CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Exploring Ensemble Learning Techniques for Ranking Project Proposals at HILTI

Master's thesis in Computer science and engineering

Filip Folkesson
Rojan Hashemi

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024

Master's thesis 2024

Exploring Ensemble Learning Techniques for Ranking Project Proposals at HILTI

Filip Folkesson
Rojan Hashemi

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2024

Exploring Ensemble Learning Techniques for Ranking Project Proposals at HILTI
Filip Folkesson
Rojan Hashemi

Supervisor: Felix Cherubini, Department of Computer Science and Engineering
Advisors: Haitham Elfaham, Johann Stadler, HILTI AG
Examiner: Krasimir Angelov, Department of Computer Science and Engineering

Master's Thesis 2024 Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2024

Exploring Ensemble Learning Techniques for Ranking Project Proposals at HILTI
Filip Folkesson, Rojan Hashemi
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Ensemble learning is a class of machine learning techniques that utilizes an ensemble of models to make a prediction. For an ensemble model to perform well it is important that the models in the ensemble are diverse, i.e. they make errors in different places. There are many different methods to achieve diverse ensembles which are being evaluated in this paper based on how well they can predict future sales of construction projects using data given in the project proposal. Furthermore, SHAP values will be used to explain the model in order to increase understanding of what the most important variables are. A new technique to determine the prediction interval for different parts of the input space for risk assessment is also presented.

Keywords: Ensemble learning, ensemble regression, variable importance, sales prediction

Acknowledgements

We would like to extend our deepest gratitude to all those who have supported us throughout our thesis project.

Firstly, we express our sincere thanks to our advisors at HILTI, Johann Stadler and Haitham Elfaham. Their expertise and experience on the topic, along with their guidance, provided us with invaluable support and the opportunity to work on this project.

We are also profoundly grateful to our supervisor, Felix Cherubini. He has supported us throughout the project, offering valuable advice with great attention to detail. His insight and guidance have been instrumental in the successful completion of our thesis.

We would also like to thank our examiner, Krasimir Angelov, for his dedicated time and effort in reviewing and evaluating our work. His feedback and evaluation have been essential to the refinement and finalization of our thesis.

Finally, we want to express our appreciation to our family and friends for their great support and encouragement throughout this journey.

Filip Folkesson & Rojan Hashemi, Gothenburg, 2024-08-07

Acronyms

Bagging	Bootstrap aggregating
EL	Ensemble learning
EN	Elastic net
GB	Gradient boost
HRP	HILTI relevant potential
LGBM	Light gradient boosting machine
MAPE	Mean absolute percentage error
MAE	Mean absolute error
ML	Machine learning
NS	Net sale
Project	Completed project at HILTI that has been evaluated
(Project) Proposal	Proposal of a project that will potentially be chosen by HILTI
RF	Random forest
SHAP	Shapley additive explanations
SVM	Support vector machine
XGB	Extreme gradient boost

Contents

List of Figures

List of Tables

1

Introduction

In recent years there has been a trend away from mass production towards mass customization in the manufacturing industry[1]. This has led to the manufacturing company HILTI receiving a large number of proposals for customer-specific projects. However, HILTI does not have the capacity to undertake all of them and some of the projects might be seen as risky or not worth the time and effort. It is therefore reasonable to expect that improving the selection-process is important to ensure that resources are being allocated to the projects with the highest expected sales.

Currently, a project manager manually selects projects from a large list of *(project) proposals*, with the help of an initial score, which is calculated using ad-hoc rules. With this master's thesis, we aim to develop a method to give the projects a more accurate initial score by using a sophisticated *ensemble learning* (EL) model.

Ensemble learning is a machine learning technique, which combines a set of weak learners to create a strong learner. It can be likened to a jury making a decision together, as opposed to, to a single person doing the decision alone. It is reasonable to assume that a jury will have a diverse set of knowledge and is thus more likely to come to the correct decision than the single person is.

Ensemble learning has been shown to outperform traditional machine learning techniques when it comes to high-dimensional and noisy data[2].

While ensemble learning is not a new concept, we can see that in recent years the popularity of ensemble learning has increased dramatically as depicted in Figure 1.1[3].

It should be noted that it is unreasonable to expect a model that can predict future sales with high accuracy. This is partly due to the quick and vast changes in the global geopolitical landscape; not having all variables at hand; and general randomness, which will obfuscate the predictions. It is for example reasonable to expect that two projects with the exact same parameters will not yield the same amount of sales. The model should, however, be able to aid the manual selection process and give insight into what parameters are the most significant in predicting sales.

One aspect of machine learning, which is often overlooked, is explainability. In other words, why does the model behave the way it does, and what data features are most important to the model. This is important in order to understand what the model is actually showing and how to make further deductions of the problem.

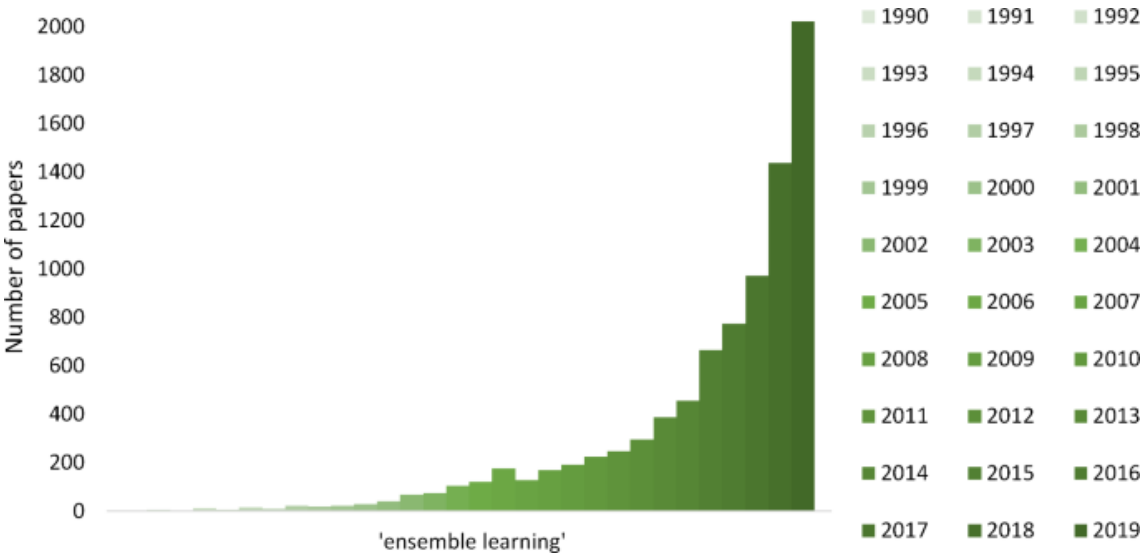

Figure 1.1: Papers on EL per year[3].

1.1 Thesis Question

With this thesis we aim to further increase the scientific understanding of ensemble techniques, in particular *why, how* and *when* certain models work best; a question that is crucial for the increased effectiveness of ensemble learning[4].

We are also exploring the usefulness of a novel uncertainty prediction heuristic, which is further described in Section 2.6, and how important the different data features are to the model, further described in Section 2.5.

2

Theory

In this section the underlying theory of ensemble learning and other relevant machine learning concepts are described. It should be noted that only ensemble learning in a regression context is discussed, thus omitting literature regarding classification and ranking. It should also be noted that the terms regressor, learner and model are used interchangeably in the context of this thesis.

2.1 Ensemble Learning Principles

The meaning of EL is to combine the results of multiple *weak classifiers* to create a stronger classifier using various techniques. There are three important concepts in EL, namely: diversity, training, and the combining of classifiers.

2.1.1 Diversity

If we have a jury, consisting of members who all vote the same all the time, then there would be no reason to have a jury at all instead of a single decision-maker. The same principle applies to the concept of ensemble learning. In order to reap the benefits of an ensemble learner it is crucial that the individual learners within the ensemble produce different, yet accurate predictions.

Mathematically, diversity can be described using the *Ambiguity Decomposition*[4][5].

$$(f_{ens} - d)^2 = \sum_i w_i (f_i - d)^2 - \sum_i w_i (f_{ens} - f_i)^2 \tag{2.1}$$

where,

$$f_{ens} = \sum_i w_i f_i \tag{2.2}$$

d is the target value; and where f_i is the predicted value of the i-th learner; and w_i is the weight of that learner, where the following equation holds:

$$\sum_i w_i = 1 \tag{2.3}$$

The equation states that the squared error of the ensemble, where each of the learners are equally weighted, can be decomposed into a *bias* term and a *ambiguity* term. The *bias* term:

$$\sum_i w_i(f_i - d)^2 \tag{2.4}$$

is the average of the squared error of each individual learners. The *ambiguity* term:

$$\sum_i w_i(f_{ens} - f_i)^2 \tag{2.5}$$

is a measure of how much the individual learners differ from varies on average from eachother.

What can be observed in equation 2.1 is that the more ambiguity, i.e. the more each individual learner f_i differs from the ensemble f_{ens}, the smaller error we get. However, if the ambiguity is increased, that means that the bias will also increase due to the nature of the squared error measure. Therefore, a trade-off between diversity and individual accuracy must be made.

With the mathematical understanding of why diversity is desirable, we can now approach how to achieve diversity among the learners. Diversity can be achieved by utilizing many different methods, which are usually categorized as *implicit* and *explicit*. Implicit methods, such as Bagging (see Section 2.2.1), rely on randomness to create diversity, whereas the explicit methods, such as Boosting (see Section 2.2.2), are deterministic in nature.

2.1.2 Training

Training is the concept of how to train the individual classifiers. There are two main techniques for training the classifiers, namely sequentially and in parallel. During sequential training, the classifiers are trained one by one. Here, a classifier focuses on fixing the errors made by the previous classifiers by skewing the dataset for the classifier to mainly include misclassified instances. This is contrasted by parallel training, whereby the classifiers are trained in parallel and thus there is no *data dependency* between them[6].

It is further important to have a good model for training the classifiers to prevent *overfitting* or *underfitting*. There are various sampling methods used for generating a training and a testing set from the available learning data. The methods differ in how the training and testing sets are chosen. In every case the training set is used to build the model and the testing set is used for evaluating the model. The most basic sampling method is *single hold-out random subsampling*. Here the training and testing data are randomly sampled but their size is set, e.g. 70% for training and 30% for testing. This method is useful when the size of the learning set is large enough that both training and test sets are large. In other cases, different methods of resampling the learning set into different training and test sets are used to provide

more data for training and multiple evaluations of the model, resulting in a better fitted model. *K-fold cross-validation* utilizes this technique.

In K-fold cross validation the learning set is split in k disjoint subsets S_1 ... S_k, each called a *fold*, of approximately the same size. In each iteration of the learning algorithm, the union of all but one of the subsets make up the training set and the remaining subset is the test set. The performance is given by the average of all k evaluations[7]. Increasing the number of folds means that the model has more data to train on, thus giving a better model. Also testing the model on more data ensures that the model is more thoroughly tested. For example, if we were not using cross validation and all the outliers happened to fall into the test data, the evaluation of the model would not give an accurate representation of the models quality. However increasing k also increases the computational complexity which means that a balance between a high enough, but not too high k needs to be found to properly train and test the model.

2.1.3 Combination of Regressors

One of the key aspects of ensemble learning is to combine the results of the individual learners. In the regression context this can be done in a myriad of different ways with varying complexity. One of the most intuitive ways is to simply average the prediction of all the individual learners[8]. I.e.:

$$f_{ens} = \frac{\sum_i^M f_i}{M} \tag{2.6}$$

where M is the number of learners in the ensemble.

Another way is to do a *weighted averaging* where one takes a convex combination of the output, which is described in equations 2.2 and 2.3.

The weights of the different learners can be considered as hyper-parameters and can thus be determined via hyperparameter-tuning.

Another way to combine the predictions of the learners is to use another learner, a so called meta-learner, which takes as input parameters the regular input and the outputs of the learners. This is further described in Section 2.2.3.

2.1.4 Benefits of Ensemble Learning

Explainability and interpretability of a ML model is of high importance for making trusted decisions in real-world applications. Interpretability refers to the ability to understand and observe how a model operates and responds to different inputs. Explainability, on the other hand, refers to the ability to communicate a model's decisions and predictions in a way that is comprehensible to humans. These concepts pose a challenge in machine learning.

There are different categories of ML models depending on their explainability level and accuracy. *White-Box* models, have a more transparent inner logic and are

therefore more explainable. Simple decision trees and and linear regression models are examples of these. *Black-Box* models on the other hand, are difficult to interpret since it is unknown exactly how their inner mechanisms work, and only their expected inputs and corresponding outputs are known. Examples of these models are deep or shallow neural networks. For this reason, using ensemble learning, where it is possible to understand where the individual models work well, is favorable over neural networks, in case where this explainability would be lost.

In machine learning, interpretability techniques fall into two main categories: intrinsic and post-hoc interpretability. Intrinsic interpretability involves prediction models, which are interpretable by nature, e.g. White-box models. Post-hoc interpretability techniques, e.g. SHAP (see Section 2.5), explain and interpret predictions of any model without accessing its internal structure or weights[9].

2.2 Methods for Constructing Ensembles

The purpose of using ensembles of classifiers to make predictions is to overcome the different limitations of the models and where they make errors. Thus, to optimize the performance in different applications, different combinations of classifiers will yield the best results. Ensembles often outperform their individual classifiers. For this to be effective, the individual classifiers must be both diverse, so the errors they make are on different data points, and accurate, performing better than random guessing[10].

There exist popular methods of combining models. Three rules that have shown consistently good results across different problems are the *linear* combiner, the *product* combiner and the *voting* combiner. For models that output real-valued numbers the linear and product combiners are appropriate. For models that output class labels, voting combiners are useful. Below, some of the most commonly used ensemble learning algorithms will be presented, namely *Bagging*, *Boosting*, and *Stacking*[11].

2.2.1 Bagging

Bootstrap AGGregatING (*Bagging*) is an *implicit* learning algorithm. This means that diversity is introduced by randomly subsampling the training data and feeding each learner a different subset. To generate each subsample, also called a *bootstrap*, N items are chosen uniformly at random with replacement from the total N instances. The probability of any instance not being chosen is then $p = (1 - \frac{1}{N})^N$. Therefore for a large N a bootstrap is expected to contain $1 - e^{-1} \approx 63.2\%$ of the entire training set. Each classifier is then trained on a different bootstrap and in parallel to the other classifiers, see Figure 2.1. This means that each member of the ensemble is trained on a different training set[11]. The variation between the data sets that each classifier is trained on establishes diversity in the ensemble. Finally every prediction in the ensemble is combined either by uniform averaging or by voting over class labels[12].

Since the algorithm aims at reducing variance, bagging works best with unstable models, also known as *high variance* models, where small changes in the training data results in measurably varying decision boundaries. This makes bagging well suited for simple problems with relatively small available training datasets[11].

A variant of bagging is the *random forest* algorithm. The algorithm constructs a number of decision trees and combines their individual results to make a final prediction. The method applies two types of randomization. Firstly, it applies bootstrap sampling, where each sample is used to construct a decision tree. Secondly, at each decision node of the tree, a subset of predictors is randomly selected—typically the square root of the total number of predictors, p, though this can be adjusted. Within these selected predictors, the algorithm evaluates all possible thresholds to determine the best variable-threshold combination that most effectively differentiates the target classes. This process continues until nodes are pure, containing only one class, or a predefined endpoint is reached. Each tree grows independently, and this process is often repeated hundreds to thousands of times, forming a "random forest". In predictive use each new case is processed by all trees and for regression tasks these predictions are averaged to get the final prediction[13].

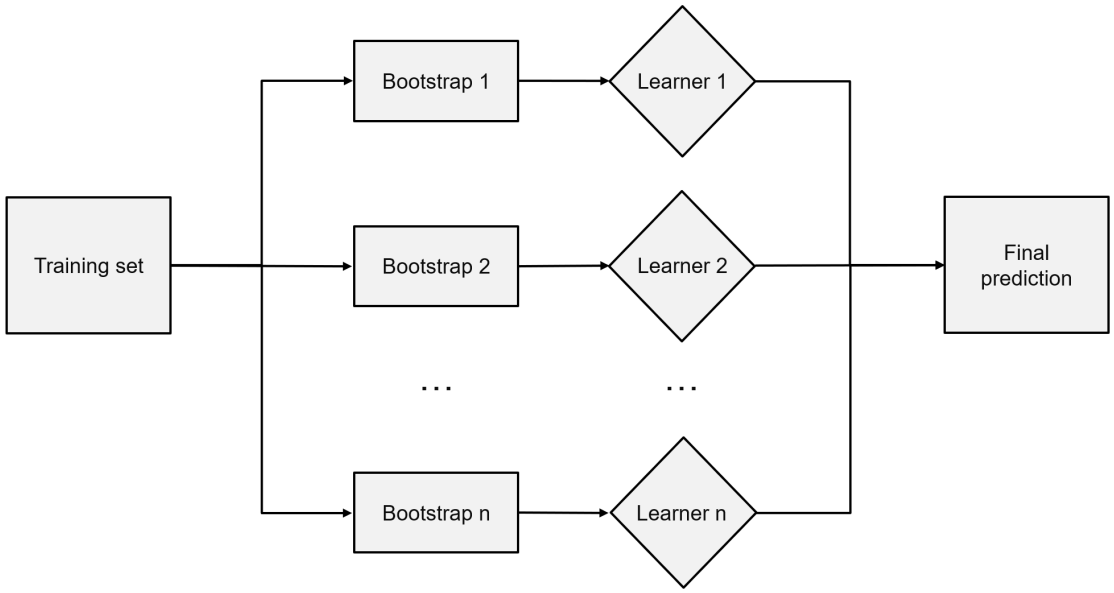

Figure 2.1: Illustration of the bagging principle.

2.2.2 Boosting

Boosting is an *explicit* learning algorithm. Diversity is introduced by constructing each ensemble member with different a training data set. Compared to bagging, boosting has a sequential and not a parallel algorithm. It works in an iterative manner where, after each trial, the subsequent classifiers focus on misclassified instances of the previous classifier, see Figure 2.2. In this way the algorithm aims at reducing the bias by letting the upcoming classifiers correct the mistakes of the previous ones[12].

Since the algorithm aims at reducing bias, boosting works well with stable but

weak models, also known as *high bias* models, where the predictive performance is consistently underwhelming across different samples of data. Boosting incrementally improves model accuracy by focusing on the errors of previous models in the sequence, making it especially effective for complex problems where simple models fail to capture underlying patterns. This approach allows boosting to adaptively enhance weak learners, making it well-suited for larger and more diverse datasets, where capturing subtle nuances in the data is crucial for good performance.

AdaBoost, short for Adaptive Boosting, is the most well known boosting algorithm. At the start of the algorithm each data instance is assigned equal weight. After each trial the vector of weights for the instances is adjusted such that the weight of misclassified instances is increased, thus reflecting the performance of the classifier[14]. In the upcoming round, the higher weighted instances have more influence on the classifier learned. This process iteratively refines the classifier, enabling it to focus more on the difficult cases and improve its overall accuracy over successive rounds.

Gradient boosting (GB), is a boosting algorithm that constructs an additive approximation of the target function by iteratively fitting new models to the residuals of previous models. The goal is to minimize a given loss function and unlike AdaBoost, which adjusts weights on the data points, gradient boosting directly optimizes the loss function by training each new model on the residuals of the previous one.

XGBoost, short for Extreme Gradient Boosting is another boosting algorithm that builds on the principles of GB to deliver improved computational speed, scalability, and predictive performance. Compared to GB, XGBoost incorporates a regularization term in its objective function. This process is governed by an objective function that integrates a loss function, which measures prediction accuracy, and a regularization term that controls model complexity to prevent overfitting. Each iteration seeks to optimize this objective function, continuously refining the model's accuracy and robustness against diverse datasets.

LGBM, short for Light Gradient Boosting Machine, also builds on GB but with some variation to enhance the training speed. It focuses on instances with higher gradients, to prioritize learning from the most informative data points and reduces the dimensionality of the data by bundling sparse features, which helps in speeding up the computation[15].

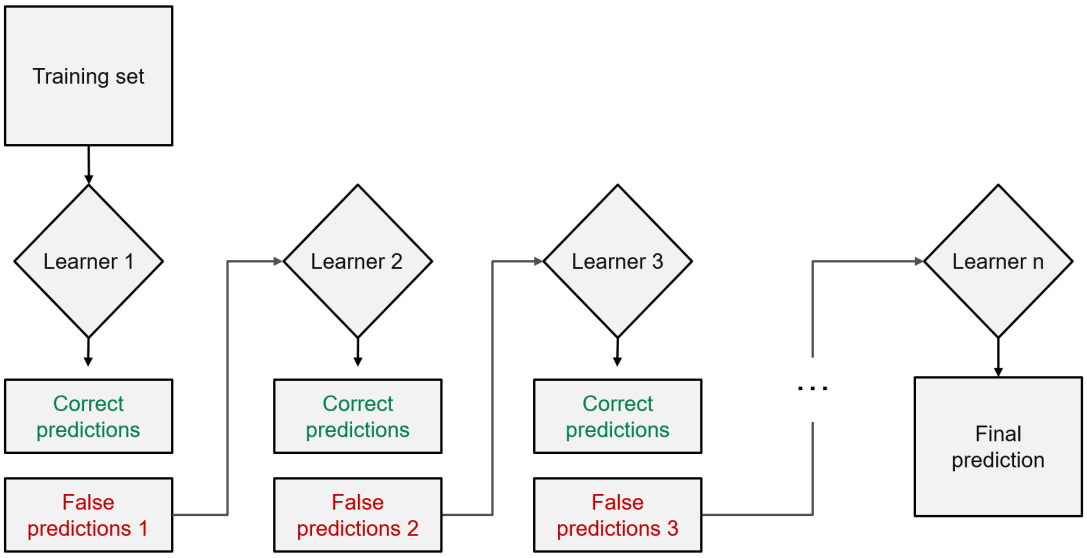

Figure 2.2: Illustration of the boosting principle.

2.2.3 Stacking

The algorithms discussed previously use non-trainable combiners, meaning the combination weights are fixed after the individual classifiers are trained. With this approach it is not possible to identify which classifier has effectively learned which partitions of the feature space. In contrast, using trainable combiners allows for determining which classifiers perform well in specific regions of the feature space and using this information to make a good combination of them. This method involves using a separate classifier to combine the outputs of the ensemble members, which is trained specifically on these outputs, thereby creating a stacked generalization model[12].

Stacked generalization, or stacking, is an ensemble learning technique designed to improve the predictive performance of machine learning models. Stacking works by combining multiple base classifiers, in a 'level 0 space', and generalizing their results with a *meta-learner* in a 'level 1 space'. A meta-learner is a model that learns to combine the predictions of multiple base models. The process works by partitioning the training data set into different blocks, each of which are used to train a level 0 classifier. The predictions of these classifiers are then used as input features for level 1 classifier, the meta-learner.

This meta-learner is trained to optimally combine the predictions of the base models, effectively learning which models perform best under various conditions. The key advantage of stacking is its ability to leverage the strengths and mitigate the weaknesses of various learning algorithms, leading to a more robust and accurate final prediction[16]. The concept can be likened to a manager selecting the opinions from a staff of workers who are all specialized in different topics. It is up to the manager to select, based on the input, who in the team would be best suited to make the best prediction. Here, it is important that the team members are both skilled and

diverse, meaning they do not usually make the same predictions as their teammates.

By fitting one or more meta-learners on the predictions made by base learners, stacking aims to correct the errors that these base learners make, ultimately reducing variance and bias[17].

An example architecture of a stacked model is depicted in Figure 2.3.

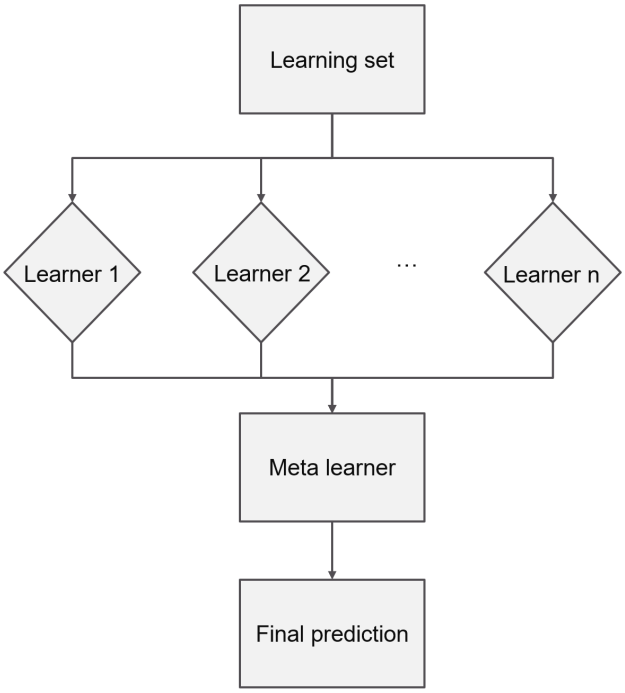

Figure 2.3: Example architecture of a stacked ensemble model.

2.3 Individual Regression Models

Apart from the ensemble methods described in Section 2.2, additional individual regression models will be incorporated into the evaluation process. These models are considered potential members for the stacked model, see Section 2.2.3, and are also of interest for assessing the overall performance of all models.

2.3.1 Ridge Regression

Ridge regression is a regularization technique that modifies the least-squares regression by adding a quadratic penalty to the loss function. The penalized residual sum of squares becomes

$$(y - Xw)^T(y - Xw) + \lambda w^T w, \tag{2.7}$$

where X is the matrix of input variables, y is the vector of output variables, w is the vector of weights in the regression model and λ is the penalization parameter. This

adjustment shrinks the coefficients, reducing their size compared to ordinary least squares. As λ increases, the model becomes less flexible but with lower variance, leading to a bias-variance trade-off. Ridge regression improves prediction accuracy but does not perform feature selection since it includes all features in the model[18].

An extension of ridge regression is the *Bayesian ridge regression*, wich incorporates Bayesian inference to estimate the parameters of a ridge regression model. Unlike traditional ridge regression, which uses a fixed penalty term to shrink coefficients, Bayesian ridge regression introduces prior distributions for the parameters, allowing for a probabilistic interpretation[19].

Kernel ridge regression is another variant of ridge regression designed to handle non-linear data by incorporating kernels, allowing the model to operate in a higher-dimensional feature space. It adapts ridge regression to the dual variable space, a technique that was originally simplified from SVM. While maintaining the core objective of ridge regression, which is to minimize prediction error with a regularization penalty on the coefficients, kernel ridge regression extends this approach by using the kernel trick to enable non-linear modeling capabilities[20].

2.3.2 Lasso Regression

Lasso regression is a regularization technique that introduces a penalty equal to the sum of the absolute values of the coefficients,

$$(y - Xw)^T(y - Xw) + \lambda \sum_{j=1}^{p} |w_j|. \tag{2.8}$$

This leads to a sparse solution where some coefficients are exactly zero, thus performing feature selection. Unlike Ridge regression, Lasso can handle high-dimensional data but can select at most N features if $p > N$. The flexibility parameter λ controls the bias-variance trade-off; as λ increases, the model becomes less flexible but with reduced variance. However, Lasso may struggle with grouped variables, often selecting only one variable from each group.

2.3.3 Elastic Net

Elastic Net combines the penalties of both Ridge and Lasso regressions to overcome their individual limitations. The penalization term is a weighted average (with weight α) of the Lasso and Ridge penalties:

$$P(w) = \alpha \sum_{j=1}^{p} w_j^2 + (1 - \alpha) \sum_{j=1}^{p} |w_j|. \tag{2.9}$$

This approach retains the feature selection ability of Lasso while shrinking the coefficients of correlated features, like Ridge. Elastic Net is particularly useful when dealing with highly correlated variables, as it can include all relevant features without dropping any by default. This method balances between Ridge and Lasso, providing more flexibility and robustness in model fitting[18].

2.3.4 Suport Vector Machine

Support Vector Machine (SVM) is a type of linear algorithm used for various applications, including classification, regression, density estimation, and novelty detection. In the simplest case of two-class classification, SVMs aim to find a hyperplane that separates the two classes with the maximum possible margin. This hyperplane is determined by the data points that lie on the margin, known as support vectors, leading to the name *support vector machine*. This approach enhances the model's generalization accuracy on new, unseen data and leverages specialized optimization techniques that enable SVMs to effectively learn from large datasets[21].

2.4 Hyperparameter Optimization

Learning algorithms often involve producing a function by optimizing a training criterion using a set of parameters. However the learning algorithm itself is also parameterized by so called *hyperparameters*, that make the learner highly configurable and impact its performance. The actual learning algorithm is determined once the hyperparameters are set[22]. Examples of hyperparameters of the random forest algorithm include the number of trees in the forest *n_estimators*, the function to measure the quality of a split *criterion* and the maximum depth of the tree *max_depth*[23]. The goal is to select these parameters so as to minimize the generalization error which is called the problem of *hyperparameter optimization*.

There are various methods for hyperparameter optimization, among the most common are grid search and manual search. Due to the large amount of models that are being evaluated, hyper-parameter tuning of all of them is computationally demanding. This is where random search proves beneficial. Random search does not allocate equal computational resources across all possible values of hyperparameters. Instead, it allocates trials randomly, which makes it more likely to explore valuable regions of the hyperparameter space. This approach is beneficial because not all hyperparameters equally influence the performance of the model. Therefore, random search can avoid excessive focus on less impactful hyperparameters, a common drawback in grid search[22]. Furthermore random search has been shown to give comparable results to a more sophisticated hyper-parameter optimizer like meta-heuristics or grid search, while being far less computationally expensive[24].

2.5 Variable Importance with SHAP Values

It is reasonable to expect that it is useful information to know what data features the model considers to be most important, but it is not trivial to extract that information. One might intuitively expect that the magnitude of the different coefficients would indicate the importance of the feature however that does not take into account the scaling of the feature. For example, if "budget" was a feature it should not matter whether it was represented in dollars or cents, however the difference in the coefficient would be of factor 100.

One way to solve this is to use *SHAP* (SHapley Additive exPlanations)[25]. SHAP

is a technique to evaluate the importance of different variables in a model. The concept of Shapley values comes from game theory. It is a way to quantify the value or cost of different players in a collaborative game. To conceptualize this consider the scenario of three friends sharing a taxi home. The three friends live on the same long street and they need to figure out how to split the cost. Shapley values indicate that each of the friends pays the average additional cost they contribute to every different permutation of friends. In this scenario the taxi cost 1 unit of currency for every meter, and the first friend live 100 meters down the road, the second 400 meters down the road, and the third friend lives 1000 meters down the road. For the first scenario friend 1 steps into the taxi, he would pay 100. Then the second player steps in and he would pay an additional 300 and the thirds steps in last and he would pay 600, see Table 2.1 for all scenarios.

Scenario	Friend 1	Friend 2	Friend 3
1,2,3	100	300	600
1,3,2	100	0	900
2,1,3	0	400	600
2,3,1	0	400	600
3,1,2	0	0	1000
3,2,1	0	0	1000
Shapley values	$\frac{200}{6}$	$\frac{1100}{6}$	$\frac{4700}{6}$

Table 2.1: Shapley table depicting the shapley values, i.e. how much of the taxi ride each friend should pay.

In a similar fashion, the importance of a variable, i.e. how much of the prediction can be attributed to it, in a model can be described by its shapley value. To understand this we must first understand what a partial dependence plot is. A partial dependence plot describes how much the output variable depends on one specific variable. An example partial dependence plot describing how the prices of houses in a block in California depends on the median income of the residents in that block is depicted in Figure 2.4. Here we are assuming linearity.

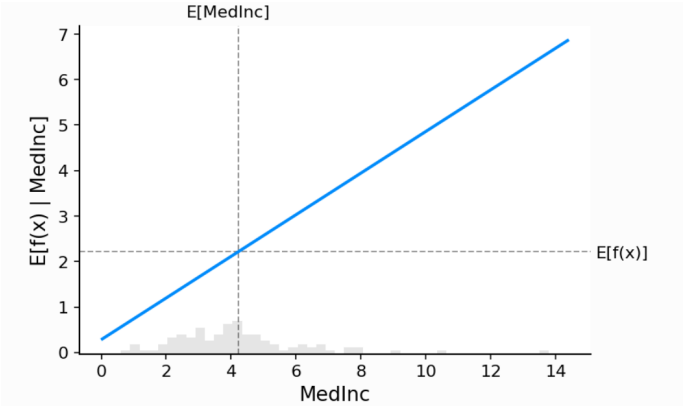

Figure 2.4: Example of a partial dependence plot.

The horizontal line here represents the average house price of the entire dataset, and the vertical line represents the average median income among the blocks in the data set. The blue line is the partial dependence plot line which is fixed to go through the intersection of the grey lines. This intersection can be considered as the center of the partial dependence plot, i.e. the value of which to set the variable in order for it to not have an impact on the output variable.

To give each variable a shapley value we consider them players in a cooperative game, similar to the taxi scenario above. The shapley values is the difference between the output when the variable opts out, i.e. setting the value of the variable to the center of the partial dependence plot, and the output when the variable joins in. This is represented by the red line in Figure 2.5

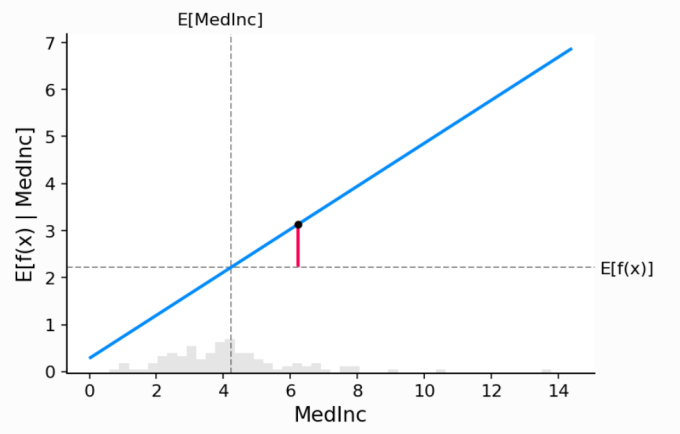

Figure 2.5: Shapley value from partial dependence plot.

For a linear regression case this is simply done by reading the distance from the centre since the different variables added together after being multiplied by their corresponding coefficients. The linearity requirement of the partial dependence plot can be relaxed to enable the use of decision tree model etc.

2.6 Novel Heuristic for a Local Prediction Interval

In this thesis we also propose a novel heuristic based method to determine a local *Prediction Interval* given unseen input. Methods such as the Jackknife method exist which produce a global prediction interval, i.e. that for a given α it returns the interval such that $1 - \alpha$ of predictions are correctly predicted to be within that interval[26]. However, this returns the same interval regardless of where the input is in the input space. We believe that models often perform better on certain parts of the input space than others, and that a local prediction interval should be able to reflect that. To conceptualize this, imagine a machine learning model that classifies a picture as either a dog or a cat. It is reasonable to expect that given a clear picture of a dog without any noise will be more accurately classified as a dog than a picture of an airplane would. The same principle ought to apply in a regression context

as well, where given some inputs the guess is very accurate while other guesses are much less accurate.

In order to calculate a local prediction interval for a new instance $\mathbf{X_{new}}$, we first need to calculate the set of the k nearest neighbors of $\mathbf{X_{new}}$. Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ be the data points in the test set and let $e(x)$ yield the error of the data point x.

We define the distance $\mathbf{d}$ between two data points using dot product between the model's coefficients $\mathbf{c}$ and the norm of the data points, i.e.:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \| \mathbf{c} \cdot (\mathbf{x}_i - \mathbf{x}_j) \| \tag{2.10}$$

Then, the set of the k nearest objects to $\mathbf{x}$ can be denoted as:

$$S_k(\mathbf{x}) = \{\mathbf{x}_{(1)}, \mathbf{x}_{(2)}, \ldots, \mathbf{x}_{(k)}\}$$

where $\mathbf{x}_{(i)}$ represents the i-th nearest object to $\mathbf{x}$ such that:

$$d(\mathbf{x}, \mathbf{x}_{(1)}) \leq d(\mathbf{x}, \mathbf{x}_{(2)}) \leq \cdots \leq d(\mathbf{x}, \mathbf{x}_{(k)}) \leq \cdots \leq d(\mathbf{x}, \mathbf{x}_{(n)})$$

Further, let $\mathbf{y_{new}}$ be the model's prediction of $\mathbf{X_{new}}$. The local prediction interval p can then be given by:

$$p = \mathbf{y_{new}} \pm \max_{x \in S_k(X_{new})} e(x) \tag{2.11}$$

In the above equations, a linear model is required in order to determine how close two data points are to each other. This is because the distance in dimensions for which the model have large coefficients matters more than the same distance in a dimension with a lower coefficient. To account for this the dot product between the coefficients and the input vector is used rather than just the input vector.

Setting the k variable is not trivial since if it is too low the accuracy of the prediction interval becomes lower, and if it is too high then there is a risk of including data points far away from the new input. To set a good value of k you have to have a good understanding of how the data looks like. For example, if the dataset is unbalanced you might have to set k to a lower value to reduce the risk of including data points from far away in the calculation of the prediction interval. On the other hand if you have a balanced and well populated dataset then you can set k to a higher value since there is now a higher chance of including only relevant data points.

To clarify, this approach is dynamic for each new input, unlike a usual prediction interval. In the regular case a percentage is set - for example 95% - and the interval is calculated in such a way that 95% of the test points are within that interval. However, this does not take into account that predictions are made with different certainty depending on how well the model is trained on that kind of input or depending on the random distribution at that point in the input space.

A prediction interval shows how sure the model is of its prediction, but in the predicting of sales from a construction project it can also be seen as a measure of risk. The larger the prediction interval, the larger the risk would be. This information would be useful for HILTI, allowing them to navigate the dichotomy between higher expected returns and greater risks.

3

Method

In this section we will describe how we obtained and processed the data; what models were used; and why we used them. We will also describe the models and how we evaluated the results of the models.

3.1 Dataset Description

The dataset consists of HILTI projects that have been completed. The data is gathered from two data sources within Salesforce.com, one from which to gain *net sales* (NS) data, and one to get parameters that are available for a project proposal. The two datasets are amalgamated to create our data, which the models can be trained and tested on. After having filtered the dataset so that only completed projects that have reported net sale data and *HILTI Relevant Potential* (HRP) remain, we are left with more than 17,000 projects.

The data consists of the following parameters:

- Duration of the project
- Country in which the project takes place
- HRP - HILTI Relevant Potential, the amount of sales that HILTI could theoretically achieve. Usually around one percent of this is achieved.
- Focus Level - whether the project is on a local, regional, or global level.
- Project Type - type of industry, for example: hospital, universities, commercial/retail etc.
- Project Segment - larger categories of the industry, for example: power, mining, etc.
- Project Classification - size of project, classified in three tiers (1,2,3) where 1 is considered the largest.
- Potential Class - three different classes describing the size of the customer. The classes are (A,B,C) where A is the largest.

We are trying to predict the net sales over time. Net sales is the revenue generated from a project through the selling of not only goods, but also services like designing and planning, etc., converted to Swiss Francs. We are dividing the net sales by

time in order to get a comparable measurement between projects of how much sales are actually being made. Take for example two projects, one that has a million in reported sales and lasted for a decade, and one that has 200,000 in reported sales but only lasted for a year. The latter one generates less sales but over a shorter time and is to be preferred in the end.

Unfortunately, there is no reported profit in the data, only net sales. It is however reasonable to expect that the more the company sells the more profit it makes, considering that it is a manufacturing company.

The dataset is rather imbalanced, there are for example, many more projects with a *Project classification* of tier 3 than there are with a *Project classification* of tier 1. Tier 1 projects and tier 3 projects usually differ a lot, due to the difference in size. It can thus be difficult for the models to accurately predict the net sales over time for the tier 1 projects, since they are trained on a majority of tier 3 projects.

It should be noted that the data is entered by many different employees all around the world and there is unfortunately no guarantee that everyone follows the same principles when inputting the data. This could be a potential reason for data outliers. It is difficult to discern between projects that in reality had low net sales and the ones that were misreported.

3.2 Selection of Models

Given that the objective of the thesis is to evaluate and predict the performance of project proposals different regression models will be tested. Regression models are used to describe the dependence of a response variable on a set of explanatory variables. The two main reasons to use regression models are for *explanation* and *prediction* purposes. In the former case, the regression model is utilized to assess the impact of an explanatory variable on the response variable, while simultaneously accounting for the influence of various other variables incorporated within the model. In the latter case, regression models are used for the prediction of unobserved instances or forecasting future values of the response based on current values of explanatory variables. In this predictive capacity, the regression model provides an estimate of the expected or predicted value of the response as a function of the explanatory variables[27]. In this thesis, the use of the term "regression model" refers to any model that outputs a continuous variable.

We have decided to test many different regression models. The reason for this is twofold: we want to gain insight in what models work best for the problem, and we want to find models whose residues have low correlation with each other in order to create a good ensemble. This was done by creating a correlation matrix, as depicted in Figure 5.2, with the correlation between each pair of models and comparing this with the *mean absolute error* (MAE)[28] of the individual models. To create an ensemble with high diversity we are interested in models with low MAE and a low correlation to the other models in the stack.

3.3 Method for Hyperparameter Optimization

The chosen method for the hyperparameter optimization is random search. Random search is chosen due to its computational efficiency and ability to effectively explore the hyperparameter space. Unlike grid search, which systematically evaluates a predefined set of hyperparameter combinations, random search randomly samples combinations, allowing it to potentially discover more effective hyperparameter values without exhaustive computation. This method has been demonstrated to achieve performance comparable to more complex optimization techniques while significantly reducing computational costs, see Section 2.4.

3.4 Evaluation Methods

HILTI currently stores their proposals in the Salesforce.com platform where parameters such as size (turnover), account (relationship to the project proposer), external risk (political and economic), segment (availability of supporting facilities), and what phase the project is in are specified and scored based on rules of thumb. These five parameters are the only ones used in the current evaluation. The parameters' individual scores are summed to give a total score for the proposal. These parameters are currently weighted equally.

The completed projects at HILTI are currently rated using a metric called 'share of wallet', which is calculated by dividing the HRP by the NS, where the HRP represents the expected margin and NS is the actual margin. Instead, we will use the metric NS per month as a rating to capture the efficiency of the projects. Furthermore, additional parameters will be included in the evaluation that are currently not being considered, see Section 3.1.

4

Evaluation

To evaluate the models, their performance in terms of mean absolute percentage error will be compared. This will be done using a k-fold cross validation.

The *mean absolute percentage error* (MAPE) is calculated by:

$$\frac{1}{N}\sum_{i}^{N}\frac{|y_i - \hat{y}_i|}{y_i} \cdot 100 \tag{4.1}$$

where N is the number of instances in the test-data, y_i is the i:th value of the test data, and $\hat{y}_i$ is the i:th prediction.

This is used to evaluate a result proportionately to the actual value. For example, if the MAE is 5000 for a data instance, that would be a rather good result if the actual value was very high, but a rather poor result if the actual value was low.

5

Results

In the following section, the results of the thesis will be presented. To evaluate whether ensemble learning methods can be used to enhance project selection at HILTI, 13 different models were evaluated. Six of these were regular regression models and another six were existing ensemble learning models. These twelve models were used to develop a seventh stacked ensemble model, specific to the problem.

5.1 Mean Absolute Percentage Error of Models

Table 5.1 displays the mean absolute percentage error of all the models for the two NS criteria: NS > 1,000 (1,395 projects) and NS > 10,000 (1,092 projects). The models are sorted in ascending order according to the performance in the NS > 1,000 case. In Figure 5.1, the same values of the model performances are presented for all models apart from the two worst performing ones.

Model name	MAPE (%) for projects with NS > 1,000	MAPE (%) for projects with NS > 10,000
Support Vector Machine	266.3	120.97
Light Gradient Boost	379.42	203.96
XGBoost (tree)	381.28	182.31
Stack (see Figure 5.3)	384.19	197.44
Elastic Net	407.44	195.93
Gradient Boost	410.19	213.24
Random Forest	438.72	204.48
XGBoost (linear)	445.03	230.44
Lasso	479.53	244.69
Ridge	483.42	218.13
Bayesian Ridge	483.5	218.16
AdaBoost	1168.98	485.32
Kernel Ridge	55135.48	6050.28

Table 5.1: MAPE of each model for projects with NS > 1,000 and NS > 10,000. Sorted from best to worst according to NS > 1,000 criteria.

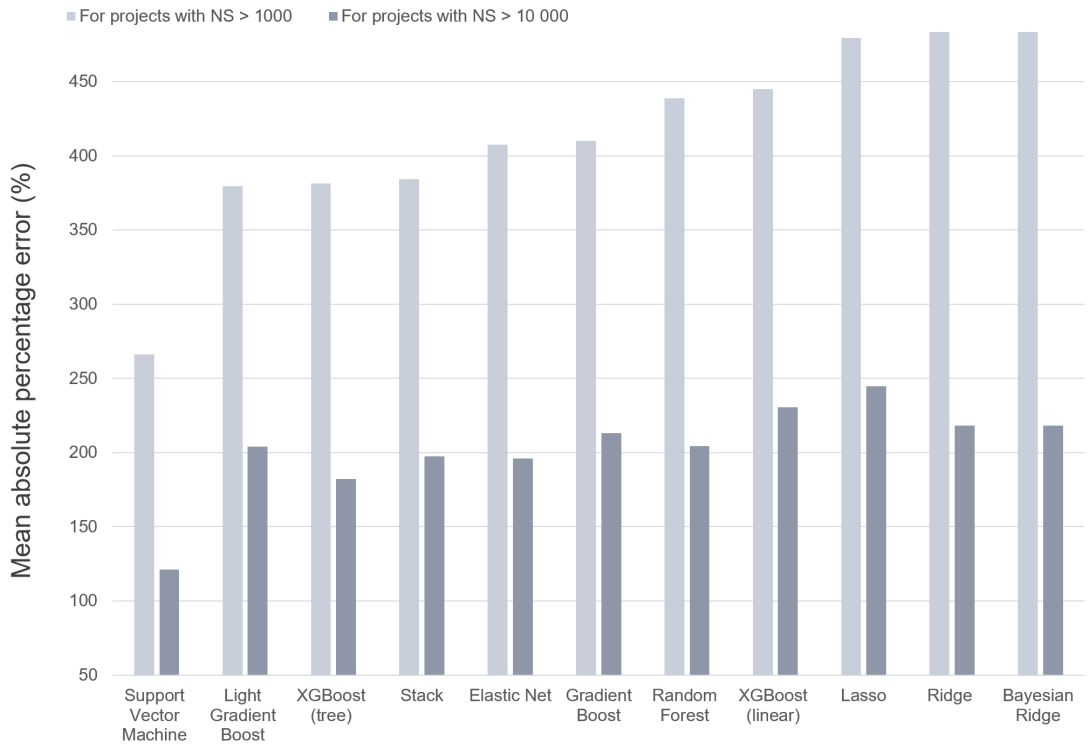

Figure 5.1: MAPE for each model, in NS/month.

5.2 Model Correlations

In Figure 5.2, the pairwise correlation of all the models are presented for the criteria NS $>$ 1,000, where low values indicate a low model correlation and high values indicate a high model correlation.

5.3 Architechture of Constructed Stack

The architechture of the final stack is illustrated in Figure 5.3. The level 0 space consists of models SVM, EN and LGBM. The meta-model for the level 1 space is Bayesian ridge.

5.4 Actual Values versus Predicted Values

The following six figures, 5.4, 5.5, 5.6, 5.7, 5.8 and 5.9, show the the predicted values of the models versus the actual values. All of the tests have been conducted using a k-fold cross validation of $k = 8$. The tests for figures 5.6, 5.7, 5.8 and 5.9, were run under the condition of NS $>$ 1,000.

5.5 Variable Importance

In the shapley plot depicted in Figure 5.10, the feature importance is shown in descending order. Since most variables (except HRP) are boolean, taking values of

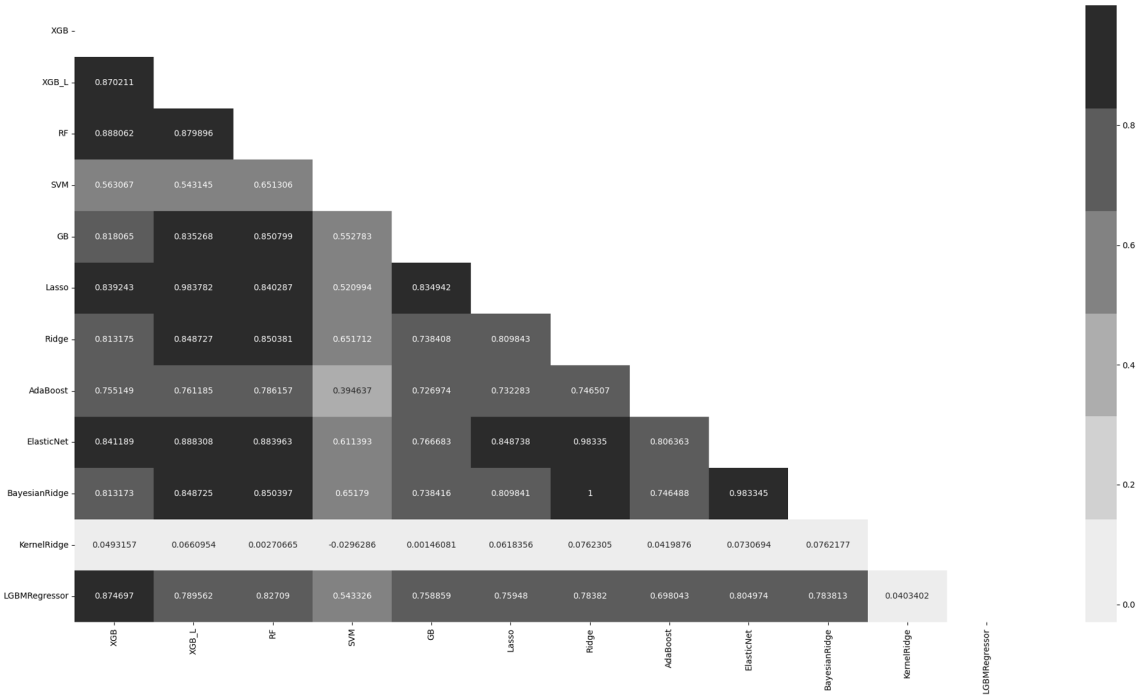

Figure 5.2: Pairwise correlation between models. Here, XGB_L is the XGB regressor using a linear booster, while XGB uses a tree based booster.

either 0 or 1, they will be represented with the color blue for 0 and red for 1. The features SHAP-values, for each instance, are plotted on the x-axis. A positive SHAP value means that the variable for that data instance caused the model to predict a higher value and vice versa.

5.6 Dataset Distribution

In Figure 5.11, the distribution of the NS values for the dataset is shown on a logarithmic scale in the y-axis.

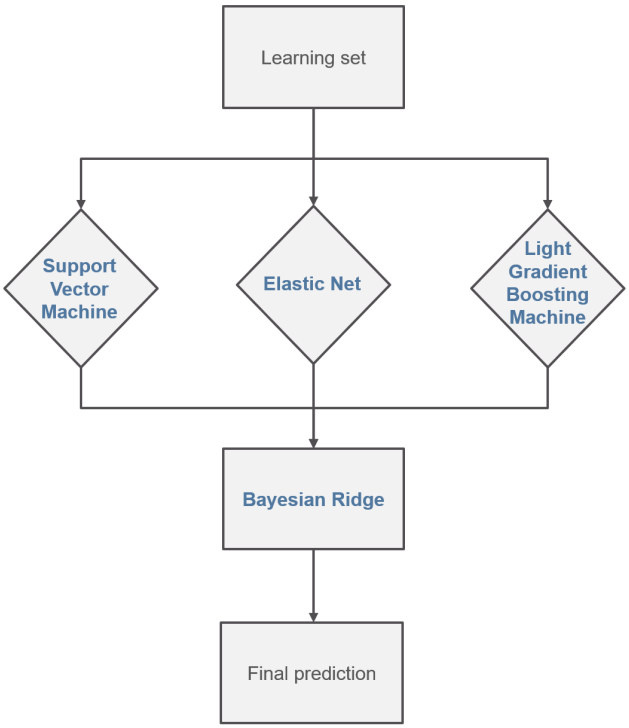

Figure 5.3: Architecture of the chosen stack of models.

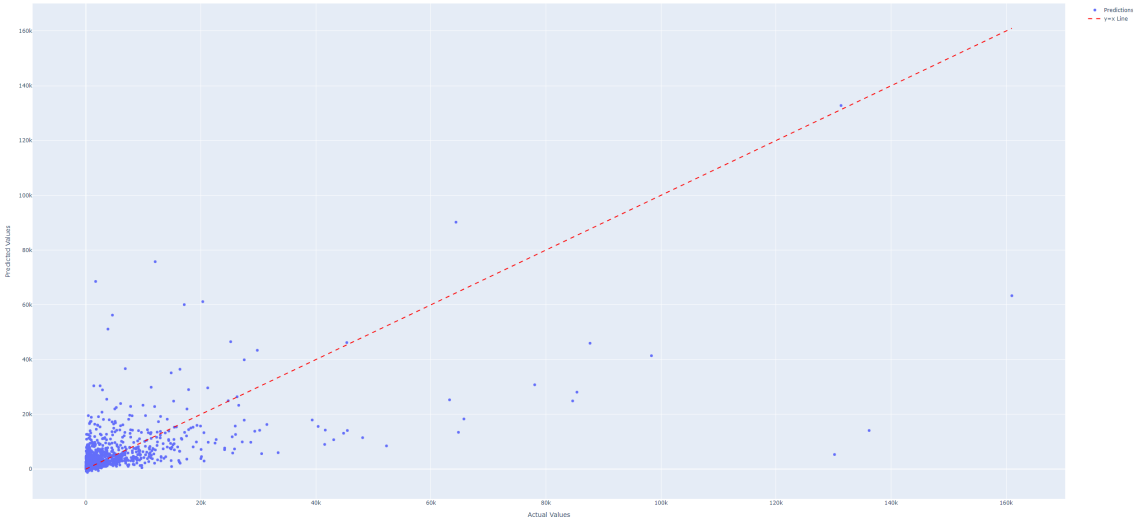

Figure 5.4: Actual versus predicted values from the stacked model for projects with NS $>$ 1,000.

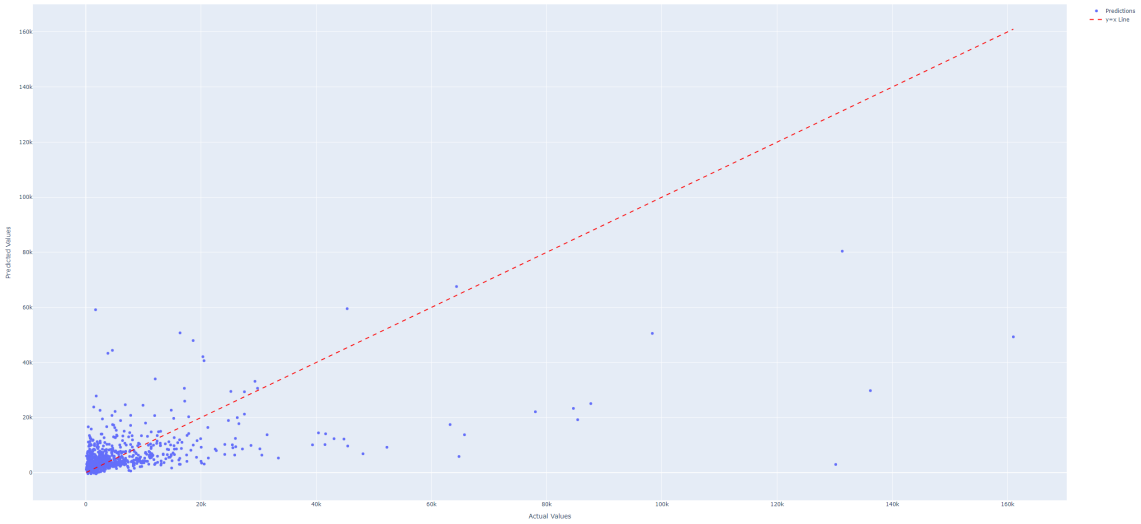

Figure 5.5: Actual versus predicted values from the stacked model for projects with NS $>$ 10,000.

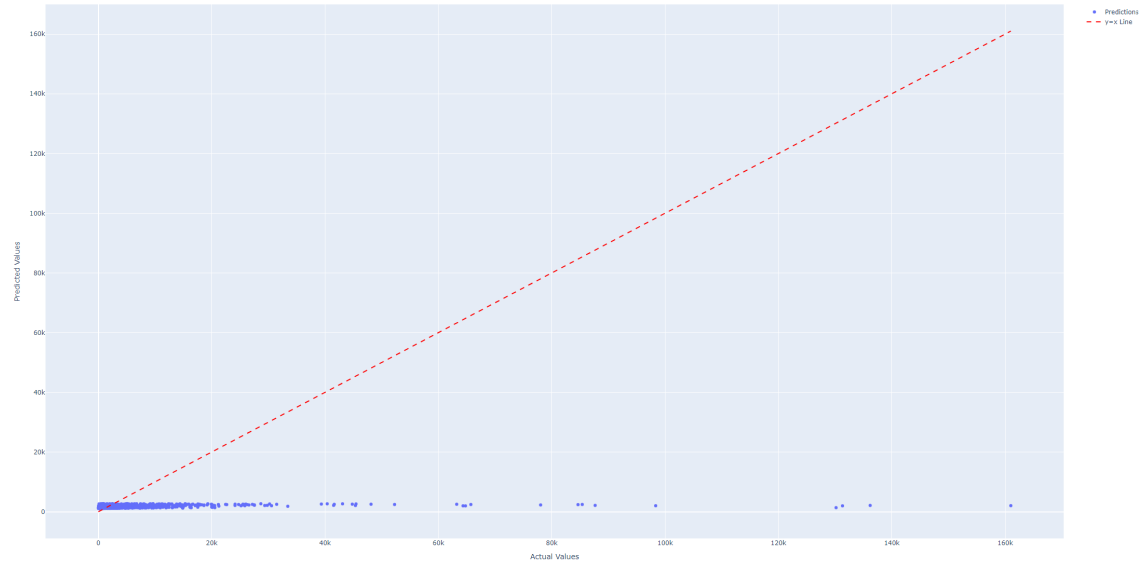

Figure 5.6: Actual versus predicted values from the SVM model for projects with NS $>$ 1,000.

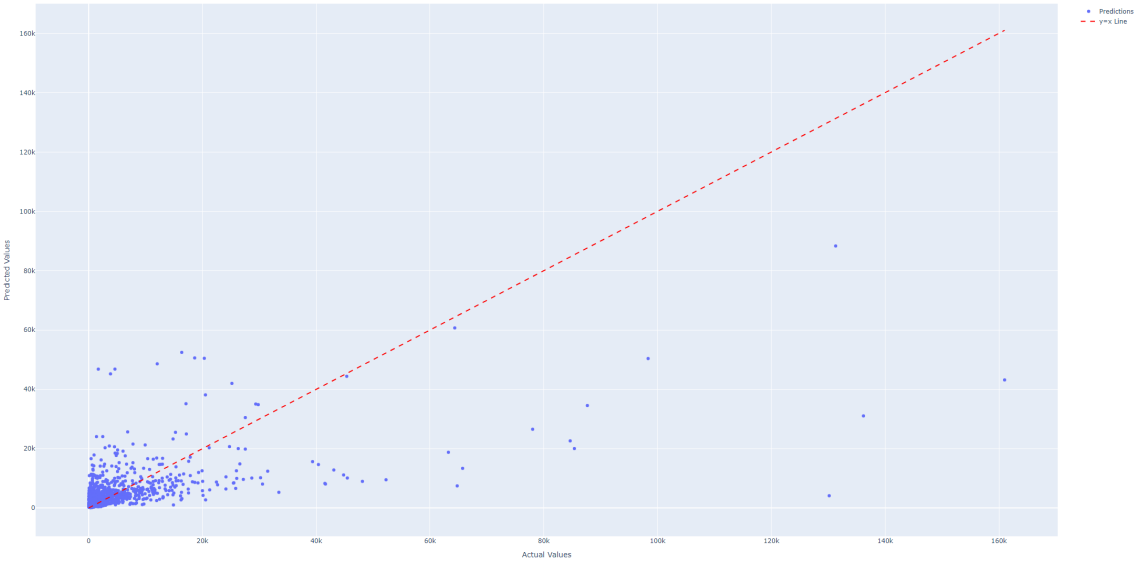

Figure 5.7: Actual versus predicted values from the EN model for projects with NS $>$ 1,000.

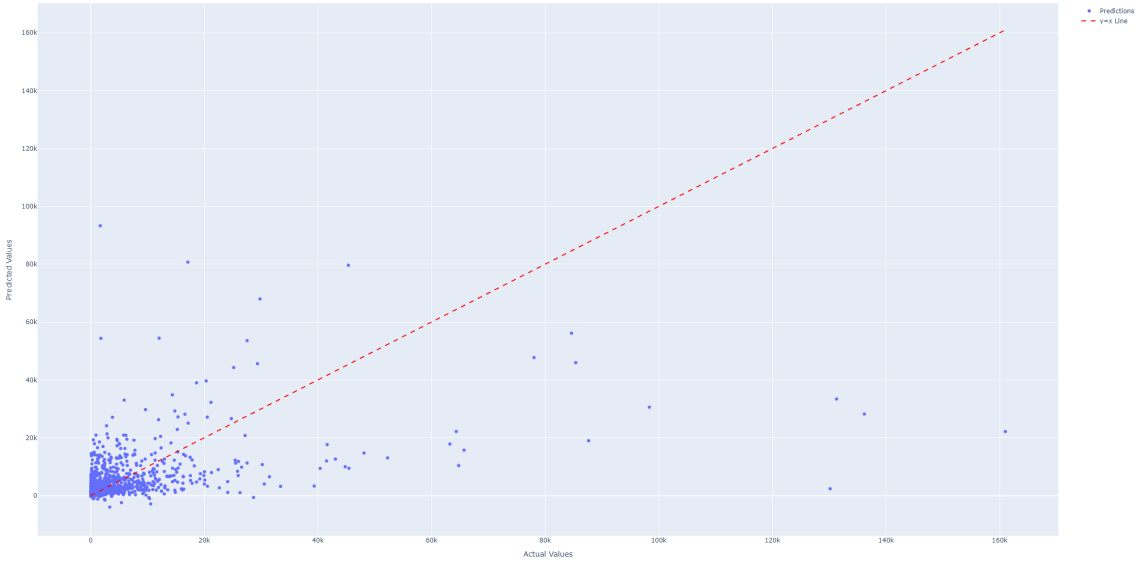

Figure 5.8: Actual versus predicted values from the LGBM model for projects with NS $>$ 1,000.

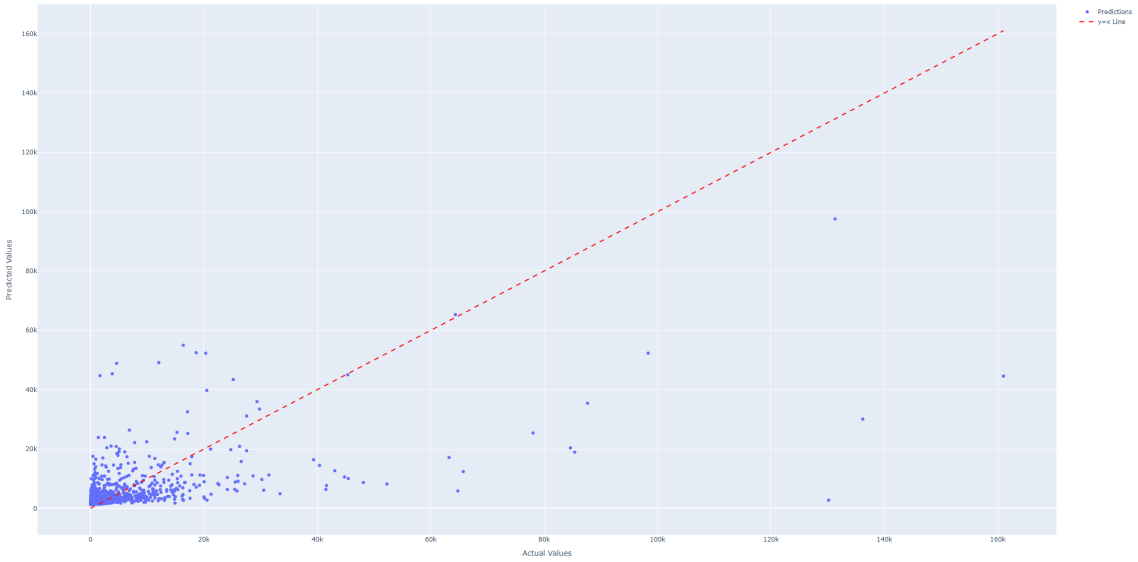

Figure 5.9: Actual versus predicted values from the BR model for projects with NS $>$ 1,000.

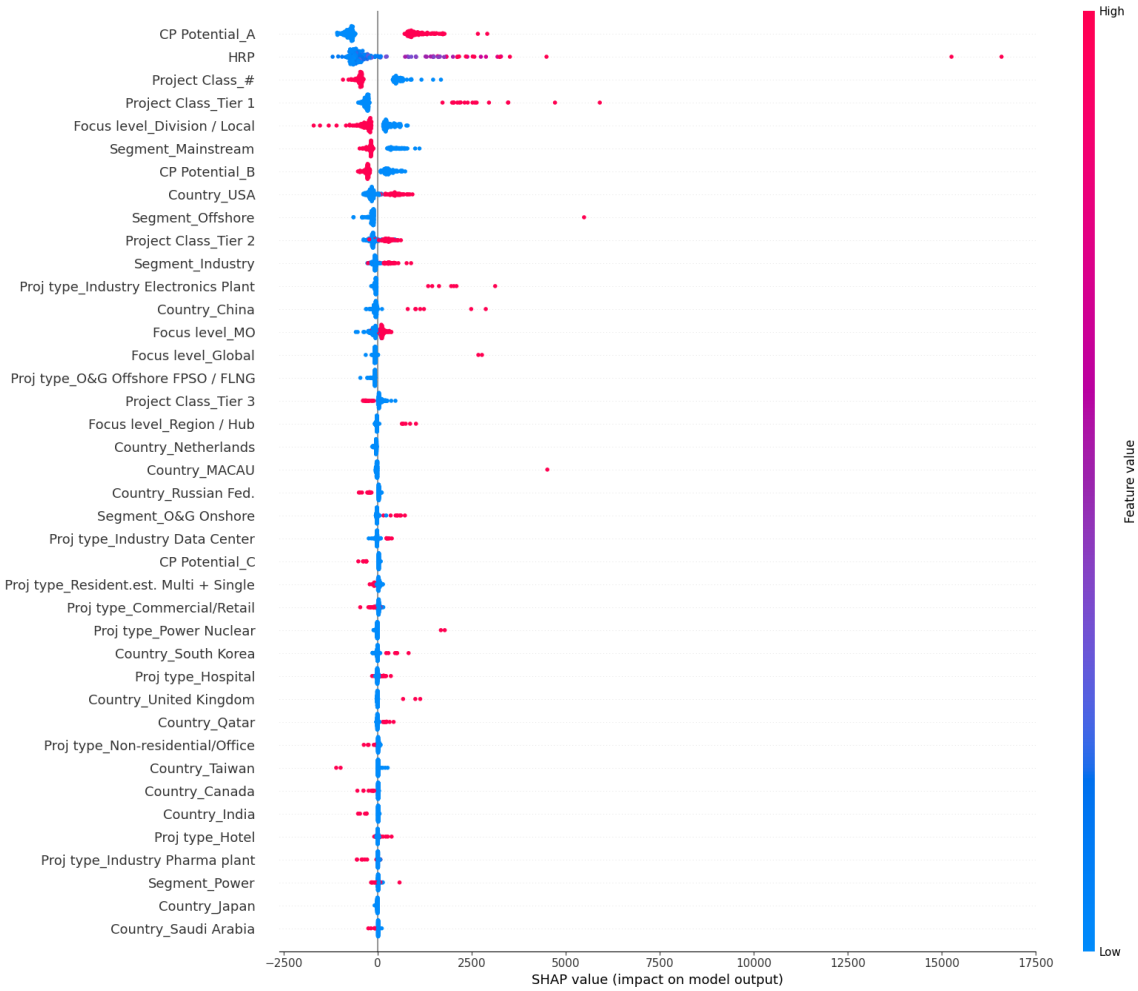

Figure 5.10: SHAP summary plot of feature importance for the top 40 variables.

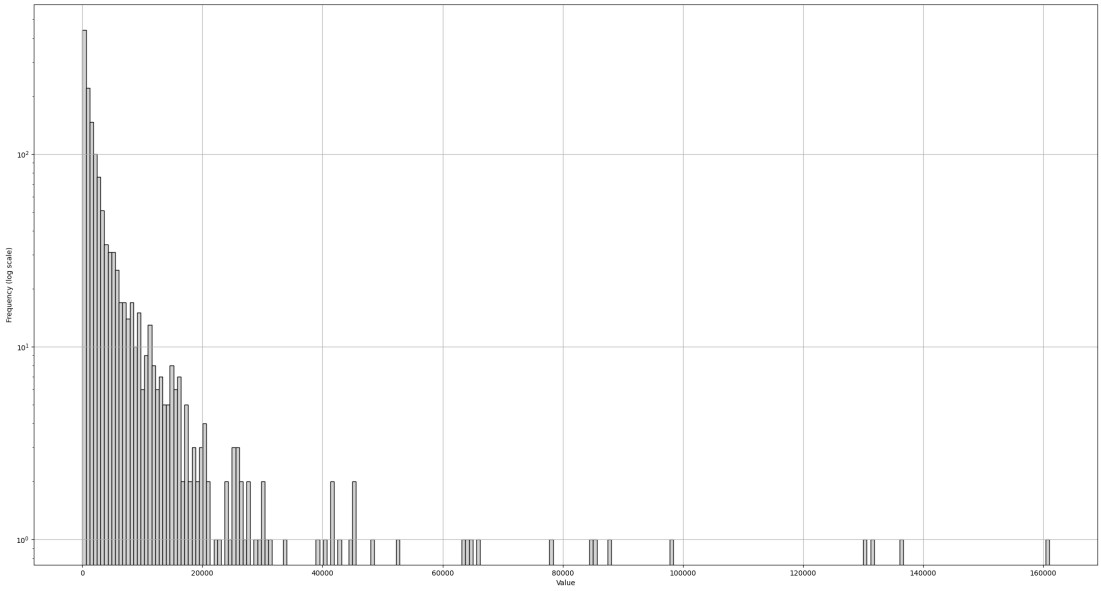

Figure 5.11: Histogram of the distribution of NS/month values for projects.

6

Discussion

In the following section, the results presented in chapter 5 will be discussed.

6.1 Model Performances

The results presented in Section 5 indicate generally low prediction accuracy for the models, primarily due to the highly unbalanced nature of the dataset. As illustrated in Figure 5.11, there is a significantly higher number of projects with relatively low net sales (NS), while comparatively few projects report much higher figures. To gain more insight into the models' prediction abilities, two different criteria were tested for the lower limit of the NS value: NS $>$ 1,000 and NS $>$ 10,000. The criterion of NS $>$ 1,000 includes more data instances, providing a more accurate representation of the dataset. However, this also results in higher prediction errors due to the dataset's imbalance. Conversely, the criterion of NS $>$ 10,000 offers a more stable dataset for model evaluation, resulting in lower prediction errors but a less accurate representation of the overall dataset. As shown in Figure 5.1, the MAPE of all models is substantially higher with the NS $>$ 1,000 criterion compared to NS $>$ 10,000. Further tests were conducted to evaluate the impact of introducing an upper limit on the NS value. However, these tests did not yield significantly different results.

The results in Table 5.1 indicate that the SVM regressor achieved the lowest MAPE for for both test cases, with a significant performance gap compared to the second-best model. For the criterion NS $>$ 1,000, the LGB regressor achieved the second-best result, closely followed by the XGB tree regressor and the stacked model. The AdaBoost and kernel ridge regressors performed so poorly that they were excluded from any further analysis.

As explained in the theory, see Section 2.1.4, it is crucial that the individual models in a stack not only perform well on their own but also exhibit low correlation with each other. This allows them to compensate for each other's shortcomings. Given the relatively small number of models, it was feasible to evaluate all possible combinations for the stack. This evaluation was performed for the criterion of NS $>$ 1,000 using an eight-fold cross-validation to ensure the optimal stacked model was identified. The best-performing combination of models was selected as the final stack, as illustrated in Figure 5.3. This stack consists of three base learners at the first level — SVM, EN, and LGBM — and a Bayesian ridge meta-model at the second level. As seen in Table

5.1, SVM and LGBM are the best-performing models. Although EN shows relatively worse performance, it still ranks within the top four models (excluding the stack at this stage). While the Bayesian ridge model was among the poorer performers, it still delivered reasonable results compared to AdaBoost and Kernel ridge. Looking at the correlation matrix in Figure 5.2, we see that both SVM and LGBM have a generally low correlation to the other models, but especially low correlation to each other. Elastic net has a high correlation to most models, however a lower correlation to SVM and LGBM. The relatively good performance of these three models, combined with their low correlation values compared to other models, suggests they are an effective combination of learners for the first layer of the stack. This was further validated by testing all possible model combinations.

Figure 5.4 and 5.5 depict the predicted values of the stacked model compared to the actual values for the cases of NS $>$ 1,000 and NS $>$ 10,000, respectively. For a well-performing model, the predicted values should be close to the actual values, which would mean that the blue dots representing each instance of the data would stay close to the dashed red line representing the $y = x$ line. However, in these plots, we see a dense area of points around the value zero. This is caused by the uneven distribution of points, as shown in Figure 5.11. For the points with higher values, there is a trend where the points move towards the $y = x$ line, with a few points very close to the line. This indicates that while the model performs better for higher values of NS, it struggles significantly with lower values, resulting in a clustering of predicted values around zero. The performance disparity suggests that the model's ability to generalize across the entire range of NS values is limited, likely due to the imbalance in the dataset. The concentration of points around zero in both figures highlights the challenge of predicting lower NS values accurately. This may be due to the inherent difficulty of capturing the variability in smaller projects, which could be influenced by numerous unpredictable factors. Conversely, the improved alignment of higher NS values with the $y = x$ line suggests that the model handles larger projects more effectively, potentially because these projects exhibit more stable and predictable patterns.

Figures 5.6, 5.7, 5.8 and 5.9 show the predicted versus actual values for the individual members of the stack for the case of NS $>$ 1,000. In Figure 5.6, the SVM model's predictions are consistently low and do not adapt to the $y = x$ line. Despite its naivety, this strategy works reasonably well in this case because there is a significant number of actual low NS values, making the SVM model's approach beneficial. However, this results in underprediction for higher actual values, as seen by the distance of these points from the red line. In Figure 5.7, the EN model's predictions show an improvement over the SVM model, especially for intermediate values where there is a more noticeable spread towards higher predictions. There is still a cluster of low values with a weak trend towards the $y = x$ line, for some values. However the model still makes underpredictions with the highest values, resulting in some points with a high actual value being far from the $y = x$ line. Similarly to EN, Figure 5.8, shows that the LGBM model performs better for the intermediate values where there is also a trend towards the $y = x$ line. The model has however a larger spread for the low values, where some points are even more overpredicted, as well as

underpredicting the highest values. Finally, Figure 5.9 shows that the Bayesian ridge model shows a mix in performance similar to the EN model. While it captures some of the variability, it still tends to underpredict higher actual values. The clustering of predictions around lower values suggests that the model struggles with capturing the full range of NS values effectively.

While the SVM model appears to perform the best based on Table 5.1, Figure 5.6 shows that its actual performance is quite poor. This inconcistency is likely due to the dataset being so imbalanced. In such cases, the MAPE of the SVM model may not be a reliable indicator of its performance. With a more balanced data set containing more instances with a high NS value, the models weak performance might have also been reflected in the MAPE of the model. Given the SVM model's strategy, the MAPE error can therefore be misleading for this dataset. The models that performed best overall were LGBM, XGBoost (tree), and the stacked model, as they demonstrated relatively good results consistently, in both MAPE and their plotted predictions.

6.2 Feature Importance

From the SHAP summary plot in Figure 5.10, the importance of the top 40 variables is shown in descending order. In the plot, each dot represents one project. Due to the one-hot encoding of the parameters, each variable within a parameter is represented independently, with either a high feature value (indicating that this variable is involved in the project) or a low feature value (indicating that this variable is not involved in the project).

The variable *CP Potential A* is shown to have the highest importance, where projects belonging to this class increase the predicted NS value, and projects not belonging to this class decrease the predicted NS value. In other words, projects belonging to *CP Potential A* are expected to perform well, while the opposite is true for projects that are not in *CP Potential A*. Conversely, projects without an assigned project class, such as those belonging to *Project Class #*, are expected to perform worse than projects not belonging to this group.

Projects in *Country USA* are also expected to perform well, but compared to *Country China*, they are not expected to have as high NS values, as seen from the lower SHAP values reported. However, when a project is in *Country Canada*, it is expected to affect performance negatively.

These findings are of value for HILTI as they indicate which variables are associated with better or worse expected performance and to what extent. However, it should be noted that the reasons behind why a variable reports certain values are unknown. For instance, does a project in the USA perform better than a project in Canada due to intrinsic properties within the countries, or is it because the sales manager in the USA is better at providing accurate project parameters, resulting in the project being carried out better and therefore reporting higher values?

6.3 Shortcomings of the Model

One fault of the model is that what the model is in practice predicting is how much construction tools a project will require. However, it is evaluated on how much sales HILTI made for the project. What is not certain is whether the tools for a construction project have been purchased exclusively from HILTI, as they could instead have been bought from multiple construction tool companies. This discrepancy is problematic because we are predicting overall sales for a project but evaluating based on the sales HILTI received. As a result, the effect is likely an increase in the randomness of the distribution. In other words, very similar inputs can yield quite different outputs depending on HILTI's share of the sales for certain projects.

7

Conclusion

During this thesis, a comparison between different ensemble techniques and other regression techniques for predicting project sales was conducted. A stacked ensemble learner was created to leverage the strengths of different models. The problem of predicting sales for construction projects turned out to be rather difficult. This was partly due to the imbalanced dataset and the high uncertainty of project sales. While the results show that the SVM model showed the best result based on the mean absolute percentage error, this is not an accurate representation of the model's performance, which can be seen in Figure 5.6. Therefore our conclusion is that the best models showed to be the ensemble models: LGBM, XGB, and the stack. These models showed marginal difference in performance as illustrated in Table 5.1.

From the SHAP plot in Figure 5.10 we can see that relatively few of the features have a significant impact on the model's prediction. This provides valuable insight for assessing future projects by showing what features are important to examine more thoroughly.

In the future it would be interesting to experiment further with the training process of the Meta-model in the stacked learner. More specifically it would be interesting if the output of the meta model was a vector of coefficients in how to weigh the different models rather than computing a value on its' own. This might increase the learning capabilities of the meta learner since a relatively good result can be achieved by averaging the predictions from the stack.

Bibliography

[1] J. M. Tien, "Manufacturing and services: From mass production to mass customization," *Journal of Systems Science and Systems Engineering*, vol. 20, pp. 129–154, 2011.

[2] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Frontiers of Computer Science*, vol. 14, pp. 241–258, 2020.

[3] Y. Yang, H. Lv, and N. Chen, "A survey on ensemble learning under the era of deep learning," *Artificial Intelligence Review*, vol. 56, no. 6, pp. 5545–5589, 2023.

[4] G. Brown, "Diversity in neural network ensembles," Ph.D. dissertation, The University of Birmingham, 2004, pp. 33–39.

[5] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation, and active learning," *Advances in neural information processing systems*, vol. 7, 1994.

[6] A. Mohammed and R. Kora, "A comprehensive review on ensemble deep learning: Opportunities and challenges," *Journal of King Saud University-Computer and Information Sciences*, 2023.

[7] D. Berrar, "Cross-validation," in *Encyclopedia of Bioinformatics and Computational Biology*, S. Ranganathan, M. Gribskov, K. Nakai, and C. Schönbach, Eds., Oxford: Academic Press, 2019, pp. 542–545, ISBN: 978-0-12-811432-2. DOI: https://doi.org/10.1016/B978-0-12-809633-8.20349-X. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B978012809633820349X.

[8] Y. Frayman, B. F. Rolfe, and G. I. Webb, "Solving regression problems using competitive ensemble models," in *Australian Joint Conference on Artificial Intelligence*, Springer, 2002, pp. 511–522.

[9] E. Pintelas, I. E. Livieris, and P. Pintelas, "A grey-box ensemble model exploiting black-box accuracy and white-box intrinsic interpretability," *Algorithms*, vol. 13, no. 1, 2020, ISSN: 1999-4893. DOI: 10.3390/a13010017. [Online]. Available: https://www.mdpi.com/1999-4893/13/1/17.

[10] T. G. Dietterich, "Ensemble methods in machine learning," in *Proceedings of the First International Workshop on Multiple Classifier Systems*, ser. MCS '00, Berlin, Heidelberg: Springer-Verlag, 2000, pp. 1–15, ISBN: 3540677046. [Online]. Available: https://dl.acm.org/doi/10.5555/648054.743935.

[11] G. Brown, "Ensemble learning," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2010, pp. 312–320,

ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_252. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_252.

[12] C. Zhang and Y. Ma, *Ensemble machine learning: methods and applications*. Springer, 2012.

[13] S. J. Rigatti, "Random forest," *Journal of Insurance Medicine*, vol. 47, no. 1, pp. 31–39, 2017. [Online]. Available: https://meridian.allenpress.com/jim/article/47/1/31/131479/Random-Forest.

[14] J. R. Quinlan *et al.*, "Bagging, boosting, and c4. 5," in *Aaai/Iaai, vol. 1*, 1996, pp. 725–730.

[15] C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz, "A comparative analysis of gradient boosting algorithms," *Artificial Intelligence Review*, vol. 54, no. 3, pp. 1937–1967, Mar. 2021, ISSN: 1573-7462. DOI: 10.1007/s10462-020-09896-5. [Online]. Available: https://doi.org/10.1007/s10462-020-09896-5.

[16] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992, ISSN: 0893-6080. DOI: https://doi.org/10.1016/S0893-6080(05)80023-1. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608005800231.

[17] M. Shahhosseini, G. Hu, and H. Pham, "Optimizing ensemble weights and hyperparameters of machine learning models for regression problems," *Machine Learning with Applications*, vol. 7, p. 100 251, 2022, ISSN: 2666-8270. DOI: https://doi.org/10.1016/j.mlwa.2022.100251. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666827022000020.

[18] G. Cerulli, "Model selection and regularization," in *Fundamentals of Supervised Machine Learning: With Applications in Python, R, and Stata*. Cham: Springer International Publishing, 2023, pp. 59–146, ISBN: 978-3-031-41337-7. DOI: 10.1007/978-3-031-41337-7_3. [Online]. Available: https://doi.org/10.1007/978-3-031-41337-7_3.

[19] P. P.-R. Sergio Pérez-Elizalde Blanca E. Monroy-Castillo and J. Crossa, "Hdbrr: A statistical package for high-dimensional bayesian ridge regression without mcmc," *Journal of Statistical Computation and Simulation*, vol. 92, no. 17, pp. 3679–3705, 2022. DOI: 10.1080/00949655.2022.2081968. [Online]. Available: https://doi.org/10.1080/00949655.2022.2081968.

[20] V. Vovk, "Kernel ridge regression," in *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*, B. Schölkopf, Z. Luo, and V. Vovk, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 105–116, ISBN: 978-3-642-41136-6. DOI: 10.1007/978-3-642-41136-6_11. [Online]. Available: https://doi.org/10.1007/978-3-642-41136-6_11.

[21] X. Zhang, "Support vector machines," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2010, pp. 941–946, ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_804. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_804.

[22] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization.," *Journal of machine learning research*, vol. 13, no. 2, 2012. [Online]. Available: https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf?ref=broutonlab.com.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[24] R. G. Mantovani, A. L. Rossi, J. Vanschoren, B. Bischl, and A. C. De Carvalho, "Effectiveness of random search in svm hyper-parameter tuning," in *2015 international joint conference on neural networks (IJCNN)*, Ieee, 2015, pp. 1–8.

[25] [Online]. Available: https://shap.readthedocs.io/en/latest/example_notebooks/overviews/An%20introduction%20to%20explainable%20AI%20with%20Shapley%20values.html.

[26] R. F. Barber, E. J. Candes, A. Ramdas, and R. J. Tibshirani, "Predictive inference with the jackknife+," 2021.

[27] G. M. Fitzmaurice, "Regression," *Diagnostic Histopathology*, vol. 22, no. 7, pp. 271–278, 2016, Mini-Symposium: Medical Statistics, ISSN: 1756-2317. DOI: https://doi.org/10.1016/j.mpdhp.2016.06.004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1756231716300627.

[28] "Mean absolute error," in *Encyclopedia of Machine Learning and Data Mining*, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2017, pp. 806–806, ISBN: 978-1-4899-7687-1. DOI: 10.1007/978-1-4899-7687-1_953. [Online]. Available: https://doi.org/10.1007/978-1-4899-7687-1_953.

Index