outlined.

The goal of this chapter is to gain an understanding on AVM methodology, how machine learning can be applied in this context and finally explain how geospatial data could be valuable in the creation of a machine learning AVM.

Automatic valuation models

Automatic valuation models are used in real estate appraisals to easily compare the value of multiple similar estate properties at the same point in time. [1]. They are software tools that use real estate data from databases to calculate different points of value for real estate analyses. [1]. According to Renigier-Biłozor et al. [1], they can be used for an individual residential mortgage portfolio, capital requirements, securitization, loss given default, loan portfolio trading, quality control, investment advice, litigations, etc.

AVM use multiple data points to estimate the value of a property, such as the property location, square footage, number of bedrooms and bathrooms, age and condition of the property and the historical selling prices of similar properties in the area. AVM have the ability to reduce appraisal cost and increase prediction accuracy in comparison to human appraisals [2].

Appraisal techniques: Manual appraisal, traditional AVM and machine learning AVM.

Hilgers [2] in his thesis describes the difference between manual appraisal value and selling price on average being approximately 13 percent. While manual appraisals have the benefit of oftentimes visiting the property in person, they are influenced by previous appraisals, have a lagging bias as they rely on historical value and can introduce human errors. [2].

AVM can reduce these human appraiser biases and provide more reliable estimates, even if some on-site information is largely inaccessible to them [2]. Currently, appraisals are oftentimes done through a combination of AVM usage and manual appraisals to reach a final valuation.

There are multiple traditional methods that are used in AVM, such as the Comparable Sales Method (CSM), the Hedonic Price Model (HPM) as well as the Spatial-Temporal Model Extension (ST). More recent methods include machine learning techniques, such as using Artificial Neural Networks (ANN), Decision Trees, Random Forest (RF) and Gradient Boosting (GBT). Hilgers [2] elaborates upon these methods and their benefits and limitations as well as evaluating their usefulness.

Hilgers recommends utilizing an extended Hedonic Price Model incorporating spatial-temporal information as a baseline regression model. [2]. He notes that machine learning techniques find mixed results but generally outperform the traditional methodology. While offering more accurate results, machine learning method results suffer from being harder to interpret, explain and test. Hilgers moves on to create an AVM using a combination of traditional and machine learning techniques.

Human level performance in real estate appraisal

Investigating Human-level performance in real estate appraisal is not a simple task. First of all, with current AVM technology being widely adopted, appraisals are rarely done by humans alone. Most people involved in appraisal, whether for their own commercial gain or for example governments trying to appraise for tax reasons, nowadays utilize AVM models to at the very least aid them in their appraisal. Appraisals done by humans alone are in our modern computerized world largely a thing of the past and the focus of research is to create better models and algorithms to perform this appraisal.

The second issue one runs into is the difficulty to directly compare humans to AVMs. The benefit of using AVM's is that they are able to value many heterogeneous properties at reasonable performance levels in a short amount of time. Humans have the ability to visit a human levelcertain property, inspect it in person and narrow their scope to a specific region

of the country. They can build experience in a specific neighborhood over time and specialize on property types. These aspects are not easily possible to achieve through AVM's, even with sophisticated machine learning algorithms.

To get an indication of human level performance, one can look at older studies where human level performance in appraisal was more clearly measured. It's important to note that these studies are somewhat dated, and the landscape has evolved with improved access to property data, such that human performance today could be improved. Cannon and Cole in their research presented in 2011 [3] investigated the accuracy of real estate appraisals prior to the sale occuring based on property sales from the NCREIF National Property Index in the time period from 1984 to 2010. They found that on average, commercial real estate appraisals were more than 12% above or below the subsequent selling price that took place two quarters following the appraisal. In a portfolio context, where individual appraisal errors could be reduced, they still found an average discrepancy between appraisal and selling price of 4% to 5%. They further state that appraisals lag the market conditions, where properties were undervalued in "hot markets" and overvalued in "cold markets". [3] There are further aspects that influence these error rates, depending on who performs the appraisal and for whom it is performed. Some appraisers are incentivized to over- or undervalue a certain property, based on their clients. AVM's in theory should not run into this bias. Internal and external appraisals for a company managing a real estate portfolio will likely also result in different error rates.

Still, we can see their study as an indication of human level performance that a newly developed machine learning AVM should try to surpass.

Hedonic Price Models & Recommended Parameters

The traditional approach to Automated Valuation Models is as previously mentioned the utilization of Hedonic Price Models. They are created on the basis of hedonic price theory, which assumes that an object or goods value is influenced by its individual characteristics. Each characteristic has its own influence on the accepted price and thus value of an object. Hedonic price theory assumes that the goods are heterogeneous and thus don't have an uniform value. The goods value arises from its several partial beneficial characteristics.

In the context of AVMs, the goods are various heterogeneous properties and their unique characteristics are the properties attributes. Here, Hedonic Price Models try to understand the impact of specific characteristics on the price of a property, then filter out the individual preferences of buyers and sellers by averaging the contribution of a single characteristic over the market price to generalize the contribution of a single attribute to the price. Finally, when the average contribution of each characteristic to the market price is

found, property market prices can be simulated based on the several characteristics a property has.

These hedonic price models work on the basis of regression. There are thus two major aspects needed to create a well performing hedonic price model. The specific regression formula forming the hedonic price model and the parameters that make up this formula. The formula itself differs in each model and is found through iteratively improving the formula and tuning the input parameters until the model performs well. More interestingly, the most relevant parameters should be identified, as they can also serve as the parameters or data input in different models besides the Hedonic Price Model.

Metzner & Kindt [4] in 2018 performed research aiming to identify relevant parameters for hedonic property valuation for residential properties through literature research. They have compiled a list of parameters and identified how frequently they have been mentioned in the literature they have found. More frequently mentioned parameters from their research could prove to be useful as input data points, for the to-be-developed machine-learning based AVM in this project.
As their research identifies 407 different mentioned parameters in the literature, they summarize the 10 most frequently mentioned parameters. These include the following in descending order in frequency ranking:

- Year of construction / building age (47 times)
- Building size (39 times)
- Lot size (32 times)
- Garage (number of garages) (28 times)
- Number of bathrooms (28 times)
- District / part of town (21 times)
- Number of bedrooms (19 times)
- Type of residence – single family detached house (19 times)
- City centre / CBD (16 times)
- State of repair / condition (16 times)

Besides these listed frequently mentioned parameters, there are several other frequently mentioned parameters as well as groups of parameters that should be considered in the creation of a model. Once the data available for this bachelor project is examined, the complete list should be revisited to identify other important parameters that could be considered.

Previous work on related project

Previous work by Wishal M Sri Rangan in his bachelor thesis “Creating accurate valuation models for real estate properties” [5] aimed to improve the precision of machine learning based valuation models by incorporating additional real estate contextual information. In his research, he found machine learning models to offer more precise property valuations than traditional methods, if provided with appropriate data. He further evaluated different machine learning methods to select the best performing one to use on a Cyprus real estate property dataset. From evaluating four ML models, Random Forest, Gradient-Boosted Tree, Adaptive Boosting and Rotation Forest , Adaptive Boosting was found to be the best performing model. For future work he recommends to incorporate socioeconomic data, optimize the choice of machine learning model, improve the usage of the available data as well as exploring more sophisticated machine learning techniques such as deep learning models. [5]

CYENS SuPerWorld Research Group Gaea & SuPerWorld Geo-API

To be able to incorporate data and functionality from Gaea and the SuPerWorld Geo-API in a later stage in the development of a machine learning AVM, first these two projects need to be described. This is done to examine what their capabilities are and how the data they provide can be incorporated into the AVM.

Gaea

According to the CYENS SuPerWorld Research Group website [6], Gaea is an AI-powered interactive online tool that offers geo-analytical services. It uses the SuPerWorld Geo-API v2.0 to provide information on the Cyprus real estate market. It further provides information on environmental risks and climatic change. It allows users to compare locations and observe trends and patters in the physical environment and geospatial real estate market.

In regards to the data it could provide for the to-be-developed AVM, it can detect features that might influence property valuation, such as the detection of “*swimming pools, vegetation, and burnt areas nearby the property, as well as land use change, building quality, subsidence, landslides, wildfires, flooding, earthquakes, slope and aspect, geology, precipitation, elevation, land use, proximity to roads, sea, blue-flag beaches, amenities, electricity network, building area, and Natura 2000.*” [6].

According to the Website article, this can be used to comprehensively assess a property’s suitability and potential risks.

If a significant correlation between some of these property factors and the property value can be observed, they should be able to assists in generating more accurate

valuations by an AVM. A machine learning based AVM could be able to find correlations between multiple of these factors to reach more accurate valuation conclusions that might not be possible to make by traditional AVM methodology.

Jamil et al. in their conference paper GAEA - A Country-Scale Geospatial Environmental Modelling Tool: Towards a Digital Twin for Real Estate [7] describe Gaea in more detail. Section 4.4 in the paper describes an overview of the environmental services it provides. Depending on the type of real estate, all service categories could likely play a role in property pricing. Geomorphological characteristics would certainly play a role in possible construction or agricultural land use. Climate monitoring would likely especially play a role in agricultural use. Proximity, such as to Infrastructure or Amenities would likely play a role in all types of property. Geohazards might be a slightly overlooked category in traditional property valuation, depending on the type of geohazard and the previous impact certain geohazards have had on the property market. Land cover monitoring factors likely would also play a role in affecting pricing, but could perhaps be harder to be directly affecting price of an individual property. It could provide insights of how the land use will evolve and might moreso affect certain markets, especially commercial or agricultural real estate.

At this point, these factors could certainly influence the price of a property, but it is yet unclear how they do and if the influence is significant enough to improve predictions through an AVM. At a later stage of development of the actual AVM, these questions will hopefully be resolved.

SuPerWorld Geo-API

As Gaea makes use of the SuPerWorld Geo-API, its capabilities should also be examined.
On the SuPerWorld website [8], the Geo-API is described. The API seems to have had a previous 1.0 version, which was made for a collaboration with WIRE FS, a real estate asset management and advisory firm. It seems that WIRE FS has moved on to develop an AVM themselves.

Version 2.0, according to the SuPerWorld website article [8], has improved capabilities in their services, as well as enabling point and polygon selections to retrieve data from the cyprus map. The services have been described in the previous sections as the Geo-API's main purpose appears to be the backend and data provider for the Gaea tool.

Chapter 3 - Related work

State-of-the-art AVM performance metrics

As AVMs have become more and more frequently used in real estate appraisal, the AVM providers are incentivized to keep the inner workings of their AVM technology secret. With a plethora of AVMs available, providers are not incentivized to disclose their model or their exact performance metrics, as they could get driven out of the market by competitors with better performing models. There is also no clear standardization on AVM performance metrics, making it difficult for users to compare different models performance [10]. Ecker et al. in their exposition on AVM performance metrics first describe the fundamental components of AVMs as the following:

1. A database of recent property sales including location, property characteristics, selling price and date.
2. A dataset of properties and their characteristics, regardless of recently sold or not
3. A theoretical property valuation model that defines the relationship between the value of a property and some or all of its characteristics
4. An algorithm or statistical method that fits the theoretical valuation model

The AVM then can look up the characteristics of a property from the dataset and apply the valuation model to find a valuation or valuation range as well as its performance metrics. [10] According to Ecker et al., unfortunately these performance metrics are not “universally defined, nor consistently calculated.” They further more state that AVM precision metrics often don’t meet widely accepted scientific standards. This makes it difficult for users to compare different models even when valuating the same property. [10]
Ecker et al. go on to compile a table (*Figure 1*) of AVM performance metric thresholds suggested in the literature.

Table 1. *AVM* performance metric thresholds suggested in the literature.

AVM Performance Metric	Reference	AVM Performance Metric Threshold (with comment)
MAPE*	Kirchmeyer & Staas (2008)	10 (Strong) and 15 (Acceptable)
	Rossini & Kershaw (2008)	10 (Reasonable) and 13 (Abs Min)
FSD*	Freddie Mac (2019a)	<13 (High); 13-20 (Medium); >20 (Low)
Error Buckets**		
+/− 10%	Rossini & Kershaw (2008)	50% (Abs Min) and 65% (Reasonable)
	Kirchmeyer (2004)	50%
	AVMetrics (2018)	68-70%
	MBA (2019)	70%
	Veros (2017)	80-90% (Top-Tier AVM)
+/− 15%	Rossini and Kershaw (2008)	65% (Abs Min) and 80% (Reasonable)
	Kirchmeyer (2004)	70%
	AVMetrics (2018)	80%
+/− 20%	Rossini and Kershaw (2008)	80% (Abs Min) and 90% (Reasonable)
Right Tail 20%	AVMetrics (2018)	10%
Failure Rates**		
+/− 10%	Rossini & Kershaw (2008)	35% (Reasonable) and 50% (Abs Max)
	Kirchmeyer (2004)	50%
	AVMetrics (2018)	30-32%
	MBA (2019)	30%
	Veros (2017)	10-20% (Top Tier AVM)
+/− 15%	Rossini & Kershaw (2008)	20% (Reasonable) and 35% (Abs Max)
	Kirchmeyer (2004)	30%
	AVMetrics (2018)	20%
+/− 20%	Rossini & Kershaw (2008)	10% (Reasonable) and 20% (Abs Max)
IAAO Metrics		
COV**	Rossini & Kershaw (2008)	13 (Reasonable) and 17 (Abs Max)
COD*	IAAO (2018)	5 to 20 for residential properties
	Rossini & Kershaw (2008)	10 (Reasonable) and 13 (Abs Max)
PRD*	IAAO (2018)	0.98 to 1.03
PRB*	Gloudemans (2011)	Statistically significantly different from zero
	IAAO (2018)	+/− 0.05

*AVM Performance Metric measuring Precision
**AVM Performance Metric measuring both Accuracy and Precision

Figure 1: AVM performance metric thresholds suggested in the literature. [10]

This table can serve as a further target metric threshold for the project. In the paper, each metric from the table is further described and recommendations are made on what further assumptions and checks should be made to properly apply the respective metric.
In their conclusion, they urge AVM developers to especially report the following metrics:

- Mean and median percentage sales error (along with a detailed description of the data of the properties used to produce these metrics)
- Use and report the common definitions for the metrics outlined in the table, especially the FSD and confidence score
- AVM vendors should allow clients to specify their desired confidence level when reporting high/low values or be transparent to the client and inform them about the default FSD's industry range of 68.26%.

Steurer et al. in their research titled "Metrics for evaluating the performance of machine learning based automated valuation models." [9] collected AVM metrics and showcased how the choice of metrics can influence the perceived level of AVM performance. This was showcased on 5 different models varying in their predictive methodology. These 5 models were then first evaluated with commonly utilized performance metrics, afterwards evaluated

with their recommended performance metrics. They go on to further recommend this list of seven performance metrics that according to them adequately evaluate AVM performance.
The following two figures (*Figure 2 & Figure 3*) showcase the used metrics and the performance of the tested models. *Figure 2* shows the performance with commonly used metrics from the literature, *Figure 3* shows the performance with their recommended metrics from their research.

Table 9. Model performance rankings based on metrics from the AVM literature.

Class	Metric	M1	M2	M3	M4	M5
Average Bias	MPE	0.059	−0.029	0.023	**0.017**	0.043
Absolute Difference	MAE/10,000	4.011	**3.624**	3.732	3.763	3.948
Absolute Ratio	MAPE	0.228	**0.195**	0.210	0.208	0.227
Absolute Ratio	COD	0.212	**0.198**	0.208	0.205	0.223
Squared Difference	RMSE/10,000	6.119	**5.214**	5.284	5.463	5.742
Squared Difference	$1\text{-}R^2$	0.448	**0.321**	0.330	0.354	0.391

The prediction methods are as follows: M1 = Linear Regression; M2 = Random Forest; M3 = Multivariate Adaptive Regression Spline; M4 = Quantile Regression with Lasso Penalty; M5 = Neural Network
The displayed metric values are the average values over 5 folds on the testing sets.

Figure 2: Model performance rankings based on metrics from the AVM literature. [9]

Table 10. Predictive performance of methods M1-M5 (Short List).

Class	Metric	M1	M2	M3	M4	M5
Average Bias	LMDPE	0.048	−0.041	**0.007**	0.009	0.008
Absolute Difference	MAE/10,000	4.011	**3.624**	3.732	3.763	3.948
Absolute Ratio	mmMAPE	0.288	**0.269**	0.270	0.273	0.284
Squared Ratio	LRMSE	0.309	0.298	**0.294**	0.300	0.306
Squared Difference	RMSE/10,000	6.119	**5.214**	5.284	5.463	5.742
Percentage Ratio	mmPER(10)	0.697	**0.658**	0.686	0.671	0.703
Quantile	IQRat	0.322	**0.297**	0.327	0.305	0.330

The displayed metric values are the average values over 5 folds on the testing sets.

Figure 3: Predictive performance of methods M1-M5 (Short List). [9]

State-of-the-Art AVM comparison

To get an even clearer picture of what reasonably achievable performance of the to-be-developed AVM for this project would look like, a table of state-of-the-art research was compiled. The table compares multiple research projects utilizing regression and machine learning techniques to create AVM.

As each research project has different access to data and amount of data, model methodology, computational capability and project timeframe, performance metrics and scores on those metrics, the table overview should largely serve as a showcase of the state-of-the-art research and create an outlook for the reader on what could be achieved within this research project.

The main focus of this this table consists of these following entry categories:

Firstly, the used property features in these related works are listed. This list of features can then be used in the development stage as suggested features that could potentially find application in the to be created models, depending on their availability in the data used for the project.

Secondly, the amount of data is described. It would be unreasonable to directly compare the performance of the to-be-developed ML AVM to models that have access to substantially more data. As the timeframe of this project is quite limited, the amount of data that can be organized, cleaned and finally used will be limited as well. Still, the target of this project will be to achieve comparable results.

Thirdly, and perhaps most importantly, the metrics used to evaluate the found related works are listed. In combination with the literature research on appropriate AVM metrics, they give insight in which statistical metrics are needed to evaluate the performance and thus the success of the to-be-developed ML AVM. Besides simply listing the metrics these related works used, the scores on those metrics are listed as well. As all these related works have different specific objectives as their research purpose, even their achieved scores cannot serve as a one to one target for my project, but they serve as an indication of what scores might be reasonably achievable.

Lastly, the type of developed AVM model is listed. Several of the studies compiled in the table compared multiple machine learning models performance or compared the performance of a ML AVM to a simpler regression AVM. By observing which types of ML models tend to perform best and considering the feasibility of developing that type of machine learning model within the projects timeframe, the choice of initial models for this project is made.

Besides these most important entry categories listed within the table, further details such as the name of the study, the property types within the respective AVM, the location of the properties used in the study and the data collection period are listed.

The discussed table can be accessed through the following link:
https://docs.google.com/spreadsheets/d/1vroqZWEXG6q3mvkO5tVoEVExnfbpC1TxQjrU1a_lOUo/edit?usp=sharing

Related work research - Preliminary conclusions

The preliminary conclusions of this related work research section focus on the type of machine learning models that can be used in the development of this project, the metrics and levels of performance suitable for evaluating the AVMs performance and thus the projects success and a table of potentially useful property parameters to be included in the models. These are the fundamental “ingredients” needed for the next phases in the project.

Recommended types of machine learning models:

From the literature and the state-of-the-art research it appears that Random Forest machine learning models consistently achieve good or the best performance within the context of machine learning AVM studies. A variant of a Random Forest model should thus definitely be tested in an initial performance comparison once the data is organized and cleaned.

Secondly, some form of Gradient Boosting model should be initially tested, as there are many different types of specific Gradient Boosting models available and as they achieved varied levels of performance, as seen from the state-of-the-art research. Furthermore, Wishal M Sri Rangan in the previous work related to this project achieved his best results with an AdaBoost model, it could be further considered to initially test multiple types of Gradient Boosting models.

Thirdly, all state-of-the-art research developed some form of baseline regression model. It would be wise to do the same, as it could hint towards a reasonable baseline of performance possible with the available data, it could potentially assist in terms of explainability and serve as a sanity check of the usage of the included property features.

Metrics performance baseline:

Human level performance found through the literature research suggests that the discrepancy between appraised price and subsequent selling price is at around 12%. Even if it is not directly comparable to a statistical metric for AVM, this Human level performance could reasonably be compared to the MAPE metric for AVM evaluation. It could be interpreted as a significant success for the project if a MAPE score below 12% for the best performing to-be-developed machine learning AVM in this project could be achieved.

From the literature research on appropriate AVM metrics and from the state-of-the-art AVM comparison table these following metrics were identified. Some metrics suggested in the AVM performance metrics literature were not clearly enough defined or not used by the identified state-of-the-art researchers, thus a comparison in performance at the end of this project would be difficult. Some metrics such as MAE and RSME were deemed important by

both the literature and the state-of-the-art research, but as they are dependent on the prices of the input properties for the models as well as the used currency of their respective market, a direct comparison cannot be made. Still, they seem to be valuable to report as they still indicate the level of performance directly related to the prediction prices.

The following list of metrics has been compiled to be used to evaluate the performance of the to-be-developed AVM:

- MAE (Mean absolute error)
- MAPE (Mean absolute percentage error)
- RSME (Root mean square error)
- R^2 (Coefficient of determination)
- Adjusted R^2
- PRD (Price-related differential)
- Coefficient of dispersion (COD)

From the compiled table on state-of-the-art research described in the previous subsection "State-of-the-Art AVM comparison", in combination with the results of Wishal M Sri Rangans previous work in his project titled "Creating accurate valuation models for real estate properties", a further table on performance levels of comparable models to the ones chosen for this project was created, which can be seen in *Figure 4* below. As each related work uses different metrics to evaluate their performance, it is incomplete for some of the metrics categories, such as PRD or COD. The purpose of this table is to outline the performance achieved of comparable machine learning models to the ones chosen for this project on the basis of the metrics deemed to be appropriate for this project.

Paper	Model Type	MAE	MAPE	RMSE	R^2	Adjusted R^2	PRD	COD
Yilmazer et al.	Random Forest			approx. 2535.8 €	0.749	0.734	1.030	11.438
Muhammed et al.	XGBoost without GIS enhancement	98505 £	0.469	204625 £	0.434	0.434	1.03	14.92
Muhammed et al.	Random Forest without GIS enhancement	95934 £	0.461	198211 £	0.469	0.469		
Muhammed et al.	Linear Regression without GIS enhancement	113075 £	0.567	227530 £	0.3	0.3		
Muhammed et al.	XGBoost with GIS Feature Enrichment	65367 £	0.311	123585 £	0.791	0.791		
Muhammed et al.	Random Forest with GIS Feature Enrichment	44888 £	0.194	101847 £	0.858	0.858		
Muhammed et al.	Linear Regression with GIS enhancement	101442 £	0.499	200851 £	0.447	0.447		
Wishal M Sri Rangan	Random Forest	47094,3335 €		74331,9899 €	0.6708			
Wishal M Sri Rangan	Gradient-Boosted Tree	53949,8003 €		84402,4863 €	0.5756			
Wishal M Sri Rangan	Adaptive Boosting	36898,4869 €		61172,8420 €	0.7771			
Wishal M Sri Rangan	Rotation Forest	45837,8670 €		78536,8005 €	0.6326			

Figure 4: Table "Performance levels of comparable models on chosen metrics".

The table in combination with the background research in the subsection "State-of-the-Art AVM performance metrics" serves to develop the target baseline for the performance of the models developed in this project. The specific target scores for each identified metric will be stated in the following chapter.

Potentially useful property parameters:

From the literature research on hedonic price models [4] and two papers from the state-of-the-art research table [11], [12], the following table in *Figure 5* of potentially useful property parameters or features was compiled. The table is not ordered in importance of its parameters. It is sorted such that several broader categories can be identified. Its purpose is to assess the potential usefulness of the features and parameters in the data available for the project.

Feature category	Model type	Feature	Study
Geography of the property	ML AVM	Aspect	M. O. Mete & T. Yomralioglu
	ML AVM	Slope	M. O. Mete & T. Yomralioglu
	ML AVM	River View	M. O. Mete & T. Yomralioglu
	ML AVM	Sea View	M. O. Mete & T. Yomralioglu
Proximity to Infrastructure / Amenities	ML AVM	Distance to Bazaar	S. Yilmazer & S. Kocaman
	ML AVM	Distance to College	S. Yilmazer & S. Kocaman
	ML AVM	Distance to Cultural Area	S. Yilmazer & S. Kocaman
	ML AVM	Distance to Garbage collection	S. Yilmazer & S. Kocaman
	ML AVM	Distance to Hospital	S. Yilmazer & S. Kocaman
	ML AVM	Distance to Mall	S. Yilmazer & S. Kocaman
	ML AVM	Distance to Market	S. Yilmazer & S. Kocaman
	ML AVM	Distance to Mosque	S. Yilmazer & S. Kocaman
	ML AVM	Distance to Primary School	S. Yilmazer & S. Kocaman
	ML AVM	Distance to Recreational Area	S. Yilmazer & S. Kocaman
	ML AVM	Distance to University	S. Yilmazer & S. Kocaman
	ML AVM	Proximity to Art Centres	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to City Centres	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Fire Stations	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Green Spaces	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Hospitals	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Other Health Centres	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Libraries	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Museums	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Parks	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Police Centres	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Post Offices	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Primary Education Centres	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Secondary Education Centres	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Further Education Centres	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Universities	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Malls	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Sport Centres	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Touristic Attractions	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Places of Worship	M. O. Mete & T. Yomralioglu
Transport access	ML AVM	Distance to Busstop	S. Yilmazer & S. Kocaman
	ML AVM	Distance to Main Road	S. Yilmazer & S. Kocaman
	ML AVM	Main Road Frontage Or Not	S. Yilmazer & S. Kocaman
	ML AVM	Proximity to Airports	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Bus Stations	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Roads (A Road)	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Roads (B Road)	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Highway Junctions	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Coach Stations	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Ferries	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Passenger Ferries	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Ports	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Subway Stations	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Train Stations	M. O. Mete & T. Yomralioglu
	ML AVM	Proximity to Tram Stations	M. O. Mete & T. Yomralioglu

Figure 5: Table "Property parameters identified from literature & state-of-the-art research"

Feature category	Model type	Feature	Study
Information on the district	Hedonic price model	District / part of town	S. Metzner & A. Kindt
	Hedonic price model	City centre / Central Business District	S. Metzner & A. Kindt
	MLAVM	Distance to Center	S. Yilmazer & S. Kocaman
	MLAVM	Address	M. O. Mete & T. Yomralioglu
Information on the entire building	Hedonic price model	Building size	S. Metzner & A. Kindt
	Hedonic price model	Lot size	S. Metzner & A. Kindt
	Hedonic price model	Type of residence – single family detached house	S. Metzner & A. Kindt
	MLAVM	Is In the Complex Style Or Not	S. Yilmazer & S. Kocaman
	MLAVM	Street Width	S. Yilmazer & S. Kocaman
	MLAVM	Developing status	S. Yilmazer & S. Kocaman
	MLAVM	Number of Facades	S. Yilmazer & S. Kocaman
	MLAVM	Number of Floors	S. Yilmazer & S. Kocaman
	MLAVM	Address	M. O. Mete & T. Yomralioglu
	MLAVM	Property type	M. O. Mete & T. Yomralioglu
	MLAVM	Built form	M. O. Mete & T. Yomralioglu
Information the individual (sub)property	Hedonic price model	Number of bathrooms	S. Metzner & A. Kindt
	Hedonic price model	Number of bedrooms	S. Metzner & A. Kindt
	MLAVM	Balcony size	S. Yilmazer & S. Kocaman
	MLAVM	Number of Baths	S. Yilmazer & S. Kocaman
	MLAVM	Number of Rooms	S. Yilmazer & S. Kocaman
	MLAVM	On Which Floor	S. Yilmazer & S. Kocaman
	MLAVM	Parking or Not	S. Yilmazer & S. Kocaman
	MLAVM	Residence Gross Area	S. Yilmazer & S. Kocaman
	MLAVM	Residence Gross Open Space	S. Yilmazer & S. Kocaman
	MLAVM	Price	M. O. Mete & T. Yomralioglu
	MLAVM	Total floor area	M. O. Mete & T. Yomralioglu
	MLAVM	Floor level	M. O. Mete & T. Yomralioglu
	MLAVM	Extension count	M. O. Mete & T. Yomralioglu
	MLAVM	Number habitable rooms	M. O. Mete & T. Yomralioglu
Amenity charges	MLAVM	Energy consumption current	M. O. Mete & T. Yomralioglu
	MLAVM	Lighting cost current	M. O. Mete & T. Yomralioglu
	MLAVM	Heating cost current	M. O. Mete & T. Yomralioglu
	MLAVM	Hot water cost current	M. O. Mete & T. Yomralioglu
Age / condition of the property	Hedonic price model	Year of construction / building age	S. Metzner & A. Kindt
	Hedonic price model	State of repair / condition	S. Metzner & A. Kindt
	MLAVM	Old / new	M. O. Mete & T. Yomralioglu
Property permits / regulation	MLAVM	Building license	S. Yilmazer & S. Kocaman
	MLAVM	Max Construction Height On A Plot	S. Yilmazer & S. Kocaman
	MLAVM	Occupancy Permit	S. Yilmazer & S. Kocaman
	MLAVM	Date of transfer	M. O. Mete & T. Yomralioglu
	MLAVM	Duration	M. O. Mete & T. Yomralioglu
	MLAVM	Current energy efficiency	M. O. Mete & T. Yomralioglu
	MLAVM	Environment impact current	M. O. Mete & T. Yomralioglu
	MLAVM	CO2 emissions current	M. O. Mete & T. Yomralioglu
"Luxury" on-property amenities	Hedonic price model	Garage (number of garages)	S. Metzner & A. Kindt
	MLAVM	Dependent saloon or not	M. O. Mete & T. Yomralioglu
	MLAVM	Elevator	M. O. Mete & T. Yomralioglu
	MLAVM	Front Garden	M. O. Mete & T. Yomralioglu
	MLAVM	Number of Parking Area	M. O. Mete & T. Yomralioglu
	MLAVM	Side Garden Width	M. O. Mete & T. Yomralioglu
	MLAVM	Swimming Pool	M. O. Mete & T. Yomralioglu

Figure 5 (continued): Table "Property parameters identified from literature & state-of-the-art research"

Overall, as seen in *Figure 5*, some broader categories of features could be identified:

Identified feature categories:

- Geography of the property
- Proximity to Infrastructure / Amenities
- Transport access
- Information on the district
- Information on the entire building
- Information the individual (sub)property
- Amenity charges
- Age / condition of the property
- Property permits / regulation
- “Luxury” on-property amenities

There are likely to be more categories and countlessly more parameters which could be utilized in the development of the ML AVM for this project, especially if agricultural and commercial properties are considered besides residential ones. Overall, the approach the selected state-of-the-art research took was to start out with an extensive list of potential property features and reduce them down according to their statistical relation to the predicted price of the model. The state-of-the-art research also indicated that different features appear to be of different importance depending on the location, thus a clear selection of the to-be-used parameters should not yet be made at this step.

Closing remarks on Chapter 3 - Related Work

The insights gained through the preliminary conclusions drawn from the related work research were adapted to the timeframe and scope of the project throughout the following stages in this report, which follows the timeline of the projects work.

For example, some of the metrics scores achieved by the researchers found in the “State-of-the-art research” table was deemed inappropriate as a comparison to the metrics scores reached with this project. The comparison to some of their models results was deemed inappropriate, as the researchers were able to use more sophisticated machine learning and data transformation techniques to extend upon their machine learning models. Still, the preliminary conclusions hopefully serve as a foundation for the next steps in the project.

Chapter 4 - Ideation & Concept

The aim of the ideation process for this graduation project was to identify the project needs and desired outcomes. Furthermore, its goal was to establish the necessary steps needed in development of the machine learning AVM.

As much of these needs and desired outcomes were identified through the literature and state-of-the-art background research, this chapter largely serves to showcase the step-by-step plan for development and to state the target performance baseline drawn from the literature research and the related work research.

Design process

The usual graduation projects within the Creative Technology study aim to follow the steps outlined in “A Design Process for Creative Technology” by A. Mader and W. Eggink [13].
Projects following this design process first start with a divergent phase, where different ideas for solutions are explored, and later diverge to reduce the explored ideas to a single solution. While moving from divergence to convergence, reflection is taken in between major stages, such that earlier choices can be revisited and no strict stepwise order needs to be followed. Instead of taking one large divergent and one large convergent step, several rounds of divergence and convergence are taken, such that incomplete knowledge at each stage can be supplemented later on.

More concretely, the Creative Technology design process outlined is split into three phases and a evaluation step:
Ideation, Specification, Realisation and finally the Evaluation phase. Each phase requires recurrent divergence and convergence and if new insights are drawn in a later phase, earlier phases can be revisited. The following *Figure 6* more clearly illustrates the stages as a diagram.

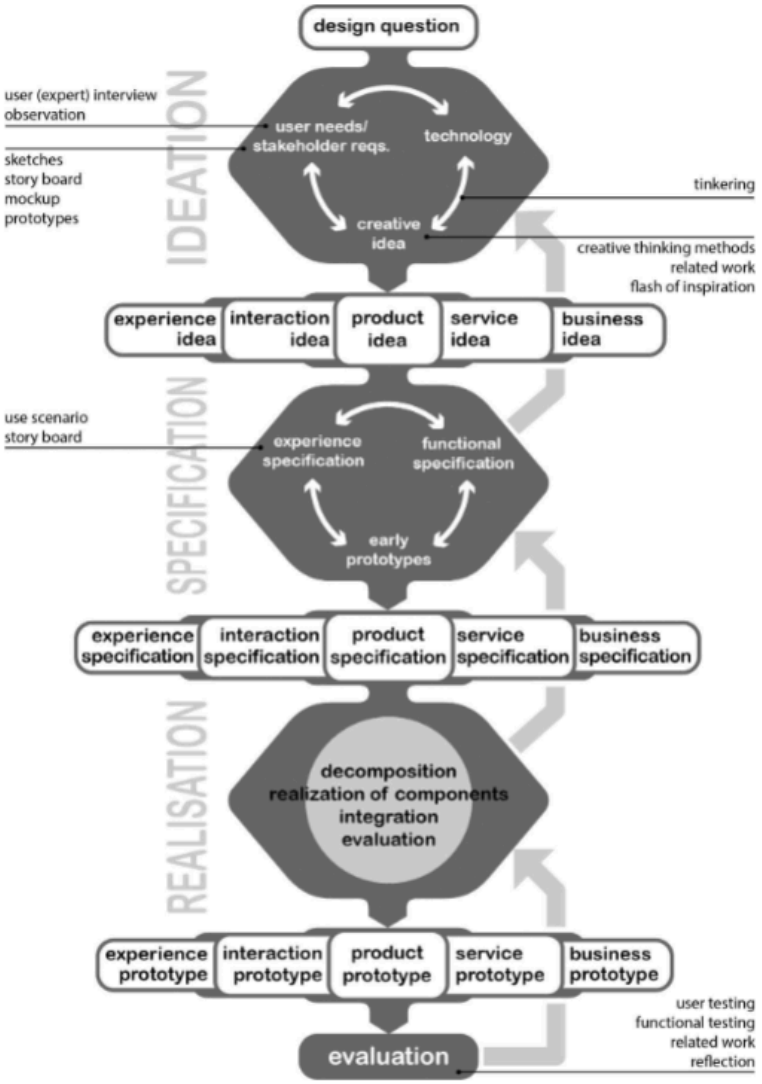

Figure 6: The Creative Technology Design Process (from [13] A. Mader and W. Eggink: "A Design Process for Creative Technology", 2014)

In the context of this project, the general stages are largely followed, with a few considerations specific to the development of machine learning models.

For the ideation stage, the general concept of creating several machine learning AVM was mostly guided by the literature and state of the art research. As the AVM model will be a piece of machine learning software with the goal of returning accurate price predictions on the basis of suitable input data, visual low fidelity prototypes such as commonly prepared paper prototypes or sketches did not seem appropriate in conveying much of the design. Instead, the literature research led to the decisions of which data could be useful in a machine learning AVM, what specific machine learning algorithms could be successfully used and what metrics and performance levels on those metrics would be appropriate to evaluate the model. Furthermore, time in this phase was spent on following two

practice-oriented online courses. The first course clarified the important steps in machine learning development, while the second course showcased different regression machine learning models and the python programming needed to create these regression models. The first objective in the ideation phase for this project then is to create a plan for the necessary steps in developing the specific machine learning models in the context of this project.

In the context of machine learning development, the specification stage and the realisation stage blend into each other. The selected machine learning models will be iteratively trained, tested and evaluated and gradually the amount of data, specific model choice and the hyperparameters will be tuned to reach a better and better performing model. Towards the end of development, the focus will be on improving the best performing candidate model and trying to optimize the final performance of it.
Lastly, the project will still conclude with a final evaluation and reflection on the development of the models, specifically focussing of the best performing model.

Approach

As described in the design process section, the development stages will follow the learned steps from the followed online course on machine learning development.
A flowchart of these steps is show in the following *Figure 7*:

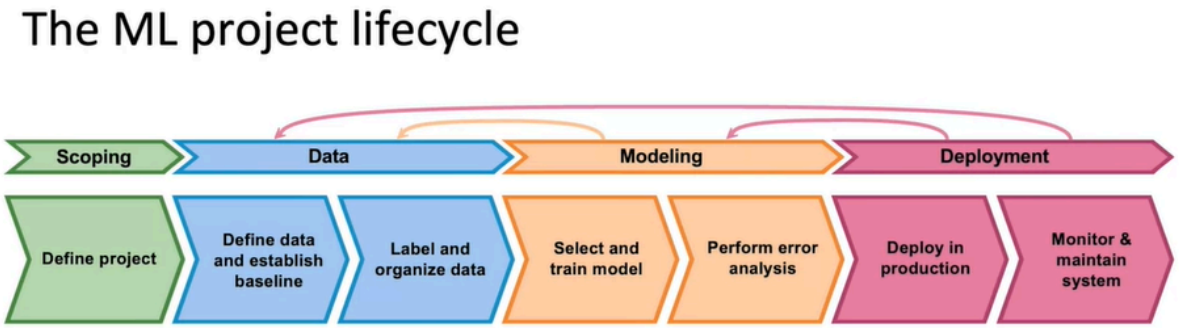

Figure 7: The ML project lifecycle (from Coursera course: Introduction to machine learning in production, Andrew Ng)

In this approach section, the steps in *Figure 7* are briefly explained and the considerations specific to the projects development are outlined. Similarly to the Creative Technology design process, machine learning development is recursive and earlier stages have to be revisited to improve upon the machine learning models performance. Additional data can be incorporated, machine learning model choice can be adapted and hyperparameters for these

machine learning models will need to be tuned and through recurring error analysis new areas of improvement will be identified.

Scoping Stage

Scoping according to the followed course involves the following aspects: First, the project needs to be clearly defined. One needs to determine the data inputs that should lead to a desired machine learning output. The key needed resources and timeline of the development need to be worked out, as well as the key metrics for success of the project. Furthermore, multiple possible solutions should be identified and diligence on the feasibility and value of these solutions should be done. Lastly, some ethical considerations about the social value for the project should be made.

Data Stage

In the data stage, one needs to think about which data could be useful as inputs to reach the target output of the machine learning model. The data stage is split into two substages:

In the first substage, considerations need to be made which specific data features could lead to the desired target output. Furthermore, the amout of data needed should be defined. Lastly, considerations about improving the dataset can already be made such that augmenting or supplementing the data is eased. A good dataset should cover all important input cases, should be consistently defined and labelled, should be timely to the real world and lastly the amount of data should be appropriately sized to the projects need.

In the second substage, the real dataset will be dealt with and improved. The data might first need to be organized and cleaned and be converted and perhaps scaled such that is correctly interpretable by a machine learning model. For example, categorical data should be converted to numerical values or represented through one-hot encoding such that the ML model can utilize it approrpiately. Importantly, not too much time should be spent on creating the perfect dataset, rather one should start with initial tests with the different identified machine learning models. Data provenance and data lineage should also start to be documented such that error analysis in the later stages is facilitated. Lastly, a training / testing split of the data should be done such that the data is balanced and representative of the projects real world context, especially with smaller datasets.

Modeling Stage

The modeling stage is again split into two substages:

First, the several concrete machine learning models need to be selected and trained. An initial performance test should be done to select the most promising models with

overfitting the model on a smaller set of training data. Hyperparameters need to be tuned, compute resources appropriately selected and the outputs should be logged. Furthermore, each version should be logged such that rollback can occur. Throughout the modeling stage, more data can be supplemented and gradually the amount of input data can be increased, while not increasing the amount of input data by 10x at each step.

In the second substage, error analysis is performed. Critically, data should be tagged to find areas in the input data on which the model underperforms. Different versions of the models can then be compared to identify in which data categories improvement occurs between model versions. To conduct successful error analysis experiments, experiment tracking should occur, such that the results can be replicated. The version of the model, the dataset, the set of hyperparameters and the prediction results should be documented for this to be successful. After each test, it is important to prioritize what errors should be improved for the greatest overall increase in model performance and to see how far off the models are from reaching the target performance metrics.

Deployment stage

As this project is a research project and not a real world application, such as a machine learning program for flaw detection through image recognition in a factories production line, the deployment stage in this graduation project is omitted.

Deployment would constitute adapting the machine learning model to the real world scenario outside of the training dataset and iteratively improve it. Furthermore it would mean a constant monitoring and maintaining of the model such that if the input data no longer matches the initial data used in development, the model would still be able to perform.
Instead, in the context of this research project, a final error analysis and evaluation on the identified performance metrics will be performed on the best performing machine learning model to conclude with the project.

These four stages conclude the development process for machine learning models that is followed in the context of this research project. Further practical aspects need to be considered in each stage, they will be documented in the later sections of this paper.

Performance baseline

This section states the target performance baseline for the models developed in this project. How these metrics were derived is described in detail in ‘Chapter 3 - Related work’, subsection ‘Metrics performance baseline’. Here, the respective scores for each metric are set as a target performance baseline. This baseline is taken as the performance goal for the project. Some of the metrics target scores are set at a relatively high level, as they were partially concluded from literature about suggested metrics for professional market AVM.

In the following, each metrics is restated, its formula is shown and the target score is derived:

MAE (Mean absolute error)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left| y_i - x_i \right| \qquad (1)$$

Where n is the number of predictions, y is the predicted property value and x is the actual property value.

MAE measures the average absolute difference between the predicted property values and the actual property values.

As the resulting score for MAE depends on the individual models’ used data, it can not directly be used to compare the project's models to the state-of-the-art models’ results. The results depend on the range of property prices and the used currency. Thus, the main goal for this metric was to iteratively lower the MAE score throughout development and end with a MAE score that was as low as possible.

To make some direct comparison possible, it was further deemed appropriate to compare this project's MAE score for each resulting model to the previous work of Wishal M Sri Rangan best models’ MAE score, as his project was utilizing a dataset of cyprus real estates with likely similar price ranges and the same currency as used in this project. His best performing model reached a MAE of approx. 36898 €, thus the eventual target score for this metric was set to a MAE of less than 36898 €.

MAPE (Mean absolute percentage error)

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left| \frac{x_i - y_i}{x_i} \right| \qquad (2)$$

Where n is the number of predictions, y is the predicted property value and x is the actual property value.

MAPE measures the prediction accuracy of the model. The downside of utilizing MAPE as a metric is that close-to-zero property values lead to a rapid increase in the resulting MAPE score. Still, at this stage of the project, it was decided that it could be utilized to evaluate the project's models, as the literature suggested its usage.

The literature on related work indicated great model performance at a MAPE score of less than 10% and acceptable performance at a MAPE score of less than 15% for commercial AVM.

The related state-of-the-art works models deemed comparable to this projects models reached MAPE scores between 56.7% and 19.4%, as can be seen in *Figure 4: Table "Performance levels of comparable models on chosen metrics"* in 'Chapter 3 - Related work' subsection 'Metrics performance baseline'.

Human level performance appraisal error was deemed approximately comparable to MAPE. Human level performance appraisal error was indicated at around 12%, as stated in 'Chapter 2 - Background Research' subsection 'Human level performance in real estate appraisal'.

From these candidate MAPE baselines, it was decided that a MAPE score of less than 20% would be the project's target score, as it seemed like an appropriate compromise between the suggested scores from the literature on related work and range of scores the state-of-the-art related models.

RMSE (Root mean squared error)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2} \qquad (3)$$

Where n is the number of predictions, y is the predicted property value and x is the actual property value.

RMSE measures the average difference between predicted values and the actual values. It provides an estimate on the models accuracy. Similar to MAE, RMSE depends on the individual models' used data, so again it can not directly be used to compare the project's models to the state-of-the-art models' results. Thus again, the main goal for this metric was to iteratively lower the RMSE score throughout development and end with a RMSE score that was as low as possible.

Again, to make some direct comparison possible, it was deemed appropriate to compare this project's RMSE score for each resulting model to the previous work of Wishal M Sri Rangan best models' RMSE score, with the same reasoning as explained above in subsection MAE. His best performing model reached a RMSE of approx. 61173 €, thus the eventual target score for this metric was set to a RMSE of less than 61173 €.

R² (Coefficient of determination)

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (x_i - y_i)}{\sum_{i=1}^{n} (x_i - \bar{x})} \qquad (4)$$

Where n is the number of predictions, y is the predicted property value, $\bar{x}$ is the mean actual property value and x is the actual property value.

R² or the coefficient of determination is a statistical measure assessing the proportion of variance in the dependent variable (in this case the actual property value) that can be explained by the independent variables in the model (in this case the included property features). It basically explain how well the model fits the data.

The state-of-the-art models' achieved R² scores ranging from 30% (for a less performant regression model) to 85.8%. Wishal M Sri Rangan models' achieved R² scores ranging from 57.56% to 77.71%.

Eventually, a R² score greater than 75% was set as the target baseline score, which was on the relatively optimistic side.

Adjusted R²

$$Adjusted\ R^2 = 1 - \frac{(1 - R^2)(N - 1)}{N - p - 1} \qquad (5)$$

Where R^2 is as calculated above, N is the size of the test sample and p is the number of predictive independent variables.

Adjusted R² is a modified version of R², accounting for the number of predictive independent variables in the model. It adjusts R² to penalize the inclusion of further predictive independent variables that do not contribute significantly to explaining the variance in the dependent variable of the model.

The state-of-the art model's achieved Adjusted R² scores ranging from 30% (again for a less performant regression model) to 85.8%.

Similarly to R² above, an Adjusted R² score greater than 75% was set as the target baseline score, which was similarly on the relatively optimistic side.

PRD (Price-related differential)

$$PRD = Mean\,ratio \,/\, Weighted\,mean\,ratio \quad (6)$$

Where (*Mean ratio*):

$$Mean\,ratio = \frac{1}{n}\sum_{i=1}^{n}\frac{y_i}{x_i} \quad (6.1)$$

And where (*Weighted mean ratio*):

$$Weighted\,mean\,ratio = \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i} \quad (6.2)$$

Where n is the number of predictions, y is the predicted property value and x is the actual property value. (For both equations 6.1 and 6.2)

PRD or the price-related differential is an AVM-specific metric that measures the uniformity in the appraisal of low value and high value properties in the total assessments. A PRD greater than 1 indicates that higher value properties are under-appraised relative to lower value properties. On the other side, a PRD smaller than 1 indicates that higher value properties are over-appraised relative to lower value properties. [14]

The literature on related work stated that PRD should lie between 0.98 and 1.03 to indicate a good level of uniformity in the assessment of low and high value properties for commercial AVM.

Thus, the range from 0.98 to 1.03 for the PRD was set as the target score for this metric.

Coefficient of dispersion (COD)

$$COD = (\,Average\,absolute\,deviation \,/\, Median\,ratio\,) \;*\; 100 \quad (7)$$

Where (*Median ratio*):

$$Median\,ratio = median\,of\,\frac{y}{x}\,over\,all\,predictions \quad (7.1)$$

And where (*Average absolute deviation*):

$$Average\,absolute\,deviation = \frac{\sum_{i=1}^{n}\left|\left(\frac{y_i}{x_i}\right) - Median\,ratio\right|}{n} \quad (7.2)$$

Where n is the number of predictions, y is the predicted property value and x is the actual property value. (For both equations 7.1 and 7.2)

COD or the coefficient of dispersion is another AVM-specific metric that measures the uniformity of appraisals of the model. It specifically measures how far on average each properties ratio of the predicted value over the actual value is away from the median ratio. Simply, it is the average percentage deviation from the median ratio. [14]

The literature on related work states that COD should lie between 5 to 20 for residential properties for commercial AVM. As the to-be-used data did not solely include residential properties, a clear target COD could not be determined on the basis of literature. Still, the chosen target score for COD was set as less than 20 for this metric.

Target performance baseline table

The prior considerations resulted in the following *Table 1* as the target baseline for each of the identified metrics:

MAE	MAPE	RMSE	R^2	Adj. R^2	PRD	COD
< 36898 €	< 20%	< 61173 €	> 75%	> 75%	0.98 to 1.03	< 20

Table 1: *Target performance baseline table*

This table concludes the target baseline for each metric, against which the models developed for the project should be evaluated. This should be done in consideration of the data used and will be done in later sections of this report.

Model choices

Finally, this section briefly states the actually chosen types of machine learning models for the projects realization. Due to the time constraints and scope of the project, just three relatively simple machine learning approaches were finally elected.

The three chosen models to be developed for this project are a multiple linear regression model, a random forest regression model and a gradient boosting model.
Elaboration on these model choices can be found in ‘Chapter 3 - Related work’ subsection ‘Recommended types of machine learning models’.

This concludes the ideation and concept phase of the project, next is the actual realization.

Chapter 5 - Realization

This chapter finally describes the process of realizing the development of the three chosen machine learning AVM models.
First, to be able to create the models, the data stage of the project is recounted, including the data definition and the data preprocessing, resulting with a list of feature categories, which will be the independent variables used by the models to predict the property values.
Second, the actual model development is highlighted. This includes describing the implementation of them in python, as well as fine-tuning the models and input data to increase their performance.
Finally, the evaluation procedure of the models is described. The actual evaluation results are then showcased in the following ‘Chapter 6 - Evaluation’.

Data collection

The initial dataset used in this project was provided by the project supervisor Andreas Kamilaris. It contains information about real estate properties in Cyprus, in the form of an excel spreadsheet. The columns in the dataset contain a mixture of features (predictor variables or independent variables) stemming from the public registry as well as geospatial features provided through Gaea. The initially used response variable (dependent variable), to be later predicted by the models, was the estimated selling price of the property, estimated by the public registry.

This dataset was later extended:

First, to get an improved response variable, the actual selling price of each property was incorporated into the dataset, replacing the estimated selling price as the response variable. The estimated selling price column was only further used to spot potential flaws within the selling price column in the data cleaning phase.

Secondly, the dataset was extended to include the property type of each property as an additional independent variable. Initially, it was suspected that creating a model for the most frequent types of property could lead to improved predictions, but as the dataset did not consistently categorize the property types, this idea was later abandoned. This issue will be further discussed ‘Chapter 7 - Discussion’.

This finally resulted in a dataset with all potential property features included to be used as independent variables, as well as the selling price as the response variable that was to be predicted by the models.

Data preprocessing

The resulting dataset from the data collection stage needs to be cleaned and organized before it can be effectively used to train the selected machine learning models. This pre-processing process is described step-by-step in this section. The initial dataset contained 96 columns and 23675 rows of properties.

Initial feature selection

Through a data review with the project supervisor Andreas Kamilaris, an initial selection of features was made. Some of the columns in the initial dataset were deemed irrelevant or of low value for prediction. Other columns were duplicates, entirely empty or only contained the unit for the adjacent column. These columns were removed from the data.

Removal of rows with formatting errors or large amounts of missing data

Some rows were found to have formatting errors, specifically information about nearby amenities ‘overflowed’ into the next column. As there was a low number of rows where this error occurred (21 properties), these rows were removed from the dataset. Further, some rows (24 properties) had missing data in at least half of the initial columns, they were also removed from the dataset.

Removal of columns with largely empty cells

Several columns contained a majority of empty cells. In some cases, this occurred as the column had a direct relationship to the previous adjacent column. In that case, these columns were kept in the dataset.

In other cases, a column had no apparent relationship to any other columns and was empty for the majority of cells. These columns were removed from the dataset, as imputing the missing values would have likely led to large errors in imputation.

Refined feature selection

Through a second discussion with the supervisor Andreas Kamilaris, it was concluded that some feature columns provided through Gaea likely had low relevance in predicting the property prices, as the properly market was unlikely to consider these factors in the pricing of properties. These columns contained a risk score evaluated through Gaea about different types of natural disasters that could occur in an area, as well as the distance of the property to that natural disaster risk area. These columns were thus removed from the dataset.

Secondly, some of the public registry information was deemed as perfectly multicollinear with other columns. This concerned feature columns containing ID's assigned by the land registry. To the models, some of these ID's translate indirectly to grant information about which properties are in the same municipality, district, etc. As some of the ID's would provide the same information to the models, their columns were also removed from the dataset.

Imputation of empty cells

Some of the columns still contained empty cells at this stage of preprocessing. As the chosen baseline regression model cannot handle null value entries, these empty cells had to be imputed. Filling empty cells with the median value (for numerical variables) or the most frequent value (for categorical variables) was chosen as the imputation strategy. For some columns, imputing with this strategy would be unwise, for example for the column representing the angle of orientation of the property. In this example, taking the median orientation angle would distort the information contained in the column. In those cases, where a clear imputation strategy could not be identified, a filler value was used to indicate missing values to the models.

Imputation of cells with impossible values

Some columns contained values deemed impossible to occur in the real world. For example negative building heights or abnormally large building heights of 9999 meters were identified. These cells were then imputed in similar fashion as done with the empty cells.

Appending of property type & selling price

As mentioned in the section data collection, the property type and selling price of the properties were appended to the original dataset. In the process of the entire data stage, this was done at this step of preprocessing. The columns were retrieved from a secondary, larger excel spreadsheet provided by the supervisor Andreas Kamilaris. This larger spreadsheet contained all the properties from the initial dataset, as well as further properties. The new two columns were merged with the initial dataset by matching on the 'subPropertyID' column between the spreadsheets. This column contained unique identifiers for each property, and was thus ideal for the merging process.

Deleting rows with 0€ selling price

Some of the rows in the response variable 'selling price' contained 0€ selling prices. This leads to errors in some of the metrics, especially MAPE, as its calculation requires a division by the dependent variable. Dividing by 0 would lead to MAPE not being able to be calculated, thus properties with a selling price of 0 were removed from the dataset.

Further preprocessing steps

After the aforementioned preprocessing steps were completed, the resulting dataset contained columns of 30 independent variables and 1 dependent variable. It further contained 23675 rows of properties. Further preprocessing steps were performed within the three programmed machine learning models, as each model required slightly different steps to be able to utilize the data.

Outlier removal via IQR with multiplier 3

To reduce the influence of the properties with extremely large selling prices on the model training, these "outliers" in selling price are removed from the dataset. This was done programmatically, as it was deemed easier to perform through python than through excel.

The Inter-Quartile Range (IQR) is used to remove properties with outlier selling prices. The properties in the dataset with prices higher than the upper bound or lower than the lower bound are removed from the 'selling price' column. In practice, the lower bound was not crossed, thus only properties with selling prices exceeding the upper bound were removed. Equation 8.1 below showcases how the Inter-Quartile Range is calculated.

$$IQR = Q_3 - Q_1 \qquad (8.1)$$

Where Q_3 is the upper quartile (the top 25%) of property prices and Q_1 is the lower quartile (the bottom 25%) of property prices.

Equations 8.2 and 8.3 below showcase how the lower bound (8.2) and the upper bound (8.3) are calculated.

$$Lower\ bound < Q_1 - 3 * IQR \qquad (8.2)$$

$$Upper\ bound > Q_3 + 3 * IQR \qquad (8.3)$$

Where Q_3 is the upper quartile (the top 25%) of property prices, Q_1 is the lower quartile (the bottom 25%) of property prices, and IQR is as calculated in Equation 8.1

The following code snippet in *Figure 8* showcases the outlier removal in the project code:

```
53
54  continuous_columns = ['Selling Price']
55
56  # Calculate the IQR for each specified continuous column
57  Q1 = df[continuous_columns].quantile(0.25)
58  Q3 = df[continuous_columns].quantile(0.75)
59  IQR = Q3 - Q1
60
61  # Identify outliers for specified continuous columns
62  continuous_outliers = ((df[continuous_columns] < (Q1 - 3 * IQR)) | (df[continuous_columns] > (Q3 + 3 * IQR)))
63
64  # Filter the dataset for outliers in specified continuous columns and remove them
65  df = df[~continuous_outliers.any(axis=1)]
66
67  print("Number of rows before IQR:", len_before) #printing the number of properties before outlier removal
68  print("Number of rows after IQR:", len(df)) #printing the number of properties after outlier removal
69  len_after = len(df)
70  rows_removed = len_before - len_after
71  print("Number of rows removed after IQR:", rows_removed) #printing the number removed rows
72
```

Figure 8: Code snippet "Outlier removal on selling price column via IQR with multiplier 3

Method '.quantile()', as seen in line 57 and 58 in *Figure 8*, is a pandas library method for finding sample quantiles in pandas dataframes. [15]

A multiplier of 3 was chosen for the IQR outlier removal, as the selling prices were not normally distributed. Normally, the multiplier is set to 1.5 for IQR outlier detection for normal distributions. The multiplier of 3 was selected through iterative testing of the model and was deemed to be a relatively conservative value for the outlier removal on non-normally distributed selling prices.

The outlier removal resulted in the removal of 957 rows of properties, leaving the dataset with 22718 rows of properties from the previous 23675 rows of properties.

Label encoding of categorical variables

To make the values in the categorical independent variables from the dataset interpretable by the machine learning models, label encoding was performed on the categorical variables. Label encoding converts categorical labels such as text or strings into numerical values representing each category uniquely through an integer value. It was especially done for the multiple linear regression model, as this model requires the input features to be numerical, but it was also applied in the other models for consistency.

In the project, it was performed with the method 'LabelEncoder()' from the sklearn.preprocessing library for python.

One-Hot-Encoding of categorical variables

For the multiple linear regression model, the categorical variables further needed to be One-Hot-Encoded. Categorical variables representation through unique numerical values is not enough for the categorical variables to be correctly interpreted by the multiple linear regression model.

One-Hot-Encoding creates a binary vector for each unique category in a categorical variable column. The length of the binary vector is equal to the number of unique categories in the original categorical variable. The binary vector is filled with value 0 except for the position corresponding to the category of the original variable, which is assigned a value of 1. In simpler terms, One-Hot-Encoding creates a binary table for each categorical variable representing belonging (value 1) or not (value 0) to each category.

This technique is applied, so that the multiple linear regression model doesn't falsely interpret the numerical representation of categorical variables, obtained through label encoding, as having a meaningful magnitude and avoids introducing an ordinal relationship between categories. Basically, it strips the numerical representation of the categorical variable of falsely being interpreted of having a numerical meaning, which could be falsely interpreted by the model.

In the code for the multiple linear regression model, it was implemented via the 'OneHotEncoder()' method from the sklearn.preprocessing library for python. Lastly, to avoid the dummy variable trap, one column created through One-Hot-Encoding was dropped for each categorical variable.

Multicollinearity

Finally, to check for multicollinearity between the independent variables, a correlation matrix was generated. One column ‘QuarterID’ was found to have a correlation coefficient of 1, thus it was removed from the dataset.

The following *Figure 9* showcases the correlation matrix for the XGBoost regression model between the features finally used in all model types, with all multicollinear columns with correlation coefficients of 1 removed. Unfortunately, the bottom labels are slightly cut off, as the python output only provided the image in a format scaled to the screen. They are identical to the labels on the left-hand side and should thus still be interpretable.

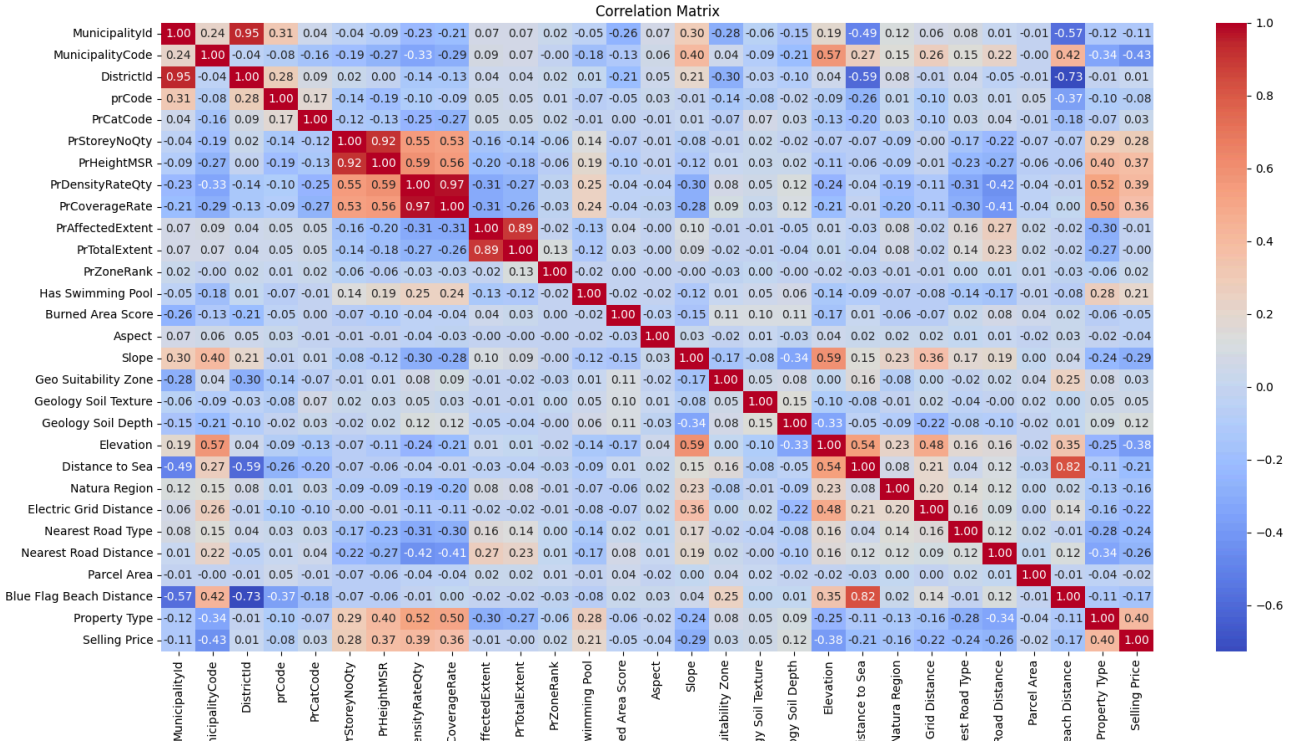

Figure 9: Correlation matrix on the example of the XGBoost regression model

The correlation matrix was generated by utilizing a combination of methods from several libraries.

The method ‘.corr()’ from the pandas library was used to create the correlation matrix, initially for the continuous numerical variables [15]. The method ‘pointbyserialr()’ from the scipy.stats library was used used to transform the categorical variables such that they could be appended to the correlation matrix [16]. The method ‘.heatmap()’ from the seaborn library was used to color-code the correlation matrix according to the correlation coefficient ranges [17]. Finally, the library matplotlib.pyplot was used to display the correlation matrix [18].

Resulting features

The steps taken in the data preprocessing stage resulted in 28 independent variables and 1 dependent variable, the 'selling price', across the models. The following list of resulting features in *Figure 10* showcases the independent variables with a small explanation on what they represent.

Feature Name	Explanation
MunicipalityId	Id representing specific municipality in Cyprus
MunicipalityCode	Code representing specific municipality in Cyprus, different than MunicipalityId
DistrictId	Id representing specific district in Cyprus
prCode	Municipality / Commnunity registry identifier
prCatCode	Secondary Municipality / Community registry identifier
PrStoreyNoQty	Amount of storeys in the building (can be 0 if no building present)
PrHeightMSR	Building height (can be 0 if no building present)
DensityRateQty	Density rate of building on plot
PrCoverageRate	Coverage rate of bulding on plot
PrAffectedExtent	Land registry zoning information, percentage of PrTotalExtent
PrTotalExtent	Land registry zoning information
PrZoneRank	Land registry zoning ranking
Has Swimming Pool	Presence of nearby pwimming pool
Burned Area Score	Information on wether the property previously experienced wildfires (if it did, if the land has recovered or is still affected)
Aspect	Orientation of the property
Slope	Slope of the property
Geo Suitability Zone	Geological suitability of the property
Geology Soil Texture	Soil type of the property
Geology Soil Depth	Depth of the soil type
Elevation	Elevation of the property
Distance to Sea	Distance to the sea
Natura Region	Information on wether the property is in a Natura 2000 region
Electric Grid Distance	Distance to the electric grid
Nearest Road Type	Nearest type of road
Nearest Road Distance	Distance to the nearest road
Parcel Area	The area of the property in sqm
Blue Flag Beach Distance	Distance to nearest Blue Flag Beach, which are beaches awarded with a Blue Flag environmental marker
Property Type	Type of property (e.g. House, Field, Plot etc.)

Figure 10: List of resulting independent variables used across all model types

This concludes the data preprocessing stage, next the implementation of the models in python is elaborated upon.

Specific model selection

As stated at the end of 'Chapter 4 - Ideation & Concept' in section 'Model choices', the three selected models to be implemented in the project are a multiple linear regression model, a random forest regression model and a gradient boosting regression model.
Here, the specific selection of model is described.

Multiple linear regression model

Multiple linear regression is a statistical method used in machine learning to model the relationship between multiple independent variables and a single dependent variable. In the case of the project, it is fit using ordinary least squares, to find the optimal values for the regression coefficients used for each independent variable.
In the project, it serves as the baseline model for experimentation and as a sanity check granting insights into the relationships between the multiple variables in the used dataset.

The multiple linear regression model was implemented through the use of the python library 'sklearn.linear_model'. The regressor was created using the method 'LinearRegression()' from the library. [19]

Random Forest regression model

Random Forest regression is an ensemble learning method used in machine learning for regression tasks. It extends the Random Forest algorithm, which is normally used for classification tasks. For regression tasks, the algorithm is adapted to predict continuous numeric values rather than discrete classes. The algorithm builds multiple decision trees during training and aggregates their individual predictions to a more accurate final prediction.

It utilizes random sampling to train each tree, also known as bagging, to reduce overfitting and increase generalization performance over the entire dataset. Furthermore, at each split of a decision tree, the algorithm randomly selects a subset of features. The randomness introduced in both steps helps decorrelate the individual trees, making the final aggregate prediction more resistant to overfitting and decreasing its variance, thus achieving better performance on unseen data. [21]

The Random Forest regression model was implemented through the use of the python library 'sklearn.ensemble'. The regressor was created using the method 'RandomForestRegressor()' from the library. [20]

Gradient Boosting regression model

Similarly to Random Forest regression, Gradient Boosting regression is an ensemble learning method used in machine learning regression tasks. The algorithm builds a series of weak learners, in our case again decision trees during training. It differs from the tree building in Random Forest by building the trees sequentially, with each new tree aiming to reduce the error of the ensemble of the previous trees. In regression, it aims to minimize the residual error between the predicted values and the actual target values.

In the project, a specific version of Gradient Boosting regression was utilized, namely XGBoost. It was chosen for its popularity stemming from its efficiency, scalability and performance. XGBoost further employs regularization techniques to control model complexity and prevent overfitting. During the testing process, the training speeds of the various iterations on the XGBoost model were considerably faster than the training of the iterations of the Random Forest models. [22]

The XGBoost regression model was implemented through the use of the python library ‘xgboost’. The regressor was created using the method ‘XGBoostRegressor()’ from the library. [23]

Hyperparameter optimization

The Random Forest regression model and the XGBoost regression model both have a set of hyperparameters, that can be tuned to significantly improve the models performance. Hyperparameter tuning aims to find the best set of hyperparameters for the model. Hyperparameters are external configuration parameters that are not learned from the training data, they are configured before the training process.

The hyperparameter optimization strategy chosen for the project is randomized search. Randomized search randomly samples a specified number of hyperparameter combinations from a predefined range of hyperparameter values for each of the hyperparameters chosen for the optimization.

It is implemented for both the Random Forest regression model and XGBoost regression model through the use of the python library ‘sklearn.model_selection’. The method ‘RandomizedSearchCV()’ from the library was used to perform the randomized search to optimize the hyperparameters. The attribute ‘.best_params_’ from the method ‘RandomizedSearchCV()’ stores the best found combination of hyperparameters, which can then be used as the hyperparameters for the model fitting.

As both the implementation of the Random Forest regression model from sklearn and the XGBoost regression model from xgboost automatically try to select good hyperparameters for training, both models were each respectively once trained with automatic hyperparameter tuning and once trained with randomized search hyperparameter tuning.

The resulting performance of both models respectively with automatic hyperparameter optimization and randomized search hyperparameter optimization is evaluated via the identified metrics in ‘Chapter 6 - Evaluation’ subsection ‘Hyperparameter optimization results’.

The following *Figure 11* showcases the hyperparameters selected for optimization of the Random Forest regression model and the XGBoost regression model.

Model	Hyperparameters
Random Forest	number of estimators, maximum features, maximum depth, minimum samples split, minimum samples leaf, bootstrap
XGBoost	learning rate, maximum depth, minimum child weight, subsample, colsample bytree

Figure 11: Table of hyperparameters selected for optimization for Random Forest regression model & XGBoost regression model

Evaluation implementation

To be able to evaluate the different models performance, the selected metrics for evaluation had to be implemented in the python code for each model. The formulae for each of the metrics is stated previously in ‘Chapter 4 - Ideation & Concept’ section ‘Performance baseline’.

The implementation of the metrics MAE, MAPE, RMSE and R^2 could be directly done by utilizing functions from the ‘sklearn.metrics()’ method from the sklearn library [24].

- For the calculation of MAE, function ‘metrics.mean_absolute_error()’ was used.
- For the calculation of MAPE, function ‘metrics.mean_absolute_percentage_error()’ was used.
- For the calculation of RMSE, function ‘metrics.mean_squared_error()’ was used, setting function parameter ‘squared’ to False to retrieve the RMSE instead of the MSE.
- For the calculation of R^2, function ‘metrics.r2_score()’ was used.

The metric Adjusted R^2 needed some further simple calculations to be implemented.

- To calculate Adjusted R^2, function ‘metrics.r2_score()’ was used in further calculation according to the formula for Adjusted R^2, as specified in Equation (5) found in ‘Chapter 4 - Ideation & Concept’, section ‘Performance baseline’, subsection ‘Adjusted R^2’

The metrics PRD and COD are AVM-specific metrics, thus there was no python implementation found for them.

To calculate PRD:

- First, the mean ratio was calculated through simple for-loops, as specified in Equation (6.1) found in ‘Chapter 4 - Ideation & Concept’, section ‘Performance baseline’, subsection ‘PRD’.
- Secondly, the weighted mean ratio was calculated, as specified in Equation (6.2) found in ‘Chapter 4 - Ideation & Concept’, section ‘Performance baseline’, subsection ‘PRD’.
- Finally, the PRD was calculated by dividing the mean ratio by the weighted mean ratio, as specified in Equation (6) found in ‘Chapter 4 - Ideation & Concept’, section ‘Performance baseline’, subsection ‘PRD’.

To calculate COD:

- First, the median ratio was calculated through a simple for-loop, as specified in Equation (7.1) found in ‘Chapter 4 - Ideation & Concept’, section ‘Performance baseline’, subsection ‘COD’.
- Secondly, the average absolute deviation was calculated through a simple for-loop and a divison, as specified in Equation (7.2) found in ‘Chapter 4 - Ideation & Concept’, section ‘Performance baseline’, subsection ‘COD’.
- Finally, the COD was calculated by dividing the average absolute deviation by the median ratio, multiplying the result by 100, as specified in Equation (7) found in ‘Chapter 4 - Ideation & Concept’, section ‘Performance baseline’, subsection ‘COD’.

These implementations of the metrics were then used to evaluate the trained models performance. Throughout development, the metrics were utilized to evaluate the progression

of iterative improvement of the implementation, by tracking the results for each iteration. When certain ideas in development led to worse performance over the metrics, the ideas were revised. If they led to overall improvement over the metrics, they were kept and the results for each metrics for that iteration were noted in a comment section of the code.
The results of the final evaluation after multiple iterations of model improvements are showcased in the following ‘Chapter 6 - Evaluation’

Testing methodology

To test the performance of a machine learning model, commonly the available dataset is split into a training dataset and a testing dataset. The training dataset is used to train the model of choice, the testing dataset is used to evaluate the performance of the trained model on data previously not seen by the model during the training stage. This section describes how this testing was initially implemented through a simple train-test-split of the data. In a later stage of development, the methodology was improved by implementing k-Fold cross validation instead of a simple train-test-split of the data.

Train-test-split

As mentioned, initially the models were tested by implementing a simple train-test-split. This involves splitting the entire dataset available to the models into two subsets, a training set and a testing set. The typically larger training set is used to train the model, and the typically smaller testing set is used for evaluating the models performance on data it has never seen before.

In the project, the dataset was randomly partitioned into a 80% training set and a 20% testing set, a common ratio for train-test-splits appropriate for the relatively large amount of rows in the dataset.
This could directly be achieved in the python code by using the method ‘train_test_split()’ from the ‘sklearn.model_selection’ library [25]. The parameter ‘test_size’ was set to 0.2, equalling 20% for the testing set. The parameter ‘random_state’ was set to the value 0 to achieve a reproducible output across multiple function calls and implementation iterations.

k-Fold cross validation

There are several downsides to utilizing the previously described train-test-split methodology.

First, the performance evaluation on the metrics can be sensitive to how the data is randomly split. A different random split can lead to different metrics results, especially for smaller datasets.

Secondly, in our implementation of the train-test-split, only 80% of the entire dataset is used for training. This utilization of the available training data is suboptimal.

Thirdly, there is a potential to over- or underfitting on patterns or noise in the training dataset, which might make the model not as generalizable for unseen data.

Fourthly, the randomness in splitting the data may lead to biased testing results, especially if patterns in the entire dataset are not represented in either the training set or testing set.

To overcome these limitations, the testing methodology was improved in the final stages of the realization. The methodology was changed to k-Fold cross validation.

k-Fold cross validation is a resampling technique used in machine learning to assess performance and generalization ability of a model. The general idea is to split the entire dataset into k subsets, called folds. The model is then trained k times, each time using k-1 folds for training and the remaining fold for testing. in each iteration, the model uses a different fold as the testing set. At the end of this procedure, the testing results on the metrics for each iteration are averaged.

In the project, k-Fold cross validation was implemented in the following fashion:

1. The rows in the dataset were first shuffled, to eliminate patterns from the ordering of the dataset.
2. Secondly, the dataset was sectioned into k-Folds, with k set equal to 5, a common value for k-Fold cross validation.
3. Thirdly, the above described procedure was applied, resulting in 5 iterations of model training and testing.
4. Finally, the resulting metrics results for each iteration were averaged.

This concludes the k-Fold cross validation testing methodology, and with it ‘Chapter 5 - Realization’. The results of the final evaluation are discussed in the following chapter.

Chapter 6 - Evaluation

This chapter serves to showcase the results of the three models in their respective final iteration. The goal here is to identify the model with the best performance on the identified metrics. Furthermore, the impact of incorporating the geospatial features in the models, retrieved through Gaea, is evaluated. Next, the feature importance overall is evaluated. Afterwards, the impact of the implemented hyperparameter optimization on model performance for the Random Forest regression model and the XGBoost regression model is evaluated. Finally, the best performing model is compared to the target metric baseline, to evaluate wether the set goal of improving upon the state-of-the-art is met or not.

Metrics evaluation

Metrics scores without Gaea features

This subsection showcases the performance for the developed models on the selected metrics, without including the geospatial features retrieved through Gaea. Firstly, the results of the initial train-test-split testing methodology is shown in *Figure 12*, secondly the results of the improved k-Folds cross validation testing methodology is shown in *Figure 13*.
The highlighted yellow cells in the Figures are the best achieved results per metric for the respective testing methodology.

Model Type	MAE	MAPE	RMSE	R^2	Adjusted R^2	PRD	COD
XGBoost with automatically selected Hyperparameters	34497,6445	2,1047	59158,8020	0.65587	0.65579	12.022	196.943
XGBoost with Hyperparameters selected by Random Search	34124,4865	1,8400	58777,0189	0.66029	0.66022	12.739	165.698
Random Forest with automatically selected Hyperparameters	34832,1992	1,7270	60590,7879	0.63901	0.63893	12.490	153.686
Random Forest with Hyperparameters selected by Random Search	34414,3411	1,8643	59197,3824	0.65542	0.65535	13.235	163.352
Multiple Linear Regression	39585,1285	2,6003	66639,1568	0.56334	0.56324	16.121	240.732

Figure 12: Table "Testing results without Gaea features, train-test-split testing methodology"

Model Type	MAE	MAPE	RMSE	R^2	Adjusted R^2	PRD	COD
XGBoost with automatically selected Hyperparameters	33875,6133 €	2,0224	58517,5082 €	0.66329	0.66321	12.675	192.277
XGBoost with Hyperparameters selected by Random Search	31935,9317 €	1,9406	56616,8570 €	0.68481	0.68474	13.558	178.842
Random Forest with automatically selected Hyperparameters	32460,6165 €	2,4831	56850,1159 €	0.68221	0.68213	16.401	215.835
Random Forest with Hyperparameters selected by Random Search	31618,8894 €	2,0066	56269,4274 €	0.68867	0.68859	14.051	179.350
Multiple Linear Regression	39608,4209 €	2,6105	66599,0146 €	0.56387	0.56377	16.157	241.885

Figure 13: Table "Testing results without Gaea features, k-Fold cross validation testing methodology"

As can be seen by observing the highlighted yellow cells in both *Figure 12* and *Figure 13*, there is no one model that consistently performs best across all metrics.

For the first results on the metrics of testing of the models without Gaea features with the initial train-test-split testing methodology, seen in *Figure 12*, the model determined to have the best performance is the XGBoost model with Hyperparameters selected by randomized search. It slightly outperformed the other models in the most amount of metrics, achieving the best results in the metrics MAE, RMSE, R^2 and Adjusted R^2. It came in second place in performance in the metric MAPE, third place in performance in the metric PRD and third place in performance in the metric COD. The exact scores for each metric are not specifically stated here, as they can be seen in *Figure 12* and as there are overall still better performing models yet to come.

For the second results on the metrics of testing of the models without Gaea features with the improved k-Fold cross validation testing methodology, seen in *Figure 13*, the model determined to have the best performance is the Random Forest model with Hyperparameters selected by randomized search. It slightly outperformed the other models in the most amount of metrics, achieving the best results in the metrics MAE, RMSE, R^2 and Adjusted R^2. It came in second place in performance in the metric MAPE, third place in performance in the metric PRD and third place in performance in the metric COD. The exact scores for each metric are again not specifically stated here, as they can be seen in *Figure 13* and as there are overall still better performing models yet to come.

Comparing the two testing methodologies for the models without Gaea features, the k-Fold cross validation testing methodology showed consistently slightly improved performance across the metrics MAE, RMSE, R^2 and Adjusted R^2, across all model types. On the other hand, the k-Fold cross validation testing methodology showed a drop in performance in the metrics MAPE, PRD and COD. The likely reason for this drop in performance will be discussed in the following ‘Chapter 7 - Discussion’.

As the k-Fold cross validation methodology is a more robust testing methodology, its results more accurately reflect the models performance across the entire dataset. The limitations of the initial train-test-split methodology are described in more detail in ‘Chapter 5 - Realization’ section ‘Testing methodology’ subsection ‘k-Folds cross validation’. Thus, when comparing the results at the end to the set target performance baseline, only the k-Folds cross validation results will be analyzed.

Metrics scores with Gaea Features

This subsection showcases the performance for the developed models on the selected metrics, including the geospatial features retrieved through Gaea. Firstly, the results of the initial train-test-split testing methodology is shown in *Figure 14*, secondly the results of the improved k-Folds cross validation testing methodology is shown in *Figure 15*.
Again, the highlighted yellow cells in the Figures are the best achieved results per metric for the respective testing methodology.

Model Type	MAE	MAPE	RMSE	R^2	Adjusted R^2	PRD	COD
XGBoost with automatically selected Hyperparameters	33875,6133 €	2,0224	58517,5082 €	0.66329	0.66321	12.675	192.277
XGBoost with Hyperparameters selected by Random Search	31935,9317 €	1,9406	56616,8570 €	0.68481	0.68474	13.558	178.842
Random Forest with automatically selected Hyperparameters	32460,6165 €	2,4831	56850,1159 €	0.68221	0.68213	16.401	215.835
Random Forest with Hyperparameters selected by Random Search	31618,8894 €	2,0066	56269,4274 €	0.68867	0.68859	14.051	179.350
Multiple Linear Regression	39608,4209 €	2,6105	66599,0146 €	0.56387	0.56377	16.157	241.885

Figure 14: Table "Testing results with Gaea features, train-test-split testing methodology"

Model Type	MAE	MAPE	RMSE	R^2	Adjusted R^2	PRD	COD
XGBoost with automatically selected Hyperparameters	33918,2952	10,3813	57957,9806	0.66996	0.66989	53.555	1003.88
XGBoost with Hyperparameters selected by Random Search	32088,7044	9,0306	55851,4588	0.69348	0.69341	49.045	818.824
Random Forest with automatically selected Hyperparameters	32527,2156	11,5117	56176,9883	0.68994	0.68987	61.459	1007.09
Random Forest with Hyperparameters selected by Random Search	31606,9923	10,3078	55624,0914	0.69597	0.6959	55.606	925.514
Multiple Linear Regression	39931,1625	20,9223	66700,9007	0.56301	0.56291	107.649	1,949.70

Figure 15: Table "Testing results with Gaea features, k-Fold cross validation testing methodology"

For the first results on the metrics of testing of the models with Gaea features with the initial train-test-split testing methodology, seen in *Figure 14*, the model determined to have the best performance is the Random Forest model with Hyperparameters selected by randomized search. It slightly outperformed the other models in the most amount of metrics, achieving the best results in the metrics MAE, RMSE, R^2 and Adjusted R^2. It came in second place in performance in the metric MAPE, third place in performance in the metric PRD and second place in performance in the metric COD.

For the second results on the metrics of testing of the models with Gaea features with the improved k-Fold cross validation testing methodology, seen in *Figure 15*, the model determined to have the best performance is again the Random Forest model with Hyperparameters selected by randomized search. It slightly outperformed the other models in the most amount of metrics, achieving the best results in the metrics MAE, RMSE, R^2 and Adjusted R^2. It came in second place in performance in the metric MAPE, third place in performance in the metric PRD and second place in performance in the metric COD.

Comparing the two testing methodologies for the models with Gaea features, the k-Fold cross validation testing methodology showed a very slightly improved performance across

the metrics MAE, RMSE, R^2 and Adjusted R^2, for the best performant model from each testing methology. On the other hand, the k-Fold cross validation testing methodology showed a significant drop in performance in the metrics MAPE, PRD and COD, when comparing the respective best performing models from the two testing methodologies. The likely reason for this drop in performance will be discussed in the following 'Chapter 7 - Discussion'.

Evaluation of the inclusion of Gaea features

In this section it is evaluated whether the inclusion of the geospatial features, retrieved through Gaea, in the dataset used to train the models was overall effective. For this, the best performing model from the testing results without Gaea features is directly compared to the best performing model from the testing results with Gaea features, both utilizing the k-Fold cross validation testing methodology. *Figure 16* below shows the metrics results for the two best performing models using the k-Folds cross validation testing methodology, once including the Gaea features, once without including the Gaea features. Once more, the highlighted yellow cells in the Figure are the better result between the two best performing models using k-Folds cross validation as the testing methodology.

Model Type	MAE	MAPE	RMSE	R^2	Adjusted R^2	PRD	COD
Random Forest with Hyperparameters selected by Random Search, with Gaea features	31606,9923	10,3078	55624,0914	0.69597	0.6959	55.606	925.514
XGBoost with Hyperparameters selected by Random Search, without Gaea features	34400,2493	7,4968	58744,3015	0.66101	0.66094	40.261	680.677

Figure 16: Table "Comparison of best performing model testing results with Gaea features against best performing model testing results without Gaea features testing, k-Fold cross validation testing methodology".

As can be seen seen in Figure 16, the score for metric MAE for the Random Forest model with Hyperparameters selected by randomized search including the Gaea features is approximately 31607€. The score for metric MAE for the XGBoost model with Hyperparameters selected by randomized search without including the Gaea features is approximately 34400€. As a lower MAE score indicates a better result on the metric, the best model including the Gaea features outperformed the best model without the Gaea features on metric MAE.

The score for metric RMSE for the Random Forest model including the Gaea features is approximately 55624€, the score for metric RMSE for the XGBoost model without including the Gaea features is approximately 58744€. As a lower RMSE score indicates a better result on the metric, the best model including the Gaea features outperformed the best model without the Gaea features on metric RMSE.

The score for metric R^2 for the Random Forest model including the Gaea features is approximately 69.597%, the score for metric R^2 for the XGBoost model without including the Gaea features is approximately 66.101%. As a higher R^2 percentage indicates a better result on the metric, the best model including the Gaea features outperformed the best model without the Gaea features on metric R^2.

The score for metric Adjusted R^2 for the Random Forest model including the Gaea features is approximately 69.560%, the score for metric Adjusted R^2 for the XGBoost model without including the Gaea features is approximately 66.094%. As a higher Adjusted R^2 percentage indicates a better result on the metric, the best model including the Gaea features outperformed the best model without the Gaea features on metric Adjusted R^2.

The score for metric MAPE for the Random Forest model including the Gaea features is 10.3078, which translates to 1030.78%, the score for metric MAPE for the XGBoost model without including the Gaea features is 7.4968, which translates to 749.78%. As a lower MAPE score indicates a better result on the metric, the best model including the Gaea features underperformed compared to the best model without the Gaea features on metric MAPE.

The score for metric PRD for the Random Forest model including the Gaea features is 55.606, the score for metric PRD for the XGBoost model without including the Gaea features is 40.261. As a PRD score close to 1 indicates a better result on the metric, the best model including the Gaea features underperformed compared to the best model without the Gaea features on metric PRD.

The score for metric COD for the Random Forest model including the Gaea features is 925.514, the score for metric COD for the XGBoost model without including the Gaea features is 680.677. As a lower COD score indicates a better result on the metric, the best model including the Gaea features underperformed compared to the best model without the Gaea features on metric COD.

The results are mixed and it is not immediately clear, whether incorporating the Gaea features led to overall improved results. To get a better understanding on which approach resulted in better performance, the *Table 1* target performance baseline table, found in ‘Chapter 4 - Ideation & Concept’, section ‘Performance baseline’ subsection ‘Target performance baseline table’ should be referred to. *Table 1* is once more shown below for easier reference:

MAE	MAPE	RMSE	R^2	Adj. R^2	PRD	COD
< 36898 €	< 20%	< 61173 €	> 75%	> 75%	0.98 to 1.03	< 20

Table 1: *Target performance baseline table*

As can be seen in *Table 1* above, the target baseline score for MAE was a MAE score of less than 36898€. Both best performing models shown in *Figure 16* achieved the target of outperforming the baseline, as their scores are both lower than the target score.

The target baseline score for RMSE was a RMSE score of less than 61173€. Both best performing models shown in *Figure 16* achieved the target of outperforming the baseline, as their scores are both lower than the target score.

The target baseline score for R^2 was a R^2 percentage score of more than 75%. Both best performing models shown in *Figure 16* slightly miss the target. As the target for this metric was set at a relatively high percentage, the scores for both models still indicate relatively decent performance, especially the Random Forest model including the Gaea features, with a R^2 score of approximately 69.597%.

The target baseline score for Adjusted R^2 was an Adjusted R^2 percentage score of more than 75%. Both best performing models shown in *Figure 16* slightly miss the target. Again, as the target for this metric was set at a relatively high percentage, the scores for both models still indicate relatively decent performance, especially the Random Forest model including the Gaea features, with an Adjusted R^2 score of approximately 69.560%.

The target baseline score for MAPE was a MAPE score of less than 0.2, which translates to 20%. Both best performing models shown in *Figure 16* miss the target by a wide margin. The MAPE score for the Random Forest model including the Gaea features is 10.3078, which translates to 1030.78%. The MAPE score for the XGBoost model without the Gaea features is 7.4968, which translates to 749.78%. As both models miss the target by such a wide margin, the reached scores are not a good indication of which model performed better on this metric. This especially will become clear in the next ‘Chapter 7 - Discussion’, where probable reason for the low scores on the MAPE metric are discussed.

The target baseline score for PRD was a PRD score in the range of 0.98 to 1.03. Both best performing models shown in *Figure 16* miss the target by a wide margin. The PRD score for the Random Forest model including the Gaea features is 55.606. The PRD score for the XGBoost model without the Gaea features is 40.261. Again, as both models miss the target by such a wide margin, the reached scores are not a good indication of which model performed better on this metric. This especially will become clear in the next ‘Chapter 7 - Discussion’, where probable reason for the low scores on the PRD metric are discussed.

The target baseline score for COD was a COD score of less than 20. Both best performing models shown in *Figure 16* miss the target by a wide margin. The COD score for the Random Forest model including the Gaea features is 925.514. The PRD score for the XGBoost model without the Gaea features is 680.677. Again, as both models miss the target by such a wide margin, the reached scores are not a good indication of which model

performed better on this metric. This especially will become clear in the next ‘Chapter 7 - Discussion’, where probable reason for the low scores on the COD metric are discussed.

Overall, from this comparison to the Target performance baseline, it can be concluded that there are still some flaws in prediction capability of the models, seen by the scores reached for metrics MAPE, PRD and COD in comparison to the target scores for these metrics.
Still, the Random Forest model including the Gaea features outperformed the XGBoost model not including the Gaea features on all the other metrics MAE, RMSE, R^2 and Adjusted R^2. As these two models were the respectively both best performing models with the testing methodology k-Folds cross validation, it can be concluded that the overall best performing model is likely to be the Random Forest model with its Hyperparameters selected through the randomized search, evaluated on the improved testing methodology of k-Folds cross validation. This indicates that the inclusion of the geospatial features retrieved through Gaea is beneficial for better model performance. This gives the first evidence towards answering the second sub-question of the main research question, which was the following:

How can these features be effectively incorporated into machine learning models to enhance AVM accuracy?

Feature importance

The first sub-question of the main research question was the following:

What are the significant external features, observable through satellite imagery, that influence property values?

This question is answered by evaluating the feature importance. The feature importance is drawn upon the basis of the previously identified overall best performing model, the Random Forest model with its Hyperparameters selected through the randomized search, evaluated on the improved testing methodology of k-Folds cross validation. A feature importance chart can be used to explain which features have greater importance in contributing to the predictive performance of the model.

The feature importance was largely consistent across the k-Folds of testing. Thus, the following *Figure 17* only depicts the feature importance for one of the k-Folds. The objective in this section is to observe the larger trends in feature importance, not the specific ranking of importance of each individual feature, as this slightly varied across the k-Folds.

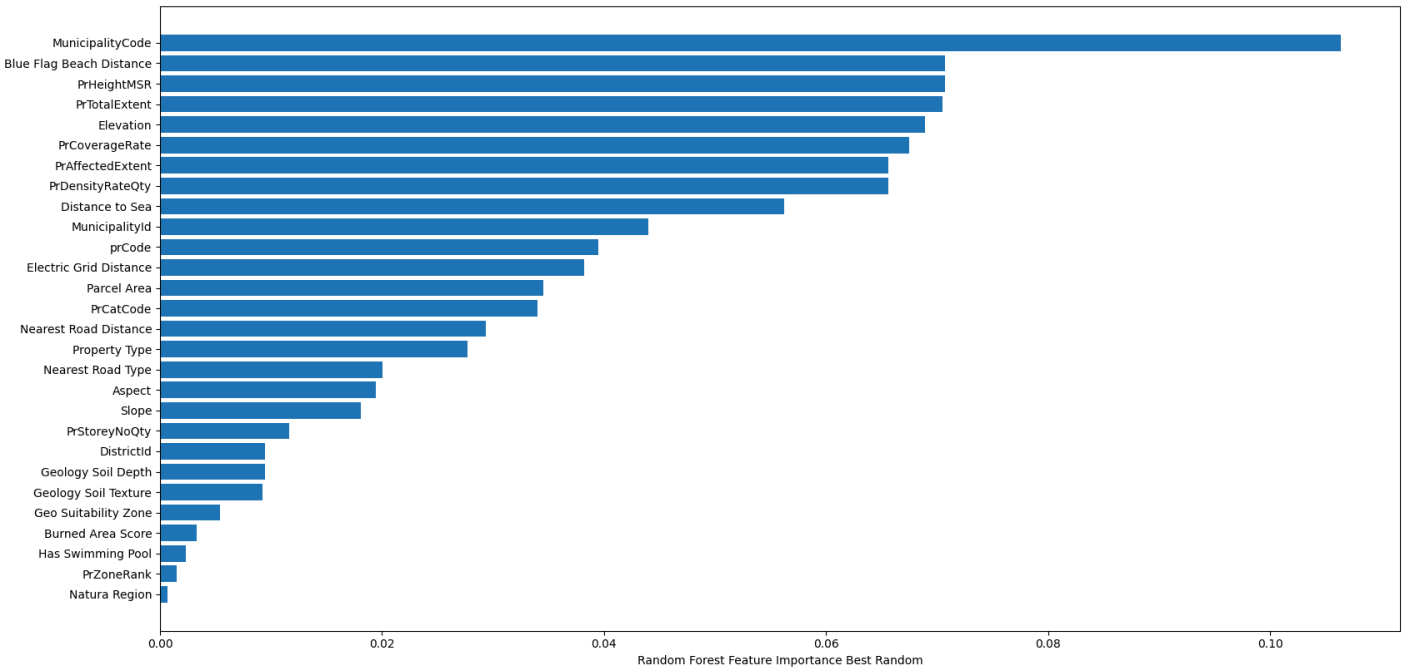

Figure 17: Bar chart "Feature importance of Random Forest model with randomized search Hyperparameter optimization, k-Folds cross validation testing methodology, fold k = 3".

From all the available Gaea features used as independent variables in the dataset, the following 6 were of categorical type:
'Has Swimming Pool'; 'Burned Area Score'; 'Geo Suitability Zone'; 'Geology Soil Texture'; 'Natura Region'; 'Nearest Road Type'.

The other 8 available Gaea features used as independent variables in the dataset were of continuous numerical type:
Aspect'; 'Slope'; 'Geology Soil Depth' ; 'Elevation'; 'Distance to Sea'; 'Electric Grid Distance'; 'Nearest Road Distance'; 'Blue Flag Beach Distance'.

By observing the Gaea features ranking in the feature importance bar chart in *Figure 17*, it can be seen that the continuous numerical independent variables retrieved from Gaea tended to play a more significant role in feature importance than the categorical independent variables retrieved from Gaea.

From all the other available features used as independent variables in the dataset, the following 7 were of categorical type:
‘MunicipalityId’; ‘MunicipalityCode’; ‘DistrictId’; ‘prCode’; ‘PrCatCode’; ‘PrZoneRank’; ‘Property Type’.

The other 7 available features used as independent variables in the dataset were of continuous numerical type:
‘PrStoreyNoQty’; ‘PrHeightMSR’; ‘PrDensityRateQty’; ‘PrCoverageRate’; ‘PrAffectedExtent’; ‘PrTotalExtent’; ‘Parcel Area’.

By observing the feature ranking in the feature importance bar chart in *Figure 17* for the non-Gaea features, the continuous numerical features tended to rank a bit higher than the categorical features, with the clear exception of the categorical feature ‘MunicipalityCode’.

As the feature ‘MunicipalityCode’ grants the models insights into what municipality a property belongs to, and as the reasoning behind property pricing is potentially consistent within a single municipality, it makes sense that this feature can be of great value for predictions. This feature at the same time has the potential to grant information to the model about a properties location and its surrounding properties, potentially allowing the models to find patterns to improve predictions.

Overall, it can still be concluded that the continuous numerical features overall tended to play a larger role in feature importance than the categorical features, except for the categorical features that granted insight into a properties location or relative location towards other properties. It would thus be interesting to incorporate more continuous numerical features into future models.

To answer the sub-research question, the 5 most significant external features, observable through satellite imagery were the following Gaea features (ordered by feature importance):
‘Blue Flag Beach Distance’, ‘Elevation’, ‘Distance to Sea’, ‘Electric Grid Distance’ and ‘Nearest Road Distance’.
The 5 most significant non-Gaea features were the following (ordered by feature importance):
‘MunicipalityCode’, ‘PrHeightMSR’, ‘PrTotalExtent’, ‘PrCoverageRate’ and ‘PrAffectedExtent’;
They could potentially be observed through satellite imagery, but a direct and accurate data source such as the public registry would most likely provide more accurate data.

Hyperparameter optimization results

Finally, the success of the Hyperparameter optimization strategy is evaluated.
From the metrics testing results with the k-Folds cross validation testing methodology for the models containing Gaea features, as seen in *Figure 15* from the previous subsection 'Metrics scores with Gaea features', it can be concluded that the models using the randomized search hyperparameter optimization strategy outperformed their respective counterparts with automatically selected hyperparameters. The randomized search hyperparameter optimization strategy is described in more detail in 'Chapter 5 - Realization' section 'Hyperparameter optimization'.

As the results for the models with randomized search hyperparameter optimization only slightly improved from their respective counterparts with automatically selected hyperparamters, one can conclude that further improvements in the hyperparameter optimization strategy could likely be made. Further potential approaches are described in 'Chapter 8 - Future work'
Figure 18 shows the hyperparameters for each of the k-Folds found through the randomized search hyperparameter optimization for the Random Forest model including Gaea features using k-Folds cross validation as the testing methodology (best overall performing model):

```
1529
1530    Hyperparameters found per k-fold through randomized search:
1531    fold 1: {'n_estimators': 911, 'min_samples_split': 2, 'min_samples_leaf': 2, 'max_features': 4, 'max_depth': 40, 'bootstrap': False}
1532    fold 2: {'n_estimators': 644, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 4, 'max_depth': 30, 'bootstrap': False}
1533    fold 3: {'n_estimators': 911, 'min_samples_split': 2, 'min_samples_leaf': 2, 'max_features': 4, 'max_depth': 40, 'bootstrap': False}
1534    fold 4: {'n_estimators': 911, 'min_samples_split': 2, 'min_samples_leaf': 2, 'max_features': 4, 'max_depth': 40, 'bootstrap': False}
1535    fold 5: {'n_estimators': 377, 'min_samples_split': 5, 'min_samples_leaf': 2, 'max_features': 4, 'max_depth': 70, 'bootstrap': False}
1536
```

Figure 18: Python comment "Identified hyperparameters per k-fold found through randomized search hyperparameter optimization for Random Forest model including Gaea features"

Secondly, for completeness, *Figure 19* shows the hyperparameters for each of the k-Folds found through the randomized search hyperparameter optimization for the XGBoost model including Gaea features using k-Folds cross validation as the testing methodology:

```
1825
1826    Hyperparameters found per k-fold through randomized search:
1827    fold 1: {'subsample': 0.7, 'min_child_weight': 3, 'max_depth': 10, 'learning_rate': 0.03689655172413793, 'gamma': 0.1, 'colsample_bytree': 0.7}
1828    fold 2: {'subsample': 0.7, 'min_child_weight': 4, 'max_depth': 10, 'learning_rate': 0.0503448275862069, 'gamma': 0.0, 'colsample_bytree': 0.6}
1829    fold 3: {'subsample': 0.7, 'min_child_weight': 4, 'max_depth': 10, 'learning_rate': 0.0503448275862069, 'gamma': 0.0, 'colsample_bytree': 0.6}
1830    fold 4: {'subsample': 0.8, 'min_child_weight': 5, 'max_depth': 10, 'learning_rate': 0.03689655172413793, 'gamma': 0.3, 'colsample_bytree': 0.8}
1831    fold 5: {'subsample': 0.7, 'min_child_weight': 4, 'max_depth': 10, 'learning_rate': 0.0503448275862069, 'gamma': 0.0, 'colsample_bytree': 0.6}
1832
```

Figure 19: Python comment "Identified hyperparameters per k-fold found through randomized search hyperparameter optimization for XGBoost model including Gaea features"

Chapter 7 - Discussion

This chapter serves to discuss the potential reasons for why the target performance baseline was only partially met by the best performing model across the specified metrics. It also serves as the basis for the potential areas of improvement that are discussed in the following 'Chapter 8 - Future work'.

Discussion of results

To facilitate the discussion of results, the target performance baseline table as well as the metrics results for the identified best performing model are shown once more below in *Table 1* and *Figure 20* respectively:

MAE	MAPE	RMSE	R^2	Adj. R^2	PRD	COD
< 36898 €	< 20%	< 61173 €	> 75%	> 75%	0.98 to 1.03	< 20

Table 1: *Target performance baseline table*

Model Type	MAE	MAPE	RMSE	R^2	Adjusted R^2	PRD	COD
Random Forest with Hyperparameters selected by Random Search, with Gaea features	31606,9923	10,3078	55624,0914	0.69597	0.6959	55.606	925.514

Figure 20: Table "Best performing model testing results with Gaea features with k-Fold cross validation testing methodology"

In *Figure 20*, the model metrics scores that outperformed the set baseline score are color coded in green, the model metrics scores that slightly underperformed the set baseline score are color coded in yellow and finally the model metrics scores that missed the set baseline score by a wide margin are color coded in red.
As previously touched upon in 'Chapter 6 - Evaluation' section 'Metrics evaluation', the models overall, and here specifically the identified best performing model missed the set target metrics by a wide margin on the metrics MAPE, PRD and COD.
The likely reason for this stems from a combination of the following factors:

The wide price-range of actual sale prices in the dataset used and the inclusion of properties with very low sale prices.

To see why these two factors influence the 3 stated metrics results, one should look at the formulae for each metric, as stated in 'Chapter 4 - Ideation & Concept', section 'Performance baseline'.

Discussion of MAPE

To illustrate the the point made in this subsection, the Formula (2) for the MAPE metric is restated:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{x_i - y_i}{x_i}\right| \qquad (2)$$

Where n is the number of predictions, y is the predicted property value and x is the actual property value.

As can be see seen from Formula (2), within the absolute value bracket, the denominator is the actual property value of property n in the dataset, which in the case of the used dataset is the dependent variable 'property price' we are trying to predict with the models.
When the denominator approaches 0, the absolute percentage error for that individual property becomes very large. As the absolute value of the difference between the predicted property value y (in our case the predicted property price) and the actual property value x (in our case the actual property price) is usually larger than x itself for small property values x, this leads to the absolute value of the entire fraction $\frac{x_i - y_i}{x_i}$ becoming very large.

Normally, this does not necessarily lead to problems, but as the provided dataset included a lot of properties with a very low sale price, the resulting calculated MAPE score increases significantly by these cases occurring relatively frequently in the face of the entire dataset. The wide price-range of the properties in the dataset also play a role here, as all selected models will likely make predictions overpricing the lowest-priced properties, as the models are trained on the entire price-range.
Thus, due to the combination of very low-priced properties (e.g. 1€) in the dataset as well as the large price-range of the property prices in the dataset, the evaluation of the models on MAPE led to results missing the target MAPE score by a wide margin.

Discussion of PRD

Again, to illustrate the the point made in this subsection, the Formula (6) for the PRD metric, the Formula (6.1) for the Mean ratio and Formula (6.2) for the Weighted mean ratio are restated:

$$PRD = Mean\,ratio\ /\ Weighted\,mean\,ratio \qquad (6)$$

Where (*Mean ratio*):

$$Mean\,ratio = \frac{1}{n}\sum_{i=1}^{n}\frac{y_i}{x_i} \qquad (6.1)$$

And where (*Weighted mean ratio*):

$$Weighted\ mean\ ratio\ =\ \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i} \qquad (6.2)$$

Where n is the number of predictions, y is the predicted property value and x is the actual property value. (For both equations 6.1 and 6.2)

Similarly as in the case of MAPE, PRD is indirectly influenced by especially the low-priced properties in the dataset. As can be seen from Formula (6.1) for the mean ratio, it divides by the property value x (in our case the actual property price). Again, when the property value x is near 0 for multiple properties, the mean ratio increases dramatically, as these very low-price properties are commonly at least slightly overpriced by the selected models, as they are fitted on the entire price-range contained in the dataset. The weighted mean ratio seen in Formula (6.2) does not run into this issue as much, as the sum of predicted property values y divided by the sum of actual property values x is still relatively good, as the average prediction quality outperforms the prediction quality for low-price properties. Thus, by dividing a high mean ratio by a relatively lower weighted mean ratio in Formula (6) for the PRD, the evaluation of the models on PRD again led to results missing the target PRD score by a wide margin.

Discussion of COD

Once more, to illustrate the the point made in this subsection, the Formula (7) for the COD metric, the Formula (7.1) for the Median ratio and the Formula (7.2) for the Average absolute deviation are restated:

$$COD\ =\ (\ Average\ absolute\ deviation\ /\ Median\ ratio\ )\ *\ 100 \qquad (7)$$

Where (*Median ratio*):

$$Median\ ratio\ =\ median\ of\ \frac{y}{x}\ over\ all\ predictions \qquad (7.1)$$

And where (*Average absolute deviation*):

$$Average\ absolute\ deviation\ =\ \frac{\sum_{i=1}^{n} \left| \left(\frac{y_i}{x_i}\right) - Median\ ratio \right|}{n} \qquad (7.2)$$

Where n is the number of predictions, y is the predicted property value and x is the actual property value. (For both equations 7.1 and 7.2)

Again, similarly to the cases of MAPE and PRD, the very low-priced properties in the dataset are causing the COD score to increase dramatically. The explanation here solely on the basis of the Formulae (7), (7.1) and (7.2) is unfortunately not perfectly possible, as the median of $\frac{y}{x}$ as calculated in the Formula (7.1) for the Median ratio depends on the used dataset. What can be stated here is that the median ratio should increase when lots of mispricings occur, which especially happens when x the property value (in our case the property price) becomes small. The resulting average absolute deviation increases by a larger factor, as now individual low-priced properties influence the average. This can be seen from the numerator in the Average absolute deviaton Formula (7.2), as bracket $(\frac{y_i}{x_i})$ increases dramatically when x is very low, moreso than the Median ratio increases. Thus, the inclusion of very low-priced properties in the dataset once more lead to worse results of the evaluation of the models on metric COD, leading to results missing the target COD score by a wide margin.

Discussion of remaining metrics and results

Even if it was shown above, that the wide price-range of property prices in the used dataset in combination with the inclusion of several very low-price properties influenced the metrics MAPE, PRD and COD, the other targets were largely met.

As elaborated previously, the metrics R^2 and Adjusted R^2 were set at a relatively high baseline score level, missing them by approximately 5% for the best model can still be seen as relatively successful. The metric baseline target scores set for MAE and RMSE were both outperformed, thus overall the performance of the best developed model, with achieved metrics scores found in *Figure 20,* is still evaluated as relatively successful. As the reasons for the poor performance levels achieved for MAPE, PRD and COD were likely identified, the overall performance is still satisfactory.

With this, we can attempt to answer the final sub-question of the main research question, which is restated here:

How does the developed machine learning AVM compare in performance to state-of-the-art AVMs?

The best performing model and with it the best performing machine learning AVM developed, in the case of the project a Random Forest regression model including geospatial Gaea features with the testing methodology of k-Folds cross validation, is overall comparable in performance to the identified state-of-the-art AVM models. It outperformed the target baseline, derived from the state-of-the-art, on the metrics MAE and RMSE, missed the target baseline scores by approximately 5% on the metrics R^2 and Adjusted R^2 and missed the target baseline scores for the metrics MAPE, PRD and COV by a wide margin.

The likely reasons for why the best performing model did not outperform the set target baseline scores on the metrics MAPE, PRD and COV have been elaborated upon. The following ‘Chapter 8 - Future work’ discusses strategies to improve the model methodology and dataset, to address the likely identified reasons for underperformance on those metrics.

Chapter 8 - Future Work

This chapter first discusses potential approaches on how to deal with the identified issues of the evaluation of the models on the metrics MAPE, PRD and COD. It briefly further showcases two ideas that were experimented upon to further improve the developed models, but were not finalized due to the time-frame limitations of this project. Afterwards, it gives avenues of future work that could be pursued to further extend the methodology used to develop the models realized in this project.

Potential approaches to improve metrics evaluation on MAPE, PRD and COD

As stated in the previous chapter, the wide price-range of properties included in the dataset used by the models developed in this project, as well as the inclusion of very low-priced properties led to underperformance on the metrics MAPE, PRD and COD. Two potential approaches have been ideated, that could improve the performance of the developed models.

First, one could set a lower bound on property prices included in the dataset. If properties priced below e.g. 100€ are not considered in the model, the performance on the metrics MAPE, PRD and COD likely already improves. This strategy is only appropriate if the lower bound on property prices included in the dataset does not exclude a significant fraction of the overall properties from the dataset. Still, this strategy necessarily eliminates a fraction of the ground truth and it might impact performance on unseen data. It can also likely only be effectively applied if unseen data evaluated by the models does not include properties with actual sale prices below the lower bound.

A second approach could be to split the price range up into a subset of the entire price range, e.g. low selling price properties, medium selling price properties and high selling price properties. This could be done by an initial model estimating the sale price and assigning a the estimated price range the property falls into. Further models could then be trained on the specified narrower price ranges, potentially leading to better results. By having a narrower price range for the model to train upon, the models make overall smaller absolute errors in their predictions and the predictions are closer in absolute value to the real prices. For this strategy to effectively work, it is likely that a larger dataset is required.
The categorization in the case of the project could potentially have been achieved by looking at a column initially included in the data. The public registry provided an estimated selling price, and categorization could have potentially been done on the basis of this column.

Besides these two mentioned ideas, are likely further more advanced techniques of outlier detection and model methodology that could also assist in dealing with the identified issue.

Model development per property type

One of the ideas, pursued in the final stages of development, was to train the models solely on the two most frequent types of property, which could be identified through the column property type appended to the original dataset. Usually, AVM are developed for a single type of property, most frequently residential properties. Oftentimes, they further are specialized on residential houses or apartments.
The two most frequent property classification categories were ‘Plot’ with 7084 rows and ‘Field’ with 16344 rows out of a total of the initial 23675 rows contained in the dataset. ‘Plot’ and ‘Field’ combined accounted for 23428 out of 23675 rows, thus representing the vast majority of property types.
Unfortunately, the classification of property type done by the public registry was not very detailed, it is likely that the classification was made for the purpose of the sale transaction.
It did not fully capture what exactly the property contained, as many of the ‘Fields’ or ‘Plots’ contained some sort of buildings, such as houses or apartments, which was evident from other columns such as the ‘PrHeightMSR’ column, which represents the building height.
Thus, the idea of developing models specialized on the property type was abandoned. If the dataset could be extended with more accurate property type classification, this could likely still lead to improvements in performance of the models.

Inclusion of nearby amenities

A second idea that was pursued in the final stages of development was to incorporate a column from the original dataset called ‘Nearby amenities’. It contained information about a large number of categories of nearby amenities, such as hospitals, airports, cafes, schools, malls etc. It further contained the distance from each property to each amenity related to the property. In theory, this nearby amenity data would likely had a impact on the actual price of properties. Unfortunately, categorization of amenities was imprecise, leading to many amenities with similar but distinct categorization. An example of this would be all various types of nearby restaurants, each categorized as e.g. ‘Lebanese restaurant’ or ‘Indian restaurant’. Furthermore, many amenities had categories that only applied to that specific amenity. Thirdly, many amenities had multiple categories assigned to them at once, making a clear categorization extremely difficult.
Besides the issue of categorization, the formatting of the data was stored in a complicated JSON-like structure. This meant that a lot of programming time first had to be spent to

properly reassign each amenity to the corresponding property, but this was eventually achieved. Only at that point the described issues associated with the categorization became apparent.

In the time frame of the project, a re-assignment to a simplified categorization was attempted, but likely had some flaws in its re-categorization. To properly re-categorize each amenity, one would have needed to look at each amenity with the help of services such as google maps, to see which simplified categorization would be most appropriate.

Finally, the last issue that became apparent was that the majority of properties contained in the initial dataset had no amenities associated with them. It is likely that in actuality there would be nearby amenities present, but they were not contained in the dataset. Imputing nearby amenities appeared impossible and would lead to a large distortion of the real world data. The simplified recategorization was as an experiment appended to the dataset used by the models, but the results worsened, as the re-categorization and large amount of missing data likely did not help the models in making predictions.

Thus, finally, the idea was also abandoned.

It is likely that accurate information on nearby amenities with a relatively simple categorization and fewer categories could have improved the models. If one has access to more complete data on more precisely categorized nearby amenities, it could likely be implemented in a future machine learning AVM.

This concludes the two approaches, that were experimented with in the final stages of the project. They could still be promising ideas, but were not feasible to be implemented with the given data and the limited timeframe of the project. Lastly, some more general future work recommendations are made.

General future work recommendations

First of all, it would likely be a good idea to incorporate a larger dataset into the model development process. Specifically, a larger and more specific range of property types could likely improve the models developed in this project.

Secondly, more numerical continuous features of any kind could be utilized by the models, as they showed the biggest feature importance. Categories of features that could be utilized are further described in ‘Chapter 3 - Related work’ section ‘Related work research - Preliminary conclusions’ subsection ‘Potentially useful property parameters’.

Further parameters on the real estate market and financial conditions could likely be effectively used. A indication of the transaction date, to adjust the sale prices for inflation

could also assist in prediction quality. Finally, data on the historical transactions that occurred could potentially further improve the models predictions, if appropriately implemented.

The imputation methodology utilized could further be improved. Strategies such as K-NN could be tried as a starting point, but there are many imputation methods that could be tried.

The used methodology of randomized search could be extended by utilizing grid search in combination with it. In that case, randomized search could narrow down the parameters to be then used as the starting point for the grid search.

Finally, more sophisticated machine learning approaches could be tried. A clear recommendation is not made here, further background research should be done to identify the most promising models.

Chapter 9 - Conclusion

The graduation project, "Leveraging Machine Learning and Geo-Analytics in Automatic Valuation Models to advance Real Estate Valuation" aims to enhance existing AVM methodology. Traditional AVM methodology often omits incorporating the properties surrounding environment. By leveraging data from a digital twin of the Cyprus real estate market, this project aimed to extend current methodologies. The main goal was to create a machine learning AVM that predicts property prices more accurately than the state-of-the-art by incorporating comprehensive information about the surroundings, retrieved through satellite data, improving upon existing state-of-the-art models.

The research aimed to first identify the significant external features observable through satellite imagery, then tried to effectively incorporate them into three different machine learning AVM and finally aimed to compare the best performing model to the state-of-the-art.

The three developed machine learning models are a multiple linear regression model, serving as a baseline model, a Random Forest regression model and a version of a Gradient Boosting Regression model, specifically XGBoost, implemented in python.

Results from the best performing model indicated mixed levels of success, outperforming the target baseline, derived from the state-of-the-art research, in the metrics MAE and RMSE, slightly underperforming the target baseline on the metrics R^2 and Adjusted R^2 and finally significantly underperforming the set target baseline on the metrics MAPE, PRD and COD. The reasons behind the missed level of performance compared to the set target baseline in the metrics MAPE, PRD and COD are likely related to the utilized dataset, as very low-priced properties in combination with the wide price range across the entire dataset interfered with these metrics specifically.

The developed machine learning model achieving the overall best performance is the Random Forest regression model including the external features observable through satellite imagery, with hyperparameter optimization through the use of randomized search.

Overall, the best models' performance is still evaluated as relatively successful, recommendations are made to address the issues that created the underperformance in the metrics MAPE, PRD and COD.

Hopefully, the developed model methodology can be further improved through the suggested future research avenues, as it has the potential to enable individual property buyers and sellers access to relatively accurate Real estate valuations, which can currently commonly only be obtained through the purchase of expensive professional AVM models or the services they provide.

References

[1] M. Renigier-Biłozor, A. Janowski, and M. d'Amato, "Automated Valuation Model based on fuzzy and rough set theory for real estate market with insufficient source data," Land Use Policy, vol. 87, p. 104021, Sep. 2019, doi: https://doi.org/10.1016/j.landusepol.2019.104021.

[2] Hilgers, B. A. J. (2018). Automated valuation models for commercial real estate in the Netherlands traditional regression versus machine learning techniques, Aug. 2018 https://pure.tue.nl/ws/portalfiles/portal/122584912/Hilgers_0831665.pdf

[3] S. E. Cannon and R. A. Cole, "How Accurate are Commercial Real Estate Appraisals? Evidence from 25 Years of NCREIF Sales Data," SSRN Electronic Journal, 2011, doi: https://doi.org/10.2139/ssrn.1824807.

[4] S. Metzner and A. Kindt, "Determination of the parameters of automated valuation models for the hedonic property valuation of residential properties," International Journal of Housing Markets and Analysis, vol. 11, no. 1, pp. 73–100, Feb. 2018, doi: https://doi.org/10.1108/ijhma-02-2017-0018.

[5] W. M Sri Rangan, "Creating accurate valuation models for real estate properties," essay.utwente.nl, Feb. 03, 2023. http://essay.utwente.nl/94361/ (accessed Feb. 16, 2024).

[6] "CYENS SuPerWorld Research Group," superworld.cyens.org.cy. https://superworld.cyens.org.cy/product3.html (accessed Feb. 16, 2024).

[7] A. Jamil, C. Padubidri, S. Karatsiolis, I. Kalita, A. Guley and A. Kamilaris, "GAEA - A Country-Scale Geospatial Environmental Modelling Tool: Towards a Digital Twin for Real Estate", Proceedings of the 37th edition of Environmental Informatics (EnviroInfo 2023), Oct. 2023. https://superworld.cyens.org.cy/papers/EnviroInfo23_paper_1790.pdf (accessed Feb. 16, 2024).

[8] "CYENS SuPerWorld Research Group," superworld.cyens.org.cy. https://superworld.cyens.org.cy/product1.html (accessed Feb. 16, 2024).

[9] M. Steurer, R. J. Hill, and N. Pfeifer, "Metrics for evaluating the performance of machine learning based automated valuation models," Journal of Property Research, vol. 38, no. 2, pp. 99–129, Apr. 2021, doi: https://doi.org/10.1080/09599916.2020.1858937.

[10] M. Ecker, H. Isakson, and L. Kennedy, "An Exposition of AVM Performance Metrics," Journal of Real Estate Practice and Education, vol. 22, no. 1, pp. 22–39, Jan. 2020, doi: https://doi.org/10.1080/15214842.2020.1757352.

[11] S. Yilmazer and S. Kocaman, "A mass appraisal assessment study using machine learning based on multiple regression and random forest," Land Use Policy, vol. 99, p. 104889, Dec. 2020, doi: https://doi.org/10.1016/j.landusepol.2020.104889.

[12] M. O. Mete and T. Yomralioglu, "A Hybrid Approach for Mass Valuation of Residential Properties through Geographic Information Systems and Machine Learning Integration," Geographical Analysis, Oct. 2022, doi: https://doi.org/10.1111/gean.12350.

[13] A. Mader and W. Eggink, "A Design Process for Creative Technology," DS 78: Proceedings of the 16th International conference on Engineering and Product Design Education (E&PDE14), Design Education and Human Technology Relations, University of Twente, The Netherlands, 04-05.09.2014,

pp. 568–573, 2014, Available:
https://www.designsociety.org/publication/35942/a_design_process_for_creative_technology

[14] R. Peterson and T. Carter, “Measuring Real Property Appraisal Performance In Washington’s Property Tax System 2009,” 2010. Accessed: Feb. 15, 2024. [Online]. Available: https://leg.wa.gov/House/Committees/FIN/Documents/2009/RatioText.pdf

[15] Pandas, “Python Data Analysis Library — pandas: Python Data Analysis Library,” Pydata.org, 2018. https://pandas.pydata.org/

[16] “Statistical functions (scipy.stats) — SciPy v1.3.3 Reference Guide,” Scipy.org, 2019. https://docs.scipy.org/doc/scipy/reference/stats.html

[17] “seaborn.heatmap — seaborn 0.10.1 documentation,” seaborn.pydata.org. https://seaborn.pydata.org/generated/seaborn.heatmap.html

[18] “matplotlib.pyplot — Matplotlib 3.5.3 documentation,” matplotlib.org. https://matplotlib.org/3.5.3/api/_as_gen/matplotlib.pyplot.html

[19] “1.1. Linear Models — scikit-learn 0.23.2 documentation,” scikit-learn.org. https://scikit-learn.org/stable/modules/linear_model.html#

[20] scikit-learn, “3.2.4.3.2. sklearn.ensemble.RandomForestRegressor — scikit-learn 0.20.3 documentation,” Scikit-learn.org, 2018. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

[21] “1.11. Ensembles: Gradient boosting, random forests, bagging, voting, stacking,” scikit-learn. https://scikit-learn.org/stable/modules/ensemble.html#random-forests-and-other-randomized-tree-ensembles

[22] xgboost developers, “XGBoost Documentation — xgboost 1.5.1 documentation,” xgboost.readthedocs.io, 2022. https://xgboost.readthedocs.io/en/stable/

[23] “Python Package Introduction — xgboost 1.5.2 documentation,” xgboost.readthedocs.io. https://xgboost.readthedocs.io/en/stable/python/python_intro.html

[24] “3.3. Metrics and scoring: quantifying the quality of predictions — scikit-learn 0.22.1 documentation,” scikit-learn.org. https://scikit-learn.org/stable/modules/model_evaluation.html

[25] scikit-learn, “sklearn.model_selection.train_test_split — scikit-learn 0.20.3 documentation,” Scikit-learn.org, 2018. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html