Erasmus University Rotterdam

Erasmus School of Economics

Master's Thesis Econometrics and Management Science

Forecasting project Cost Flow using Machine Learning

Dylan Buter
415000

Supervisor Dr. Maria Grith

Second Assessor Dr. Erik Kole

Abstract

This paper focuses on forecasting project cost flow using cost curves of completed projects. The paper uses Machine Learning in a time-series framework to model the cost curve as introduced by Boussabaine and Kaka (1998) and Cheng and Wu (2009). The research extends on these papers by validating the effectiveness of the methodology using several Machine Learning models. Artificial Neural Network, Support Vector Machine and Random Forest are used on a large data-set of varied projects of Royal Schiphol Group. Subsequently, an out-of-sample comparison is made based on several forecasting criteria. Overall Artificial Neural Network has the lowest error in forecasting at most stages throughout a project's lifecycle. Conversely, the errors from Support Vector Machine performed relatively poor and show the highest variance. Random Forest shows decent results but also signs of overfitting. Using Random Forest it is found that the project's budget, duration and performing company division are main cost drivers. The research shows that the Machine Learning framework has better forecasting capability than the logit model.

February 18, 2020

Contents

1 Introduction

Over the course of the life cycle of any business project, many uncertain factors affect profitability. For instance, in the construction industry these factors are highly context-dependent, explaining why the industry has the largest number of bankruptcies compared to any other sector (Boussabaine and Kaka, 1998). According to Hwee and Tiong (2002) cash flow is the main factor affecting a project's profitability. They state that the construction bankruptcies are caused by insufficient cash resources and the inability to convince creditors it is only temporary. For this reason, it is of critical importance that controllers have accurate information on cash flow trends so they can acquire financial resources to guarantee a project's success. This information can further serve as a guide for cash management or as an early warning for financial problems.

For any project-based company to achieve tighter control of its costs, it needs a quantitative model that uses historical data and project-variables to forecast project cost flow. The question is raised how reliable cash flow information provided by traditional methods is and if it can be improved by more recent models. Along common workflow, a project controller is provided with a budget estimate based on the planned activities and requirements. Then, a cost flow forecast follows by projecting the budget over time using a single-equation model. Early attempts in the literature include polynomial or logit regression models, as the resulting S-shape of the cash flow follows the theory of natural growth (Miskawi, 1989, e.g.). However, it is found that these types of models generally do not hold in practice (Balkau, 1975, e.g.). The reason being that projects are non-linearly affected by many risk factors. Since the traditional models are single-equation, they have trouble incorporating nonlinear risk factors.

To handle the non-linearity, Boussabaine and Kaka (1998) introduced Artificial Neural Network (ANN) for project management problems. As Machine Learning (ML) models such as ANN do not need to make any distributional assumptions on the data they can capture complex relations. However, some of the drawbacks of ANN models are that they are computationally expensive to train and tend to not overfit or not generalise well for new data.

Another application of ML in cost flow forecasting comes from Cheng et al. (2015), who use the Support Vector Machine (SVM) model and show its potential to improve generalisation. Nonetheless, also SVM has drawbacks. For instance, SVM can be difficult to tune without the aid of other Artificial Intelligence (AI) algorithms, which can lead to inadequate results. The last model of interest in this paper is Random Forest (RF), although this model has yet to be used in forecasting project cost flows. Even so, RF has found popularity in other research areas for its ease of use and ease of parameter interpretation.

A benefit of the ML models over traditional methods is that they are more flexible in design. Whereas, older papers use single-equation regression models that result in general cost curve equations used for all projects. Instead, this paper uses a time-series inspired framework to model cost flow using ML as in Boussabaine and Elhag (1999). In this framework cost behaviour at an upcoming project stage is forecast using the history of the same project. The research of this paper validates the effectiveness of this methodology on a large set of projects from Royal Schiphol Group. The company holds a diversified project-portfolio in businesses ranging from aviation, construction and IT.

The paper raises the question which ML model produces the most accurate project cost flow and what the main drivers behind cost movement are. Also, can estimates of individual projects be aggregated for a company's full project-portfolio? The research shows that ANN achieves the lowest error overall, although ANN shows considerably larger computation time. RF also shows decent results with signs of large overfitting. Lastly, SVM was comparatively the lowest performing model out of the three. Overall, the ML models show largely improved capability over the logit model to track the cost flow. In addition, the main drivers for cost flow variation are budget and duration of the project, moreover, the business division that develops the project also has large influence. The rest of the paper is organised as follows. First, a review of literature is given, followed by an overview of the data and the methodology used in the research of this paper. Afterwards, the empirical findings are discussed in the results section. Finally, the paper closes with a concluding remark on the performed research.

2 Literature Review

Acquiring cash flow forecasts has long been a point of interest for both researchers and project controllers. Traditional methods rely on working schedules and bill of quantities to create estimates. Given that the input information is accurate and does not change during the project, the estimated cash flow is reasonable (Lowe, 1987). However, this practice to produce estimates requires extensive time and money. Instead, a quicker and cost-effective method for forecasting is preferred.

First, a clarification of the definition for the term cash flow and its distinctions should be given. Cash flow is a business's inflow and outflow of money. Furthermore, the term 'net cash flow' is used for the difference of these two flows. This paper focuses solely on the outflow element of the cash flow, known as the 'cost flow'. Kaka and Price (1991) show that net cash flow models are generally unreliable, whereas cost flow models show higher reliability in practice. For this reason, the focus of this paper is on modelling project cost flow.

2.1 Traditional Cost Flow Modelling

Nazem (1968) introduced the idea of using averages from similar projects to create a general cash flow curve, commonly referred to as 'cost profile' in project management. The goal of the model was to predict capital requirements of new projects. Nazem argued that the model should be used for firms overall capital requirements since a construction firm usually has multiple active projects and the prediction holds better at an aggregated level. To do so, the curve is generated by overlaying data of all projects rather than fitting for each project individually. Although several firms used this methodology for forecasting, it did not become common practice since no clear consensus exists on how the average over past projects should be taken.

Research on cost profile methods became increasingly popular during the 70s and 80s (e.g.: Balkau (1975); Drake (1978); Tucker and Rahilly (1982)). At this time, the focus was on different regression models to fit cash flow. Popular forms include polynomial or logit models such that the fit has the desired S-shape. These models fit a general or nomothetic cost curve used as the forecast for all upcoming projects. Kenley and Wilson (1986) mention that these traditional regression methods are questionable since they rely on several assumptions related to regression analysis. For instance, regression assumes that shocks to the dependent variable are normally and independently distributed. However, projects suffer from systematic and idiosyncratic shocks which get lost by averaging the data. Nonetheless, these early statistical approaches were generally accepted, since within companies the estimated S-curve holds to a certain extent for different types of work. On the other hand, the approaches at the time were criticised by scientific peers. They argue that a simple mathematical equation can not capture the cash flows of real-life projects, which are likely to be non-linearly affected by multiple risk factors.

Kenley and Wilson (1986) started the discussion that projects should be studied using an idiographic methodology as opposed to a method that tries to find general curves. The reason being that each project is individual and unique, in addition, the risk factors are systemic and difficult to capture within the sample data. They also acknowledge that their idiographic methodology is meant for post hoc analysis and can not be employed for forecasting. Odeyinka et al. (2008) study the occurrence and impact of different factors causing discrepancies between construction cash flow forecasts and actuals. The study is based on a structured questionnaire given to 370 UK based contractors attempting to identify the factors. They find that the significant factors with high impact on cost flow can be grouped into the generic factors: "Changes in Design/Specification", "Project Complexity" and "Natural Inhibition". Furthermore, in Odeyinka et al. (2012) measure the impact of the risk factors on cash flow forecast using linear regression. The research shows that incorporating the risk factors in the model has the potential of improving forecasts. The authors' hint that the effect of the risk factors differ at different stages of the project and most likely have a non-linear relationship. For this reason, ML should be employed, as it can model non-linear data.

2.2 Machine Learning

The field of ML focuses on learning from data by function approximation and thus overlaps with traditional statistical methodology. Although ML is rooted in statistics, it has been influenced by other fields such as computer science. Therefor, the terminology and notation can differ from statistics. A supervised learning problem in ML uses a predefined set of inputs to predict an output using an approximated function found through a learning algorithm calibrated on training data. Alternatively, statistical methods use independent variables to forecast the dependent variable based on the joint probability distribution estimated by maximizing the likelihood of observed data. The first ML models started appearing during the 1940s, however, lack of data and computing power prevented massive breakthroughs around this time. Not until recently did ML start to resurface as computers continued to improve.

A central aspect to consider for model choice is the bias-variance trade-off. This trade-off is a property of a predictive model, where the effectiveness of the model is determined by the assumptions and complexity which should represent the underlying data generating process. Model bias stems from wrong or too strict assumptions and can lead to underfitting. A model has high variance when it is sensitive to a small variation in the training data. When the model is influenced too much by the noise in data it will be overfitted and have high variance.

One of the major benefits of ML models is that there is less need for distributional or model assumptions on observed data. However, this freedom does come with a price as illustrated by Friedman et al. (2001). The book compares the statistical model Ordinary Least Squares (OLS) with the ML model K-nearest-neighbors (KNN). These models can be considered total opposites as OLS strictly assumes a linear relation over the whole variable-space. Conversely, KNN assumes that the observations in local neighbourhoods display similar behaviour. This assumption is less stringent considering the size of the neighbourhoods can be made infinitesimally small. The KNN forecast is the average of all observations in a neighbourhood, where the neighbourhood is defined by the distance between independent variables of different observations. When the model is trained with $K = 1$ all observations in the training data are their own neighbourhood. In such KNN is able to perfectly fit a dataset without error, whereas OLS most likely contains a lot of model bias. However, KNN has a low chance of showing good results in a new dataset as the model will most likely be overfitted, the estimates will be too random and have high variance such that they are inaccurate. OLS, on the contrary, has low variance since the assumed linear structure ensures stable estimates.

The first application of ML in cash flow forecasting stems from Boussabaine and Kaka (1998). They propose to use ANN to deal with the complex non-linear mapping of input variables to the predictions. ANN is modelled after the brain and processes information similarly using a mathematical network of neurons and synaptic links. The variables enter the network from the input layer and flow through the neurons like nerve signals. At each neuron, the signal is processed using a predefined activation function and at the final layer, a forecast is made. By varying the network-structure, activation functions and optimization algorithm the model can be trained to perform a variety of complex tasks. Cheng and Wu (2009) introduce the use of SVM in estimating project cash flow. They combine SVM with the fast messy Genetic Algorithm (fmGA), an AI method used to efficiently find hyperparameters. The paper shows the potential of SVM to estimate cash flow. In essence, SVM maps a set of inputs into a higher-dimensional feature space using a predefined kernel function. As SVM aims to find the hyperplanes that best segregates the output. The found hyperplanes are known as support vectors, which are used to forecast for new projects.

RF became popular because of its simplicity in interpretation and implementation. However, the model has yet to be applied for project cost flow forecasting. Breiman (2001) proposes the RF model and shows its effectiveness in non-linear regression problems using various datasets. The model tends to have low error relatively insensitive to the two tuning parameters of the model (Segal, 2004). RF combines decision trees and bagging, where each tree is trained on a bag of data. Then the individual tree estimates are combined into a forecast by summarizing. Salas-Molina et al. (2017) showed RF improved the forecast accuracy on simulated company cash flow, which lead to higher cost savings due to better allocation of the company's financial resources.

Further extensions on ML in cash flow forecasting have been made by combining AI algorithms into hybrid models, e.g. the combined SVM and fmGA model in Cheng and Wu (2009). Another hybrid combination is SVM and Fuzzy Logic as in Cheng and Roy (2011). Fuzzy logic is used to address complex uncertainties and ambiguity in cash flows caused by non-probabilistic factors. However, these hybrid models are out of the scope of this paper. Instead, the focus is on an empirical comparison of ANN, SVM and RF in project cost flow forecasting using a large dataset of company projects. Given that older papers generally use small sets of projects to base their research on.

Kumar and Thenmozhi (2006) made an empirical comparison on the ability of ANN, SVM, RF and other statistical models on predicting stock index movement. They state that algorithms like ANN tend to get stuck in locally optimal solutions. Although the solution gives good in-sample fit for even extreme cases in the training data, the model tends to overfit and not generalize well for new data. Overall, they find SVM to have the best performance as the model generalizes the best due to the structural risk minimization principle. Caruana and Niculescu-Mizil (2006) show a comprehensive evaluation of the 10 most prominent ML algorithms including ANN, SVM and RF. The results are drawn from multiple binary classification problems where several evaluation metrics are employed. They find that a universally best algorithm does not exist considering all the investigated use-cases and metrics. The ensemble learning models Boosted Trees, Bagged Trees and RF performed best overall, however on some problems they perform worst. The latter result hinting that the optimal model is problem-oriented.

3 Data

Transaction data of completed Schiphol projects are used for forecasting cost flow curves. Besides the construction industry, the issue of uncertain cash flows is present for any company that engages in high capital expenditure projects. Especially a project-based company such as Royal Schiphol Group faces high risk stemming from project cash flow. The airport aims to become the main hub of Europe as well as achieve sustainable company growth that keeps negative environmental effects under control. To accomplish this feat, Schiphol has a large investment budget for projects to improve the airport and its operations. Throughout 2018, Schiphol invested a total of 581 million euro on capital expenditures from its project-portfolio. The projects range from areas within business, construction and IT.

The data used in the research is collected from the ERP systems of Schiphol (Oracle and SAP). The data contains all transactions on projects over the period 01-01-2005 to 06-05-2019. In total, the data contains 364,592 daily total-transactions on 1,999 closed projects. Furthermore, the transaction data is complemented with explanatory variables that classify the type of project or measures a project's complexity. Table 1 shows the variables used in building the models and table 2 shows their descriptive statistics and distribution.

Variable	Description
Business Area	Classifies projects to different Schiphol organizations.
Project Category	Categorizes the project on the type of product delivered.
Project Classification	Classifies projects based on project controllers judgment on project size and complexity, descending from A to D.
Project Duration	Duration of the project in days till delivery
Task Amount	Amount of tasks necessary to deliver the completed project.
Vendor Amount	Amount of vendors involved in the project.

Table 1: List of explanatory variables used in the research.

Project Category	Business Area	Project Classification
549 Aviation	568 Business	28 A
99 Consumers	249 Construction	82 B
25 Direction	1182 IT	297 C
1258 IT		87 D
68 Real Estate		1505 None

Variable	Mean	St. Dev.	Min	Max
Budget	€ 1,103,083	€ 5,159,874	€ 798	€ 117,147,995
Project Duration	642 days	549 days	15 days	4822 days
Task Amount	6	7	1	102
Vendor Amount	5	8	1	121

Table 2: The descriptive statistics and distribution of the explanatory variables.

The data consists of small to large projects, where the smallest project has a total budget of € 798, the largest project € 117 million and the average project having a budget of around € 1.1 million. In terms of duration, the average project is around 1.5 years, with the longest project spanning 13 years and the shortest project being only 15 days. Furthermore, task and vendor amount are also used to quantify the size of the project since having more supplying vendors or having more distinct steps to complete the project makes the cost flow more complex. The quantitative variables are standardized using the z-score such that these variables are on a similar scale.

The categorical or nominal data includes business area, project category and project classification. The first variable shows the company division which develops and completes the project, the variable ranges from: Aviation who work on project directly with the airlines i.e. runways or hangars; Consumers who work on projects that include airport shops or parking spaces for long term travellers; Direction who develop projects that affect the whole of Royal Schiphol Group; IT who develop the wiring infrastructure or online/digital opportunities; and Real Estate who built and rent out offices on the Schiphol terrain. The second categorical variable shows the type of product that the project delivers, construction in case of a physical structure such as an office building or new road, IT for projects with a large digital component and business for all other types of projects. Lastly, project classification is a complexity ranking given by project controllers based on the financials and risks involved. However, the ranking is badly maintained or missing for around 75% of the projects. To make use of the categorical variables in modelling, they are transformed into binary dummy variables through one-hot encoding. This method is used since ML models can only work on numerical variables. Given that there is no natural ordering among the categorical variables a simple mapping to a sequence of natural numbers will not suffice as this would imply a natural order among the different categories. Instead, one-hot encoding creates binary dummy variables for all but one of the classes within a categorical variable. Introducing a dummy variable for each class will result in a problem known as the dummy variable trap. In case an intercept and a dummy variable for each class are used, the sum of all dummy variables will always be equal to the intercept and thus creating perfect multicollinearity in the data.

The cost curves are created from the transaction data in several steps. First, the data is cleaned from missing values and manual cost-corrections. Second, the set of projects are filtered from ongoing, unrealised and outlier projects. Third, the transaction data is accumulated to create time-series of project cost. In the research cost spent scaled by realized cost is used as the dependent variable which needs to be forecast. Furthermore, each project will be split in $n = 19$ periods or project time-points, with each point denoting 5% of project progress with the end period having 100% cost.

This framework is chosen so that the ML models are applicable to any project with different duration. Furthermore, the cost is scaled by the realized cost due to lacking data quality of project budgets. Initial budget estimates are generally ambiguous numbers to get projects started and badly maintained over time. The underlying reason being that project duration and total cost are subject to a lot of uncertainty. This is illustrated by Yang (2005), who describes a project consisting of a set of decisions and activities that are constrained by time and cost. The goal of a project planner is to find the optimal decisions and activities that lead to a desirable outcome. However, since these decisions and activities are subject to financial, market, weather and political uncertainty so are the resulting project cost and duration. The traditional cost flow models also suffer from this uncertainty as they assume total cost and duration to be known. Kaka (1999) shows how to incorporate this uncertainty by analyzing significant discrepancies between project actuals and simulated cost commitment curves. However, this research focuses on which model can forecast cost flow and what the main drivers are behind cost curve variation. Further research has to be done on combining the methodology of Kaka (1999) with the ML models used in this paper.

In the last step, cost curves at the project time-points are interpolated from the cumulative project costs using a Piecewise Cubic Hermite Interpolating Polynomial (PCHIP). The method is chosen because of its shape-preserving feature such that the interpolated curve does not overshoot the data. Besides, the first-order derivative of PCHIP is continuous ensuring a smooth curve. Moreover, the research of this paper is done using R programming and the 'caret'-package is used for its built-in features to estimate optimal (hyper)parameters.

4 Methodology

Given the set of independent variables denoted as $x_i \in \mathbb{R}^P$ and project cost as dependent variable $y_{i,t} \in \mathbb{R}$ of project i at time t. These variables are assumed to have an underlying relationship. The goal is to approximate this relationship as a function $f(x_i, t; \theta)$. Given parameters θ, the function predicts $y_{i,t}$ with some error $\varepsilon_{i,t}$ that is independent of x_i and has an expectation of zero, as in equation (1).

$$y_{i,t} = f\big(x_i, t; \theta\big) + \varepsilon_{i,t} \tag{1}$$

Given that ε is expected to be zero, the statistical approach shows that $f(x_i, t; \theta) = \mathbb{E}(y_{i,t}|x_i = x)$, the conditional expectation of $y_{i,t}$ given the information that x_i is some x. One of the most powerful statistical tools to solve this problem is Maximum Likelihood. This method finds the parameters θ of the model by optimizing $P(y_{i,t}|x_i, \theta)$. By making assumptions on the distribution of $\varepsilon_{i,t}$ and assuming a function-form $f_\theta(x_i, t)$, the log likelihood formula is constructed and maximized over θ.

As mentioned in the literature review ML has also received influence from different scientific fields than mathematics and statistics. However, in essence the same problem is solved: approximating the underlying function between observed independent and dependent variables. ML learns θ in $f(x_i, t; \theta)$ through approximation using a learning algorithm on the observations in a training set. However, since equation (1) is most likely non-linear the ML approach might be more appropriate for cash flow forecasting as the exact functional form is unknown. The model's performance depends on the ability of the respective learning algorithm to infer a functional form that should hold for new projects as well.

4.1 Logit Cost Flow Model

To compare the results of ML models to the traditional technique for forecasting the cost flow curve, the logit model is used as a benchmark. The logit model was originally introduced as the S-shape shows similarity to growth patterns found in economics. The model is described by Kaka (1999) with the equation as follows

$$f_{\text{Logit}}(t) = \frac{e^{\alpha}\left(\frac{t}{1-t}\right)^{\beta}}{1 + e^{\alpha}\left(\frac{t}{1-t}\right)^{\beta}}, \quad t \in [0,1] \tag{2}$$

Here, t denotes project completion time. Furthermore, α and β are the model parameters that need to be estimated. The part $e^{\alpha}\left(\frac{t}{1-t}\right)^{\beta}$ can not be negative since t ranges from 0 to 1 and the output of an exponential function is always positive. Therefor, the model ensures that the y estimate of equation (2) is also bounded between 0 and 1. This means that no technical conditions are needed on α and β, which makes the estimation procedure quite straightforward. Through a logistic transformation of the variables, the parameters can be estimated with standard OLS. The transformed model can be rewritten as

$$\begin{aligned} &v = \alpha + \beta w, \\ &\text{where} \\ &v = \ln \frac{y}{1-y} \text{ and } w = \ln \frac{t}{1-t} \end{aligned} \tag{3}$$

Furthermore, the original quantities of y and t can be found by inverting the formulas for v and w. The model equation is estimated for each project on grouped project data. Nazem (1968) proposed that the grouped data should be created by averaging costs from projects within a group. Since it remains unclear from literature how to select projects for averaging, three different methods are introduced for the overlaying process. The first is random selection, where all the data from a random set of projects is averaged. The projects in this set are selected at random without replacement. Furthermore, project-sets are created with sizes of $k = 2, \ldots, 500$. The second method is called quantitative selection and is inspired by

KNN. In this method data from the k closest projects is averaged, with closeness measured in terms of euclidean distance as in equation (4). The idea behind this practice is that similar projects have similar cost flow behaviour and therefor lower error when using those projects to forecast the cost curve.

$$d(i,j) = \sqrt{\sum_{m=1}^{p} (x_{i,m} - x_{j,m})^2} \tag{4}$$

The equation above shows the distance between projects i and j, with $x_{i,m}$ and $x_{j,m}$ denoting the value of variable x_m from project i and j respectively. Moreover, the sum is taken over all quantitative variables. With the distance calculated as equation (4) there is a risk that one or a few variables dominate the distance measure. To ensure equal importance, they are standardized using the z-score: $z = \frac{x - mean(x)}{stdev(x)}$. The last method is called combined selection. This method uses the previous distance measure and adds a penalty term λ based on the qualitative variables as

$$d^*(i,j) = d(i,j) + \lambda \sum_{r=1}^{q} 1_{x_{i,r}}(x_{j,r}) \tag{5}$$

The function $1_{x_{i,r}}(x_{j,r})$ denotes the indicator function that is equal to either 1 or 0 based on the condition:

$$1_{x_{i,r}}(x_{j,r}) = \begin{cases} 1, & x_{i,r} \neq x_{j,r} \\ 0, & x_{i,r} = x_{j,r} \end{cases} \tag{6}$$

The equation above is equal to 1 when projects have different values for the respective qualitative variable. The goal of the research is not to find the optimal λ on which to select projects for averaging. Instead, the goal is to show that using projects from the same categories have similar cost flow patterns. Thus λ is set to 1. This value ensures that the influence of qualitative and quantitative variables is of similar scale.

4.2 Modelling Cost Flow with Machine Learning

The ML models in this paper are built along with the recent approach of Boussabaine and Elhag (1999) and Cheng and Wu (2009). Inspired by time-series models such as the Auto-Regressive Exogenous (ARX) model, the model in this paper uses projects own past to predict future cost movement. Conversely, older papers such as Nazem (1968) attempt to find general cost curves from past projects. Since project duration varies from project to project, the model in this paper differs from traditional AR models. The reason being that the time interval between subsequent time-periods depends on the duration of project i. As mentioned in the data section, the curve of each project is split at $n = 19$ time-points. The resulting time interval is denoted as $\Delta t = 5\%$. In this paper, a lag of 3 periods is used to forecast the expenditure at an upcoming period as follows

$$y_{i,t} = f\left(y_{i,t-\Delta t},\ y_{i,t-2\Delta t},\ y_{i,t-3\Delta t_i},\ x_i; \theta\right) + \varepsilon_{i,t} \tag{7}$$

where $y_{i,t-\Delta t},\ y_{i,t-2\Delta t},\ y_{i,t-3\Delta t_i}$ denote the cost flow at the three previous periods. Figure 1 shows an example of how past cost flow information is used to predict the curve.

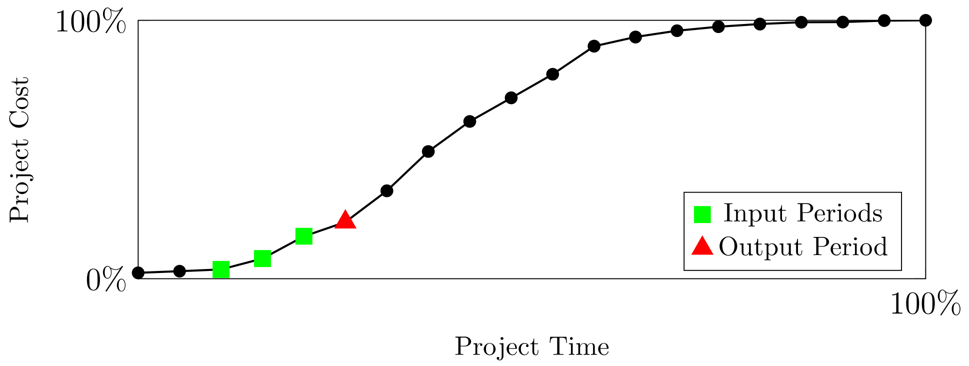

Figure 1: An example of the time-series framework, where three subsequent periods of a project are used to forecast an upcoming period.

The functional approximations of ANN, SVM and RF are discussed in the following sections of the methodology. Furthermore, the project and time subscripts i and t respectively are left out for ease of notation. Moreover, the set of all input variables is denoted by x.

4.3 Artificial Neural Network

As mentioned in the literature review, ANN is modelled after the human brain hence consists of mathematical neurons and synapses organised in different layers. A typical network consists of an input layer, one or more hidden layers and an output layer. A diagram of a single hidden layer network is shown in figure 2.

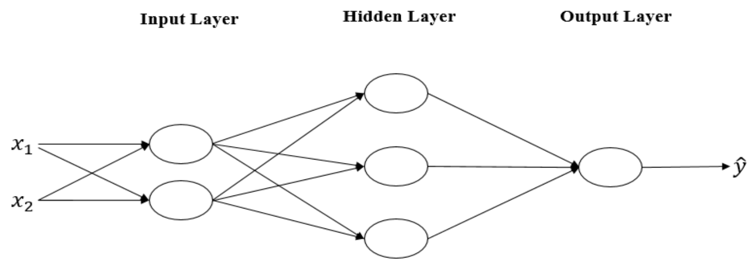

Figure 2: Example of an ANN diagram

The variables enter the network through the input layer and flow like signals through the nodes and synapses. The nodes process the signals via an activation function and send the output to nodes in subsequent layers. The activation function used in this paper is the logistic function, shown in equation (8). This function takes any input denoted by v and transforms it accordingly. The logistic function is chosen as it enables the model to identify nonlinear relationships. In addition, the function binds output between $0 - 1$ and has smooth form compared to threshold functions with sharp borders between outputs of either 0 or 1.

$$s(v) = \frac{1}{1 + \exp(-v)} \tag{8}$$

The following notation of ANN follows Friedman et al. (2001) notation of a single layer network and is extended to accommodate for multiple layers. ANN can be considered a two-step regression. The first layer transforms the independent variables x into Z as in equation (9). The nodes in the subsequent layers take the output of previous layers as inputs. With the previous layer output denoted as $Z^{l-1} = (Z_1^{l-1}, \ldots, Z_{M^{l-1}}^{l-1})$, the nodes in the subsequent layers transform their inputs according to equation (10).

$$Z_m^1 = s({\alpha_m^1}^T x), \qquad m = 1, \ldots, p \tag{9}$$

$$Z_m^l = s({\alpha_m^l}^T Z^{l-1}), \qquad m = 1, \ldots, M^l, \quad l = 1, ..., L \tag{10}$$

Where the scalar Z_m^l denotes the output of node m in layer l. Furthermore, L is the total amount of hidden layers and M^l denotes the number of nodes in layer l. The dimension of Z^l depends on the number of nodes in the layer, for example the first layer has a node for every explanatory variable such that $Z^1 \in \mathbb{R}^p$. Furthermore, the parameters or model weights that need to be estimated is the slope α_m^l which controls the rate of node activation. The effect of an intercept can be created by adding a constant of 1 as a variable. The second step of the regression combines the derived Z^L of the final hidden layer into an estimate of y as

$$f_{\text{ANN}}(x) = \beta_0 + \beta^T Z^L \tag{11}$$

When the set of variables x enter the network they are nonlinearly transformed, which makes the interpretation of model-parameters α and β difficult. Friedman et al. (2001) state "the model is especially effective in problems with a high signal-to-noise ratio and settings where prediction without interpretation is the goal". Furthermore, the model's hyperparameters are the number of layers and neurons, which are found through experimentation. A network better has too many hidden nodes than too few, since a network with too few nodes may underfit (Fletcher et al., 1998). However, having a model with an extreme number of nodes increases the chance of overfitting and the time to train the network significantly. Typically one hidden layer is enough for the majority of problems. Since one layer can approximate any function with a continuous mapping from one finite space to another and a second layer is only beneficial in some cases. For this reason, only one and two hidden layer(s) networks are built in the research of this paper. Furthermore, a rule of thumb for the number of neurons in the hidden layer is that it should be between the size of the input layer and the output layer.

The optimal parameters are found by minimizing the network error E, since the goal of the model is to match the forecast output with the dependent variable y with little error. The parameters are iteratively updated according to the Resilient-Propagation (RPROP) algorithm of Riedmiller and Braun (1993). This method is chosen over Back-Propagation (BP) since it has been reported to have better convergence speed, robustness and accuracy over the latter. Let θ represent the vector with all parameters included in the model, given the set of all observations denoted by $\{(x_1, y_1), \dots, (x_N, y_N)\} \subset \mathbb{R}^P \times \mathbb{R}$ the sum of squared errors is taken to optimize model fit:

$$E(\theta) = \sum_{i}^{N} \big(y_i - f(x_i)\big)^2 \tag{12}$$

Starting with random parameter initialization, the network is iteratively trained by calibrating the parameters. The algorithm stops when E converges and becomes smaller than some predefined threshold C. Stopping early by setting a threshold prevents the model to overfit, which can happen when stopping at the global minimum of $E(\theta)$. The parameter updating steps of each iteration r are shown in equations (13). These updating steps follow from the derivatives of E with respect to the model parameters, this approach is known as the generalized delta rule.

$$\begin{aligned} \beta_m^{(r+1)} &= \beta_m^{(r)} - \Delta\beta_m^{(r)} \\ {\alpha_{m,j}^{l}}^{(r+1)} &= {\alpha_{m,j}^{l}}^{(r)} - \Delta{\alpha_{m,j}^{l}}^{(r)} \end{aligned} \tag{13}$$

Here β_m is the weight assigned to the output of node m in the final hidden layer, $\alpha_{m,j}^l$ is the weight in vector α_m^l that is assigned to the output of node j in the layer $l-1$. The partial derivative with respect to β_m is straightforward from plugging equation (11) into (12) and taking the derivative as follows

$$\frac{\partial E_i}{\partial \beta_m} = -2\big(y_i - f(x_i)\big) z_{m,i}^L \tag{14}$$

Here $z_{m,i}^L$ is the output of node m in the final hidden layer L for observation i. The partial derivatives with respect to $\alpha_{m,j}^l$ is less intuitive, since for a node in layer $1 \leq l < L$ it depends on the results of derivatives in subsequent layers. The formulas for the derivatives of final hidden layer L nodes and nodes in other layers l are shown in equations (15) and (16) respectively, the derivations of which are in the appendix.

$$\frac{\partial E_i}{\partial \alpha_{m,j}^L} = -2\big(y_i - f(x_i)\big)\beta_m s'\big(v_{m,i}^L\big) z_{j,i}^{L-1} \tag{15}$$

$$\frac{\partial E_i}{\partial \alpha_{m,j}^l} = s'(v_{m,i}^l) z_{j,i}^{l-1} \sum_{k=1}^{M^{l+1}} \delta_{k,i}^{l+1} \alpha_{k,m}^{l+1} \tag{16}$$

Here, $v_{m,i}^l = {\alpha_m^l}^T Z_i^{l-1}$ denotes the input for node m before it enters the transformation function. $\delta_{k,i}^{l+1}$ is known as the error from the current model and is formally defined as $\delta_{m,i}^l = \frac{\partial E_i}{\partial v_{m,i}^l}$. Furthermore, since the logistic function is used $s'(v)$ can be written as $s(v)(1 - s(v))$. In RPROP the parameter update steps depend on the sign of the partial derivative in the current step and in previous steps. RPROP is therefor known as an adaptive learning algorithm. Let $w \in \theta$ denote a parameter in the full set of model parameters. RPROP updates w with Δw, which is calculated according to the rule:

$$\Delta w^{(r)} = \begin{cases} \eta^+ * \Delta w^{(r-1)}, & \text{if } \frac{\partial E}{\partial w}^{(r-1)} * \frac{\partial E}{\partial w}^{(r)} > 0 \\ \eta^- * \Delta w^{(r-1)}, & \text{if } \frac{\partial E}{\partial w}^{(r-1)} * \frac{\partial E}{\partial w}^{(r)} < 0 \\ \Delta w^{(r-1)}, & \text{else} \end{cases} \tag{17}$$

Where $0 < \eta^- < 1 < \eta^+$, for which the original author suggests using values of $\eta^- = 0.5$ and $\eta^+ = 1.2$. The logic behind the equation is that the step size increases when the sign of the partial derivatives stays the same in consecutive iterations and decreases when they are opposite. Because of this feature, the algorithm can accelerate or slow down to locate the minimum quicker. The change in weight is either added or subtracted as in equation (13), depending on the sign of the derivative $\frac{\partial E}{\partial w}^{(r)}$ such that E effectively decreases. The pseudo-code for the full RPROP algorithm can be found in the appendix.

Parameter estimation for ANN proceeds as follows, the first iteration starts with random parameter initialization after which the parameters are iteratively updated until the error converges below the threshold. Each iteration consists of a forward phase and a backward phase. During the forward phase, the model parameters are fixed and the model output $f(x_i)$ and the derived Z are computed. The propagation equations (15) and (16) are then used to calculate the model errors $\delta^l_{m,i}$ starting from the final hidden layer back to the first layer during the backwards phase. These values are then used to calculate the partial derivatives used for the updating steps of equation (13) along with the rules of the RPROP algorithm.

4.4 Support Vector Machine

SVM is first introduced by Cortes and Vapnik (1995) and used primarily for classification problems. The goal of the method is to find a hyperplane in the explanatory variable space that best segregates the different classes. This is done by setting a linear boundary in a higher-dimension space, which translates into a nonlinear boundary in the original space. More recently, the model has been extended to regression problems by use of a loss function. The model found popularity because of better out-of-sample results compared to ANN for some problems. SVM is based around the structural risk principle, which focuses on restricting generalisation error. This error is a combination of estimation error and the error stemming from poor model choice (Gunn et al., 1998). This feature makes SVM less likely to overfit than ANN, which solely attempts to minimize estimation error. Furthermore, the following notation of SVM using linear basis expansion functions follows that of Friedman et al. (2001). SVM uses a loss function on ε as in equation (18). This measure means that the model is robust to errors of at most ϵ and only penalizes large residuals.

$$L(\varepsilon) = \begin{cases} 0 & \text{if } |\varepsilon| \leq \epsilon \\ |\varepsilon| - \epsilon & \text{otherwise} \end{cases} \tag{18}$$

$$f_{\text{SVM}}(x) = \sum_{m=1}^{M} \beta_m h_m(x) + \beta_0 \tag{19}$$

SVM searches for $f(x)$ with the form shown in equation (19), where the set of linear basis expansion functions h_m translate x into a higher variable dimension. Finding the optimal $f(x)$ comes down to solving the minimization problem shown in equation (20).

$$\min_{\beta,\beta_0} \frac{1}{2}\sum \beta_m^2 + S\sum_{i=1}^{N} L\big(y_i - f(x_i)\big) \tag{20}$$

Here the term $\sum \beta_m^2$ is added to keep the model parameters small and to ensure SVM does not overfit. SVM makes a trade-off between model complexity and in-sample fit, as it attempts to find a simple model with small β such that it generalises well for new data. This trade-off is controlled by the predefined constant $S > 0$, where large S means more emphasis on reducing the estimation errors. Using Lagrange multipliers it can be shown that given any choice of $L(\varepsilon)$ the minimization problem has the solution:

$$\hat{f}_{\text{SVM}}(x) = \sum_{i=1}^{N} \hat{\alpha}_i K(x, x_i) \tag{21}$$

Here α_i is some function of $\hat{\beta}$ and $K(x_i, x_j) = \sum_{m=1}^{M} h_m(x_i)h_m(x_j)$ is the kernel function in SVM. Since the kernel only depends on the inner product of the basis functions it is not necessary to compute or specify all $h_1, \ldots, h_M$. The kernel used in this paper is the Radial Basis Function (RBF):

$$K_{RBF}(x_i, x_j) = \exp\Big(-\sigma\|x_i - x_j\|^2\Big) \tag{22}$$

A detailed explanation of the linear decomposition of the RBF kernel can be found in Friedman et al. (2001). Furthermore, other popular kernels include the Polynomial and the Sigmoid kernel shown in equations (23) and (24) respectively. However, RBF is chosen because of its flexible nonlinear form. Moreover, RBF has only one parameter σ.

$$K_{POL}(x_i, x_j) = (1 + \langle x_i, x_j\rangle)^d \tag{23}$$

$$K_{SIG}(x_i, x_j) = \tanh(\kappa_1\langle x_i, x_j\rangle + \kappa_2) \tag{24}$$

The parameters of SVM are estimated by optimizing equation (20) in terms of β and β_0. However, the performance of SVM is strongly dependent on the hyperparameters σ and S. Since nonlinear RBF is used in this paper, standard Lagrange form solutions are not available. Instead, a grid-search is used to find the optimal hyperparameters.

4.5 Random Forest

The underlying idea of RF is that of model averaging. This method involves averaging a lot of unbiased but noisy models, such that a forecast is made with lower variance than that from the individual models (Dormann et al., 2018). Breiman (2001) introduces the model and shows using the law of large numbers that the generalisation error converges sufficiently small as the forest grows larger. Another benefit which popularised the model is the lack of hyperparameter tuning needed. Furthermore, RF is easy to interpret where the strength of relations between independent variables and dependent variable can be quantified through the Gini-index, whereas the inner-workings of ANN and SVM are typically 'black-box'.

RF is made of many individually estimated Regression Trees (RT). A single RT represents a set of hierarchic if-then conditions that parts the data based on optimal splits among the independent variables. The tree is constructed using an tree modelling algorithm, a example of an RT is shown in figure 3.

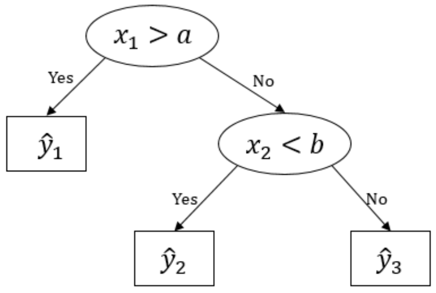

Figure 3: Example of a Regression Tree

The nodes represent the if-then conditions, which split the data into groups until a terminal node is reached and the tree forecast is made. RT models have a low bias in-sample since the trees can be grown sufficiently deep without regularization. However, these type of trees are generally noisy and have high variance estimates. RF makes use of B different trees by averaging the estimates, as in equation (25), where T_b denotes the b-th tree. Each RT is trained using a random subset of the data created through bagging. In short, bagging creates B different datasets by uniformly sampling with replacement from the original data.

$$f_{\text{rf}}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x) \tag{25}$$

Through bagging each RT is trained on the same data, therefor the outcome of each tree is identically distributed. This means that the averaged estimate of all trees holds the same bias as an estimate of an individual tree. Conversely, the variance of the averaged estimate can be reduced by increasing the number of trees B. The formula for the variance of the RF estimate is as follows

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 \tag{26}$$

Here ρ denotes the (positive) correlation between the individual trees. The formula shows that variance decreases when B increases, with the limit being the first term $\rho\sigma^2$. This term shows that the variance reduction is optimal when the correlation between the trees is low.

The tree modelling algorithm searches for the optimal tree by finding the best splits over the independent variables. The goal of the algorithm is to create homogeneous regions of the variable space, where within regions the distribution $f(y|x)$ is independent of the variables and thus 'pure' (Gokhale and Lyu, 1997). The algorithm iteratively splits the data until a predefined stop condition is reached and a node becomes a terminal node. To keep the cross-correlation ρ in equation (26) low, during each iteration only p^* out of all p independent variables are randomly selected to be considered for the splitting condition. The algorithm searches for the optimal splitting condition s among the randomly selected variables by maximizing the decrease of impurity:

$$\Delta I_t^s = I_t - I_{t_l} - I_{t_r} \tag{27}$$

Here I_t denotes the impurity at node t. Furthermore, t_l and t_r denote the left and right child node respectively. Impurity is measured by the deviance of the group of observations flowing through a given node. Where deviance is defined as the distance of observation y_i

and the population mean $\hat{\mu}$, as in equation(28) and (29) for calculating the deviation and impurity respectively.

$$D(y_i) = \big(y_i - \hat{\mu}\big)^2 \qquad (28) \qquad\qquad I_t = \sum_i D(y_i) \qquad (29)$$

Here in equation (29) the sum over i represents the group of observations that flow through a given node. At each node, all possible splits given the input variables are considered. A node becomes a terminal node if the amount of observations reaches the minimum size m and can not split further as the child-nodes will have a size lower than m. In essence, the model parts the data based on the independent variables such that similarly distributed groups of y observations are created. Afterwards, the forecast follows from the average of the similarly distributed groups of data that finish at a given terminal node.

As mentioned before, RF can determine variable importance in terms of the Gini-index. This measure is a sum over all trees and nodes of the decrease in impurity by all splitting conditions on the given explanatory variable. Given that t_b denotes a node of the b-th tree and $I_{t_b}^{s_i}$ the splitting condition on x_i at this node, the index is calculated as

$$GI_{x_i} = \sum_{b=1}^{B} \sum_{t_b} \Delta I_{t_b}^{s_i} \qquad (30)$$

The RT modelling algorithm is greedy in growing the tree since it only looks at the decrease in impurity and does not take overfitting into account. Therefor, the individual tree models tend to have high variance, which is why RF improves the individual RT models by averaging them. According to the original authors of the model, best practice values for p^* is $\lfloor p/3 \rfloor$ and a minimum node size of $m = 5$. By setting m sufficiently large, the size of the trees can be controlled such that they do not overfit. By controlling p the cross-correlation between different trees can be kept low. Other than these two parameters, only the amount of trees B has to be tuned. However, from equation (26) it can be concluded that B should be set high to optimize the variance reduction. Therefor, RF only has two real tuning parameters, which are m and p^*. The pseudo-code for the RF algorithm can be found in the appendix.

4.6 Model Evaluation

To evaluate model performance, the data is split into two sets of in-sample and out-of-sample projects. The model parameters are estimated on a random subset of 75% of the projects and tested for out-of-sample performance on the remaining 25%. By measuring the model fit out-of-sample, it is shown which model generalises the best to new projects. The model fit is calculated in terms of Mean Absolute Error (MAE) and Root Mean Square Error (RMSE):

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{31}$$

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{32}$$

The difference between the measures is that the absolute error is used to measure the bias of the model on the same scale of the data. RMSE also measures the bias of the model. However, RMSE is derived from the second moment of the error, thus incorporates both the bias and variation of the model. A relative high RMSE indicates more variance among the errors or worse fit for the extreme projects within the data.

The optimal hyperparameter specifications for each model are found through a grid search in combination with 5-fold cross-validation (CV). The grid search loops through the different model specifications and CV estimates the out-of-sample predictive power of each model. This estimate is used to select optimal hyperparameters. K-fold CV splits the data into K equally sized groups. Afterwards, the model parameters are estimated K times where each time one of the groups is left out and used to estimate MAE. A less biased estimate of out-of-sample performance is constructed by averaging the MAE of the K groups. This way the models are built using only in-sample projects based on forecasting potential and the remaining projects can be used for testing.

5 Results

The result section starts with a discussion of the realised cost flow curves of the projects in the dataset and the performance of the benchmark model, the logit model. Afterwards, the ML models are built. It is shown how the optimal hyperparameters are chosen for ANN, SVM and RF. Lastly, an out-of-sample comparison is made between the resulting optimal models. A plot of the cost flows from a random subset of 50 projects is shown in figure 4.

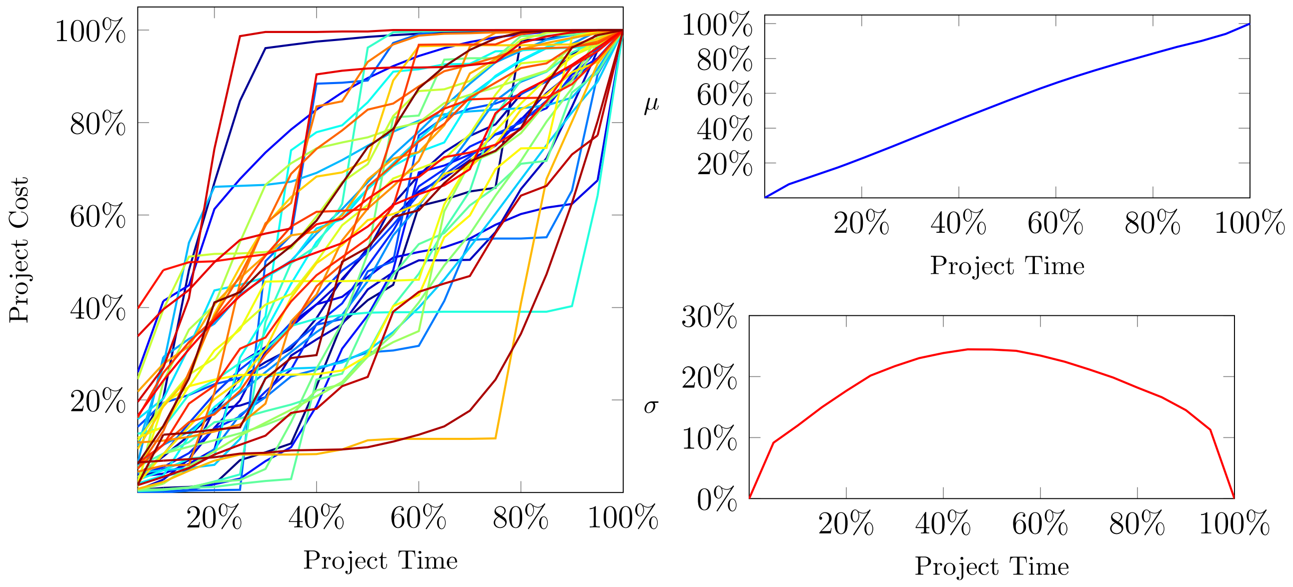

Figure 4: The left plot shows the realized cost flow curves of a random subset of 50 projects. In addition, the right plots show the mean μ and standard deviation σ of the cost curves over time.

The left plot shows that the cost curve does not always display an S-shape. Some of the lines have a more linear shape and others have long periods of no activity resulting in flat lines and sharp corners. The right plot shows that the average curve is close to linear. Moreover, the figure shows high standard deviation during the middle stages of a project. Meaning that cost flow can be quite dissimilar as seen from the 50 curves. This result supports the findings of papers as Kenley and Wilson (1986). Who state that searching for general curves is futile and can not be used for forecasting as project inherent factors drive the variation of cost curves between projects. Therefor, the time-series framework could prove more useful. Since this framework directly uses a project's own past to forecast cost flow.

5.1 Logit model

The logit model uses averaged data from a group of completed projects to estimate the cost curve of an upcoming project. A visual example of how this is done is shown in figure 5.

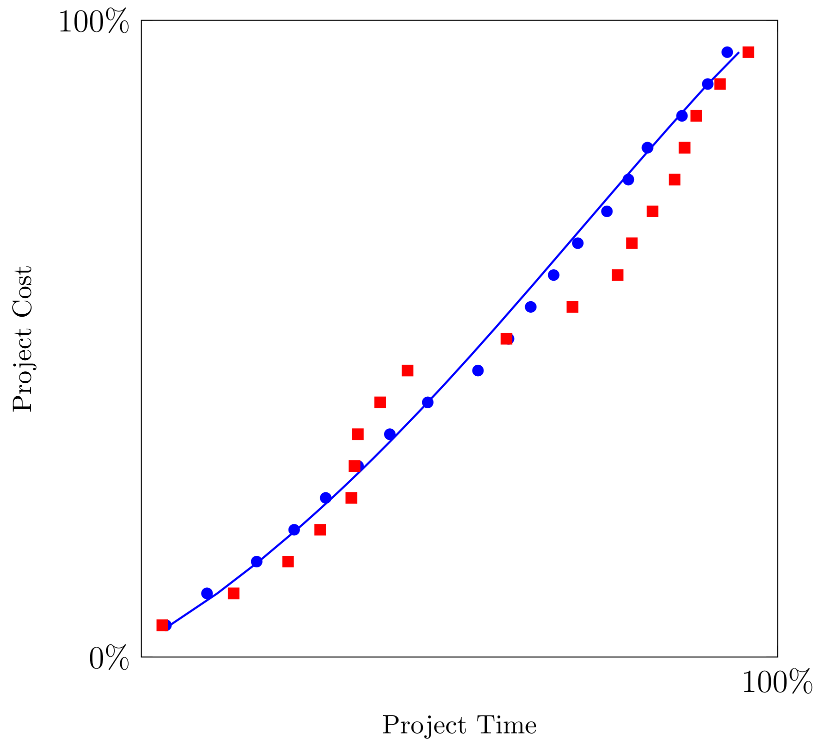

Figure 5: Example of using averaged data from other projects to estimate the logit cost curve of one project. With the averaged data denoted by the blue circles and the project data denoted by red squares.

In the plot above, the blue squares denote the averaged project-data. This data is used to fit the logit curve and serves as a forecast of the red squares, which are the realized data-points of the new project. Three methods are introduced to select projects for averaging. The first method is random selection, the second method is quantitative selection and only makes use of the quantitative variables. Finally, the third method uses all variables to select projects and is named combined selection here. The results of using the three methods with varying amounts of projects averaged k, is shown in figure 6.

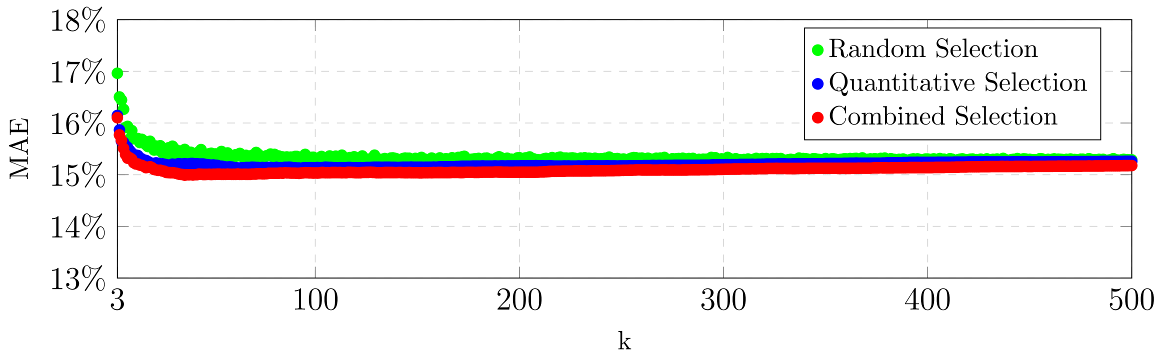

Figure 6: Plot of the MAE averaged over all projects, by forecasting the cost curve using averaged data of past projects and different numbers of k, the amount of averaged projects.

The plot shows a clear benefit in using quantitative and combined selection compared to selecting projects at random. For these methods, MAE is lower over the whole range of k. Furthermore, for low k these methods show a significantly smaller error. Random selection shows a pattern where the biggest estimation errors are made for low k and MAE converges to an average of 15.30% when k grows large, which is around $k = 200$. The other two methods show a global minimum when k is small. Only when k grows large, the errors of these methods start to rise to a similar error as random selection. The combined method reaches the lowest MAE overall with 14.90% at $k = 38$. This suggests that using similar projects does improve the cost curve forecast slightly. Especially when a few past projects are available for averaging, this method will improve the forecasts from the logit model. However, the average MAE is still quite large. Such that the factors used in this report are only able to explain a small amount of the variation. The challenge is thus finding variables that closely measures a project's complexity that explains the variation.

Figure 7 shows RMSE-boxplots for the three methods. In line with previous results, random selection has the largest errors, as its box is more shifted to the right. This indicates that selecting at random has a higher variance and is less reliable compared to the other two methods. Furthermore, the combined selection method has a median RMSE at 15.96%. The relatively large RMSE shows that the logit model is not only biased, it also holds high variance. This means that the model performs especially poor with extreme cases.

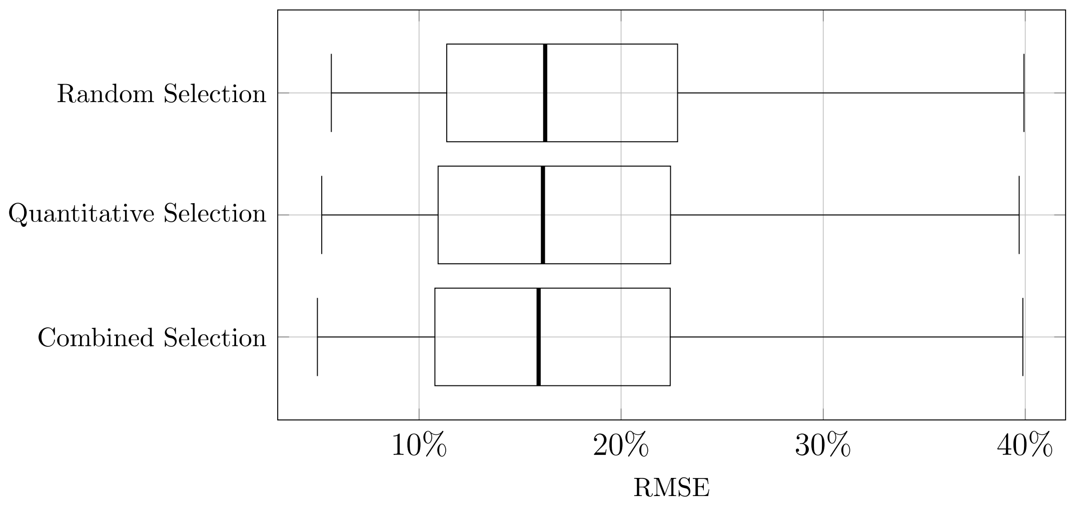

Figure 7: RMSE-boxplots calculated using the logit model for $k = 50$ and the three proposed methods.

5.2 Artificial Neural Network

Single-layer ANN models are built using different settings for C. Table 3 shows the resulting MAE and the optimal amount of neurons, which are estimated using CV.

C	0.1	0.5	1
MAE	3.340%	3.471%	3.506%
#Neurons	6	6	7

Table 3: MAE from ANN models with different hyperparameter specifications. Furthermore, the table also shows the optimal amount of neurons in the hidden layer.

The table above shows the optimal hyperparameters at $C = 0.1$ and 6 neurons with an MAE of 3.34%. Furthermore, the results show that lowering the thresholds results in lower forecasting error. However, computational time also increases significantly. The model does not converge within a satisfactory time for C lower than 0.1. However, the error size is not significantly worse for other settings. For example, the difference in average error between the worst and best specification is around 0.17%. Besides, the optimal amount of neurons appears to be around 6, but the exact amount can vary per specification.

Figure 8 shows the in and out-of-sample MAE for different number of neurons. The latter estimated by CV and C fixed at 0.1.

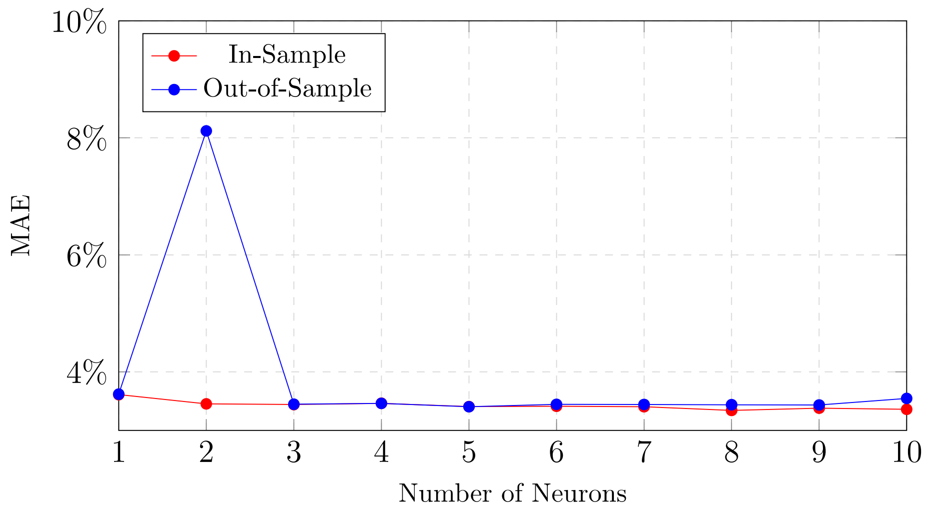

Figure 8: The MAE from ANN with different number of neurons in the hidden layer and $C = 0.1$. *Furthermore, red and blue lines denote the models in and out of sample fit respectively.*

The plot shows that MAE in-sample decreases as the amount of neurons increases. However, a low amount of neurons can cause underfitting as seen from the spike at 2 neurons for the out-of-sample fit. These findings support the notion that more neurons grow the potential of the network to learn complex dependencies. However, too many neurons increases the risks of overfitting as the plot shows an increase in out-of-sample MAE at 10 neurons. Furthermore, more neurons drastically increase computing time. The computation time to estimate the model can be extensive for a large network. For example, the time to estimate the ANN models in this paper range between 1 and 24 minutes depending on the hyperparameters. Therefor, a grid-search can be extremely slow. Although an extensive grid-search for this problem is not necessary. As there is little difference in performance between the optimal ANN and an arbitrary one. The MAE for the 3 neuron network is not significantly larger than the minimum found value. Where the in-sample minimum is at 8 neurons and out sample minimum at 6, which are both 3.34%. Although the network error is quite similar for the other networks, except for the 2 and 10 neuron network.

Finally, ANN models with 2 hidden layers are built. As a second layer could enable the network to find complex dependencies which a single layer network can not. The 2 layer networks could thus have improved forecasting performance. However, a second layer also increases the chance of overfitting and increases computation time. Figure 9 shows a 3D-surface plot of the MAE levels for the number of neurons in the first and second hidden layer.

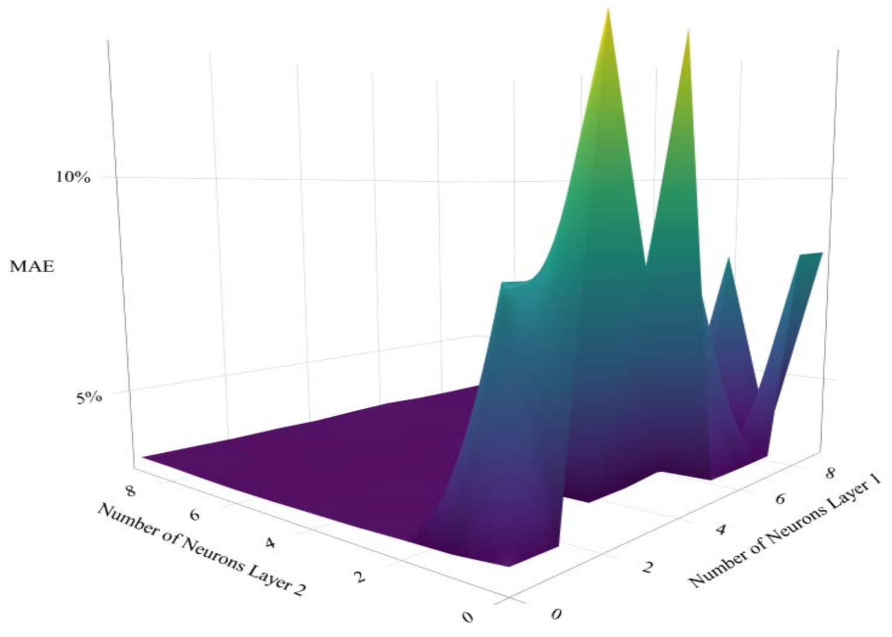

Figure 9: Surface plot of the MAE values from a 2 layer network with different number of neurons in the first and second hidden layer.

The plot shows similar behaviour for the 2 layer network as for the 1 layer networks. Where more neurons in both layers increases performance as the model is able to learn more. The networks show unstable performance with few neurons in the second layer. Since the model underfits, it is unable to learn meaningful relations resulting in poor forecasting performance. Overall, adding a second layer does not significantly improve the model. As the best specified 2 layer network has a comparable error to its 1 layer counterpart, which has an MAE around 3.34%. Given that estimating such a model through grid-search requires multiple days of computation, the additional accuracy might not be worth it.

5.3 Support Vector Machine

The hyperparameters for SVM are the regularization parameter S and the kernel parameter σ. Table 4 shows MAE for different levels of σ.

σ	1	0.01	0.001	0.0001	0.00001
MAE	4.271%	3.500%	3.464%	3.519%	3.795%

Table 4: MAE for different levels of σ estimated by CV. Furthermore, S is fixed to 10.

The results show that $\sigma = 0.001$ holds the lowest error overall at 3.46%. Compared to the hyperparameters of ANN the influence of σ is larger as there is more variance in error among the different levels. Furthermore, the overall error-size from SVM is also larger compared to ANN. Figure 10 shows MAE for varying S.

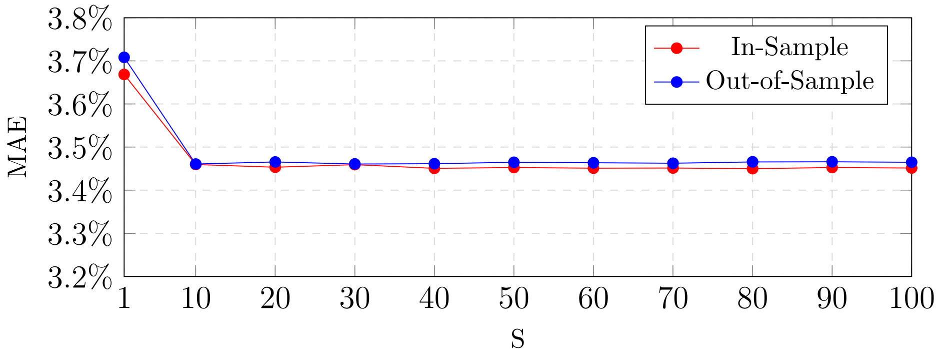

Figure 10: MAE from SVM for different levels of S with $\sigma = 0.001$. Furthermore, red and blue lines denote the models in-and out-of-sample fit respectively.

S controls the emphasis on reducing the in-sample estimation errors and thus controls over-fitting. Where large S gives more emphasis on in-sample accuracy. The plot shows that when S is small the overall error is larger since there is less emphasis on reducing estimation error. When S grows large the model converges to an error of around 3.46%. For this application $S = 10$ is enough to balance the accuracy both in and out-of-sample. Overall, the hyper-parameters of SVM have a bigger influence on the outcome. Such that finding appropriate hyperparameters is more important for comparable results than with ANN. On the other hand, SVM has the benefit of significantly shorter computation time.

5.4 Random Forest

The last ML model used to forecast cost curves is RF. Figure 11 shows MAE for different amount of RT and m. Overall the model performance shows lower error than SVM and higher than ANN. As the lowest found out-of-sample error is around 3.42%.

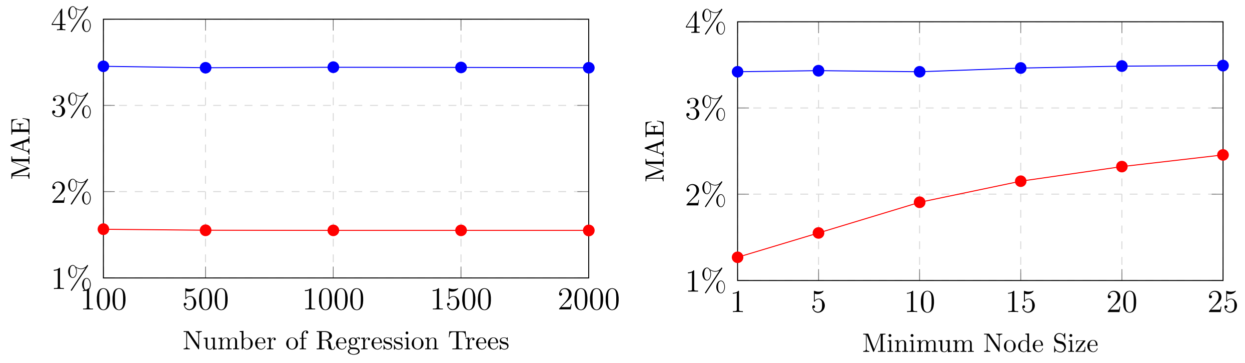

Figure 11: MAE from RF with different values for the number of RT and the minimum node size. Furthermore, red and blue lines denote the models in- and out-of-sample fit respectively.

The left plot shows that more RT improves performance as expected. Since a higher number of trees reduces the variance of the estimator and thus results in a lower error. However, the error reduction is quite small. For instance, MAE reduction is only 0.02% when growing the forest from 100 trees to 2000 for the out-of-sample estimates.

The right plot shows the effect of varying minimum node-size m on the forecast error. Setting m high prevents the model to overfit the data, however, it also limits the learning potential of the model. From figure 10 it can be concluded that the model is overfit. As in-sample error is around 1.56% in the left plot, which is significantly lower compared to the out-of-sample fit. The right plot shows the error quickly increases in-sample when m increases. Although the error also increases out-of-sample. Moreover, a large m reduces computing time and the out-of-sample performance does not become significantly worse. Therefor, m should be set around 10 to balance the model performance. Finally, table 9 in the appendix shows that the optimal number of splitting variables is around $p^* = 7$, which is in this case closely around $\lfloor P/3 \rfloor$.

5.5 Variable Importance

The importance of each explanatory variable is measured in terms of the Gini-index, where the optimal RF model is used to calculate this measure. Figure 12 shows the resulting Gini-indices. Further note that there is not a dummy variable for each categorical variable, since adding one for each class would induce perfect multicollinearity in the data.

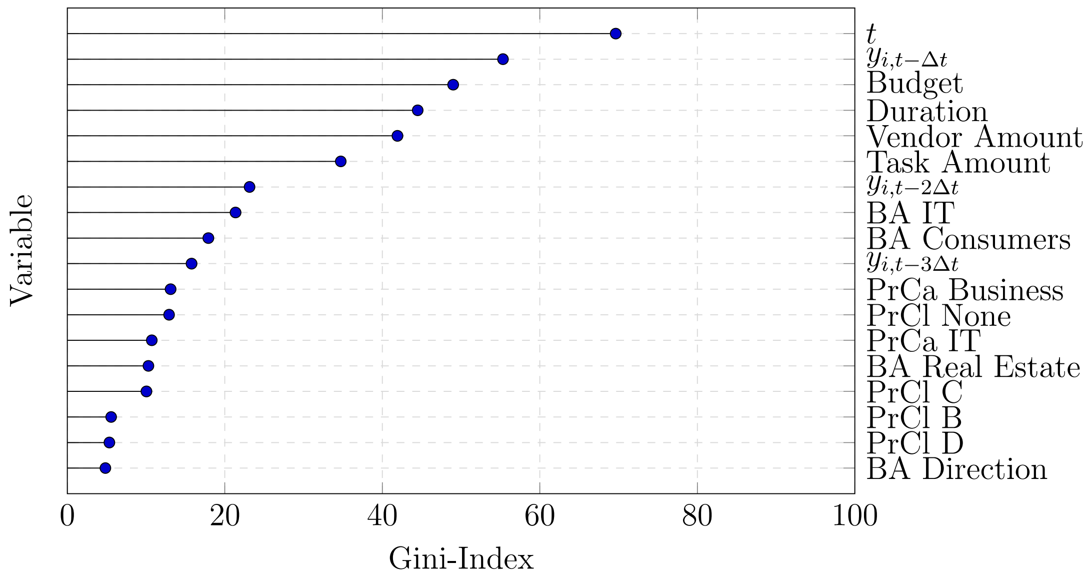

Figure 12: Gini-index for every explanatory variable, calculated using the optimal RF model.

Overall the most important variable is t, which shows that time is still the most important factor to determine cost flow. Moreover, $y_{i,t-\Delta t}$ is second most important, showing that using a projects own history improves the model significantly. From the indices of $y_{i,t-2\Delta t}$ and $y_{i,t-3\Delta t}$, the figure further shows that cost information from earlier periods has diminishing importance though they are still relatively important. Conversely, their influence is smaller than that of the variables that quantify the projects complexity: budget, duration, vendor and task amount. Budget shows the largest Gini-index of these four, although their indices are at a similar level. Furthermore, the categorical variables show to be significantly less important which could be due to poor data quality in the case of project classification. Of these variables, business area shows to be the biggest determining factor, which can be explained by the difference in spending habits of separate organizations.

5.6 Out-of-sample Evaluation

Overall the ML models display a significant improvement over the traditional logit model in the set of projects used for model estimation. Using CV, MAE is estimated at 3.34%, 3.46% and 3.42% for ANN, SVM and RF respectively. Whereas the logit model shows an average error of around 15%. The results show that a time-series approach has the potential to give more accurate forecasts on project cost flow. Since this framework dynamically adjusts forecasts as projects reach maturity. Whereas forecasts from the logit model are static. Furthermore, the logit model has limited capability of using project-variables in the model, whereas the ML models directly use these variables as input. However, thus far the focus of the research has been on short term forecasting, with the forecast window being 5% of project-duration. The logit model forecasts the entire curve from project start till end. For a fair model comparison on forecasting potential, the optimally parameterized ANN, SVM and RF model are tested on the 500 projects that have been left out in model-tuning and parameter-estimation. Table 5 shows the goodness of fit of these models.

	In-Sample			Out-of-Sample					
Model	R^2	MAE	RMSE	R^2	MAE	RMSE	Hyperparameters		
ANN	96.625%	3.381%	5.591%	96.412%	3.423%	5.742%	$M^1 = 6$	$M^2 = 0$	$C = 0.1$
SVM	96.230%	3.500%	5.908%	96.051%	3.539%	6.025%	$\sigma = 0.001$	$S = 10$	
RF	98.931%	1.859%	3.147%	96.305%	3.623%	5.827%	$B = 500$	$m = 10$	$p^* = 7$

Table 5: The goodness of fits of the optimally specified ANN, SVM and RF. Furthermore, the out-of-sample fit is calculated using the 500 test projects.

The results show that the models generalise well for new projects. Since the error-sizes are roughly the same in and out-of-sample. Only RF is highly overfit as expected from previous results. Furthermore, the out-of-sample MAE results are a bit higher than the CV estimates. However, they are quite close, showing CV is a reliable method to select models on forecasting potential. The highest and lowest MAE are from ANN and RF respectively. As the RF model is overfit, the model estimates are the most biased out of the three. However, SVM performs worst in terms of RMSE, which indicates that this model holds the highest variance.

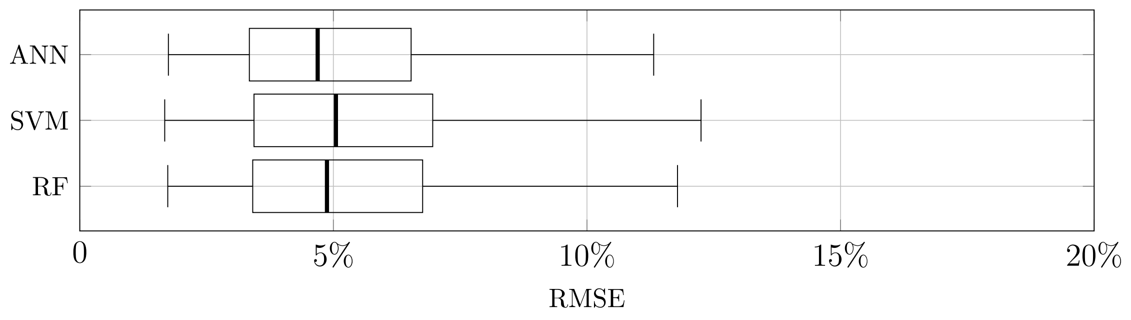

Figure 13: RMSE-boxplots calculated using the optimal ANN, SVM and RF.

Figure 13 shows the RMSE-boxplots per ML model. The median RMSE values are around 4.69% and 4.87% for ANN and RF respectively. With SVM having the highest median RMSE of 5.05%. The short term forecasting results show that out of the three models, ANN shows the lowest level of bias and variance. Furthermore, to investigate how the forecast error develops over the course of a project's duration, the average RMSE over time is plotted in figure 14.

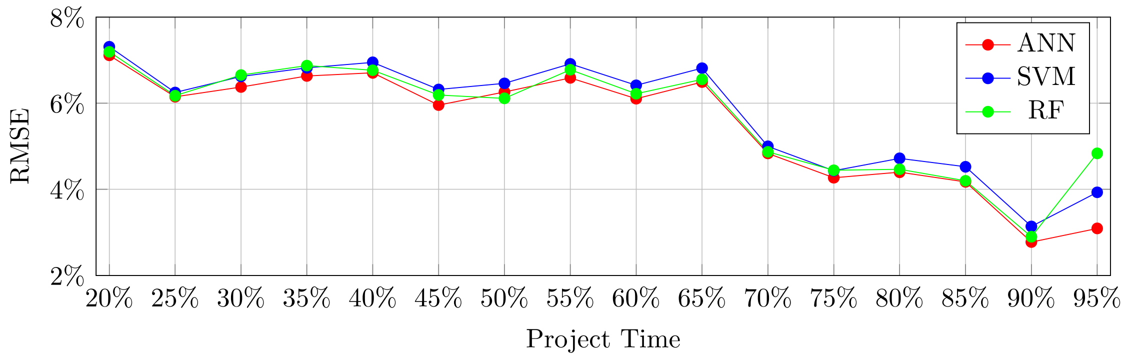

Figure 14: RMSE at the time-points during a projects lifecycle. Furthermore, the red, blue and green lines denote the optimal ANN, SVM and RF respectively.

Overall the errors of the models display similar patterns. However, relative model performance varies at each time-point during the project. Where ANN performs the best at most time-points and SVM the worst. The line of RF is mostly in between the other lines, with the exception at project end. Furthermore, the errors are higher during the first 65% of the project which decreases as the project reaches maturity. Which shows the large cost flow uncertainty during the early to middle stages of a project. Still, the average errors at each time-point are lower than the average errors from the logit model.

5.7 *h*-Step ahead Forecasts

In this section, the ML models are compared on their long term forecasting ability. The forecast is now made multiple periods ahead of time, also known as h-step ahead forecasts in time-series literature. The goal is to forecast $\hat{y}_{t+h|t}$, the cost flow at period $T + h$ given information up to t. Since the project cost-curve is divided into 19 periods and 3 consecutive periods are used as input, the h-step ahead forecasts are made for $h = 1, \ldots, 16$. The resulting MAE at each h is shown in table 6.

h-step	1	2	3	4	5	6	7	8
ANN	3.423%	5.934%	7.885%	9.417%	10.534%	11.276%	11.680%	11.797%
SVM	3.539%	6.095%	8.122%	9.763%	11.042%	11.983%	12.672%	13.121%
RF	3.623%	6.144%	8.020%	9.476%	10.479%	11.093%	11.346%	11.342%
h-Step	9	10	11	12	13	14	15	16
ANN	11.649%	11.265%	10.762%	9.968%	9.070%	8.083%	7.029%	5.633%
SVM	13.304%	13.350%	13.184%	12.808%	12.218%	11.405%	10.312%	9.082%
RF	11.014%	10.492%	9.930%	9.234%	8.390%	7.413%	6.330%	4.784%

Table 6: MAE for h-step ahead forecasts made using the optimal ANN, SVM and RF. Furthermore, MAE is calculated on the out of sample projects.

Overall SVM has the highest error for most forecasting windows, especially when h is large. The other two models show a reversing pattern in the table above. As in previous results, ANN has the lowest error for short term forecasting, i.e. when $h = 1, \ldots, 4$. However, the estimates of RF are more stable in terms of long term forecasting as the model has the lowest error for $h = 5, \ldots, 16$. Such that it could depend on the forecasting window which model to use. Besides, the forecast-error decreases at $h = 11$ and above. This result is reasonable since estimates for large h are mostly at time-points near the end of a project. As seen in figure 14, the uncertainty in cost flow decreases near the end of a project as costs converge to 100%. Lastly, the table shows lower errors at each h than the average error from the logit model, further showing the benefits of the ML framework even for long term forecasting.

5.8 Budget Forecasts

In this section, the effectiveness of the ML cost flow models in a practical application is investigated. To see if the individual estimated project cost flows can be aggregated to forecast a company's full project-portfolio. Here, the model forecasts are used to estimate the portfolio's budget a fixed period ahead of time. However, the cost flow from the ML framework can not be directly computed at all dates. Instead, the ML forecasts are used as indicators for future cost flow movement. In this application, the forecast is made at the earliest project time-point after the end of the next period. Afterwards, the cost flow at the end of next period is interpolated between this forecasted value and the current cost percentage. Furthermore, values for the starting periods of each project are imputed from project averages. For this application only long term projects of over one year are used from the out-of-sample set, the resulting portfolio consists of 303 projects with a total value of around € 460 million. Figure 15 shows the actual and estimated cumulative budget with forecasting windows of a month, a quarter and a year.

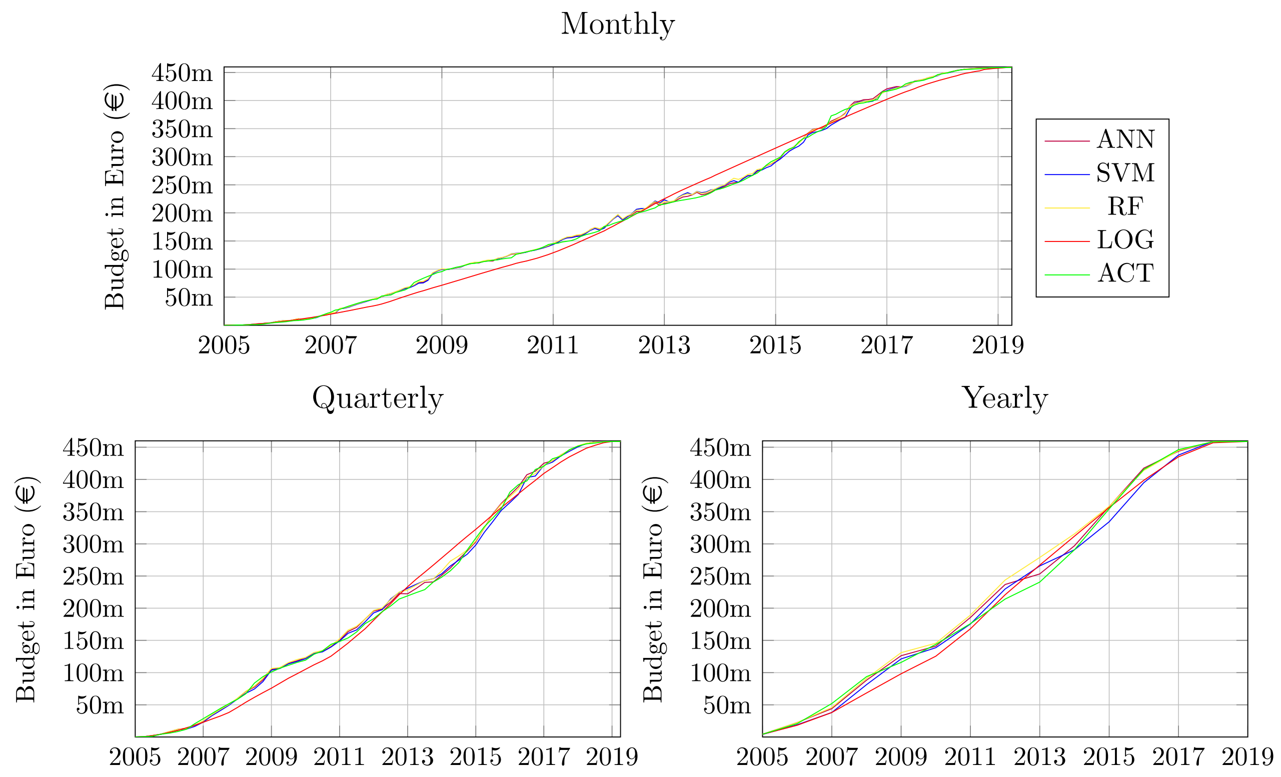

Figure 15: The budget forecasts versus the actual costs of a long term project-portfolio, with forecast windows of a month, a quarter and a year.

From the plot, it can be seen that the lines of the ML models track the actual costs more accurately. This shows the effectiveness of these models to dynamically adjust forecasts, whereas the line of the logit model is unable to capture the deviations. This result is more visible for the monthly and quarterly forecasts, as the ML models are more accurate for short term forecasting. Therefor, the yearly estimates are less stable, which can be seen in figure 15 as the lines are more diffuse in the yearly plot. Moreover, table 7 shows the MAE between the estimated and actual budget per period.

	Month	Quarter	Year
ANN	€ 2,512,261	€ 3,513,332	€ 5,877,436
SVM	€ 2,907,849	€ 4,521,167	€ 8,699,726
RF	€ 2,871,679	€ 4,480,553	€ 8,129,818
LOG	€ 11,682,371	€ 11,522,018	€ 11,402,776

Table 7: MAE per period-size between the estimated and the actual budget of the long term project-portfolio.

The table further shows that the ML models perform better on short term forecasting since MAE increases for larger period sizes. Only the logit model shows a reverse pattern of decreasing error. Although the error of this model is the largest of all the models and is above € 11 million every period size. The results for the ML models are in line with results in previous sections. ANN holds the lowest error, which is expected since the model has shown the lowest MAE and RMSE in previous sections. Furthermore, previous results show that RF estimates are biased due to overfitting and SVM having unstable estimates due to high variance. However, in this application, RF slightly outperforms SVM as it deviates less from the actual costs.

6 Conclusion

In the research of this paper, several quantitative models are used to forecast project cost flow using data from past projects. First, the benchmark logit model is discussed and three methods to average data of past projects are introduced. Afterwards, the ML models: ANN, SVM and RF are optimized based on the out-of-sample estimates of CV. The Gini-Index of the RF model is used to find the main cost drivers in the data. Finally, the optimal ML models are tested using a set of 500 test projects. They are compared on several criteria for forecasting project cost flow. That is on their ability to model the curve at different stages during the projects lifecycle, on the long term forecasting ability and their applicability to estimate the budget of a large project-portfolio.

The logit model shows that selecting similar projects based on project variables improves forecast accuracy. As the third method achieves the global minimum average MAE at 14.90%. However, this error is significantly higher than the average errors from the ML models. Model-tuning shows several strengths and weaknesses of these models. ANN holds the lowest MAE of all of the models at 3.43% and shows the strongest performance based on all criteria. However, the model is unstable when the network has not enough neurons. Moreover, when the network has too many neurons it starts to overfit and computation time significantly increases. Overall, SVM displays the weakest performance out of the ML models based on all criteria. As SVM has an MAE of 3.54% out-of-sample. Furthermore, the model has a relatively high RMSE, which indicates that the estimator has the highest variance. Lastly, the RF model has the lowest in-sample fit overall at 1.86%. Although the resulting model is overfit, as its out-of-sample MAE is the largest out of the ML models at 3.62%. Even so, the model shows more promising results in the h-step forecasting results, where RF achieves the lowest error for long term forecasting. In the budget forecasting application, ANN again holds the lowest average error out of all the models. The results also show the ability of the ML models to track the actual cost curve more closely compared to the logit model. Such that the ML models significantly outperform the latter in forecasting a project-portfolios budget.

The Gini-index of the RF model gives insight into the importance among the explanatory variables. The results show that time is still the main cost driver, however, it also shows that the model cost flow forecasts can be significantly improved by using a project's own curve history and additional project-variables. The cost information at a previous period is the second most important variable and the quantitative variables such as budget and duration follow close behind. Out of the categorical variables, Business Area has the largest influence, which shows the difference in spending habits between separate organizations. Overall, including more project-related variables could explain more of the curve variation. Improved data quality and well maintained initial budget estimates will improve the model further and enables it to find undershoot or overshoot of the budgeted cost. The ML framework has the same drawbacks as the traditional logit model since the total realized cost is used to scale the costs. The drawback is that this methodology is only applicable for thoroughly reviewed projects, such that project controllers have made an unbiased estimate of total budget and duration, ensuring these are semi robust estimates. Given that these estimates have no bias and similar variance the estimates will hold reasonably well on an aggregated level as in the budget forecasts. Even so, with the same drawbacks as the logit model, the ML framework shows significant improvement.

Further improvements to the ML models can be achieved by combining other AI algorithms into hybrid models. For example, genetic algorithms for optimal parameters can improve the SVM results or fuzzy logic can be used to handle the large uncertainty during the early stages of a project. In addition, different regularization methods could be used to prevent RF or ANN from overfitting. Another model improvement would be to model outlier behaviour of certain projects. As the data includes projects with long periods without costs, resulting in flat lines and sharp corners. These type of events could be modelled separately using e.g. jump processes.

References

Balkau, B. (1975), A financial model for public works programmes, *in* ‘National ASOR Conference, Sydney’, pp. 25–27.

Boussabaine, A. H. and Elhag, T. (1999), Applying fuzzy techniques to cash flow analysis, *Construction management & economics* 17(6), 745–755.

Boussabaine, A. and Kaka, A. (1998), A neural networks approach for cost flow forecasting, *Construction Management & Economics* 16(4), 471–479.

Breiman, L. (2001), Random forests, *Machine learning* 45(1), 5–32.

Caruana, R. and Niculescu-Mizil, A. (2006), An empirical comparison of supervised learning algorithms, *in* ‘Proceedings of the 23rd international conference on Machine learning’, ACM, pp. 161–168.

Cheng, M.-Y., Hoang, N.-D. and Wu, Y.-W. (2015), Cash flow prediction for construction project using a novel adaptive time-dependent least squares support vector machine inference model, *Journal of Civil Engineering and Management* 21(6), 679–688.

Cheng, M.-Y. and Roy, A. F. (2011), Evolutionary fuzzy decision model for cash flow prediction using time-dependent support vector machines, *International journal of project management* 29(1), 56–65.

Cheng, M.-Y. and Wu, Y.-W. (2009), Evolutionary support vector machine inference system for construction management, *Automation in Construction* 18(5), 597–604.

Cortes, C. and Vapnik, V. (1995), Support-vector networks, *Machine learning* 20(3), 273–297.

Dormann, C. F., Calabrese, J. M., Guillera-Arroita, G., Matechou, E., Bahn, V., Bartoń, K., Beale, C. M., Ciuti, S., Elith, J., Gerstner, K. et al. (2018), Model averaging in ecology: A review of bayesian, information-theoretic, and tactical approaches for predictive inference, *Ecological Monographs* 88(4), 485–504.

Drake, B. (1978), A mathematical model for expenditure forecasting post contract, *Technician Israel Institute of Technology, Haifa* .

Fletcher, L., Katkovnik, V., Steffens, F. and Engelbrecht, A. (1998), Optimizing the number of hidden nodes of a feedforward artificial neural network, *in* '1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36227)', Vol. 2, IEEE, pp. 1608–1612.

Friedman, J., Hastie, T. and Tibshirani, R. (2001), *The elements of statistical learning*, Springer series in statistics New York.

Gokhale, S. S. and Lyu, M. R. (1997), Regression tree modeling for the prediction of software quality, *in* 'proceedings of the Third ISSAT International Conference on Reliability and Quality in Design', Citeseer, pp. 31–36.

Gunn, S. R. et al. (1998), Support vector machines for classification and regression, *ISIS technical report* 14(1), 5–16.

Hwee, N. G. and Tiong, R. L. (2002), Model on cash flow forecasting and risk analysis for contracting firms, *International Journal of Project Management* 20(5), 351–363.

Kaka, A. P. (1999), The development of a benchmark model that uses historical data for monitoring the progress of current construction projects, *Engineering, Construction and Architectural Management* 6(3), 256–266.

Kaka, A. P. and Price, A. D. (1991), Net cashflow models: Are they reliable?, *Construction Management and Economics* 9(3), 291–308.

Kenley, R. and Wilson, O. D. (1986), A construction project cash flow model—an idiographic approach, *Construction Management and Economics* 4(3), 213–232.

Kumar, M. and Thenmozhi, M. (2006), Forecasting stock index movement: A comparison of support vector machines and random forest, *in* 'Indian institute of capital markets 9th capital markets conference paper'.

Lowe, J. (1987), Cash flow and the construction client–a theoretical approach, *Managing Construction Worldwide, E & FN Spon, London* 1, 327–336.

Miskawi, Z. (1989), An s-curve equation for project control, *Construction Management and Economics* 7(2), 115–124.

Nazem, S. (1968), Planning contractors capital, *Building Technology and Management* 6(10), 256–60.

Odeyinka, H. A., Lowe, J. and Kaka, A. (2008), An evaluation of risk factors impacting construction cash flow forecast, *Journal of Financial Management of Property and Construction* 13(1), 5–17.

Odeyinka, H., Lowe, J. and Kaka, A. (2012), Regression modelling of risk impacts on construction cost flow forecast, *Journal of Financial Management of Property and Construction* 17(3), 203–221.

Riedmiller, M. and Braun, H. (1993), A direct adaptive method for faster backpropagation learning: The rprop algorithm, *in* 'Proceedings of the IEEE international conference on neural networks', Vol. 1993, San Francisco, pp. 586–591.

Salas-Molina, F., Martin, F. J., Rodriguez-Aguilar, J. A., Serra, J. and Arcos, J. L. (2017), Empowering cash managers to achieve cost savings by improving predictive accuracy, *International Journal of Forecasting* 33(2), 403–415.

Segal, M. R. (2004), Machine learning benchmarks and random forest regression.

Tucker, S. and Rahilly, M. (1982), A single project cash flow model for a microcomputer, *Building Economist* 21, 109–115.

Yang, I.-T. (2005), Impact of budget uncertainty on project time-cost tradeoff, *IEEE Transactions on engineering management* 52(2), 167–174.

Abbreviations

AI:	Artificial Intelligence
ANN:	Artificial Neural Networks
ARX:	Auto-Regressive Exogenous
BP:	Back-Propagation
CV:	Cross-Validation
fmGA:	fast messy Genetic Algorithm
KNN:	K-nearest-neighbors
MAE:	Mean Average Error
ML:	Machine Learning
OLS:	Ordinary Least Squares
RBF:	Radial Basis Function
RF:	Random Forest
RMSE:	Root Mean Square Error
RPROP:	Resilient-Propagation
RT:	Regression Tree
SVM:	Support Vector Machines

Table 8: List of abbreviations used in this paper.

Additional Tables

p^*	1	2	3	4	5	6	7	8	9	10
MAE	13.362%	5.942%	3.986%	3.576%	3.473%	3.438%	3.421%	3.421%	3.418%	3.418%
RMSE	15.791%	7.616%	5.998%	5.746%	5.688%	5.677%	5.676%	5.684%	5.690%	5.694%

Table 9: Estimates of the predictive power from the RF model for different values of p^* estimated by 5-fold CV. Furthermore, amount of trees was set to 100 and $m = 5$.

Algorithms

Algorithm 1 Estimating Artificial Neural Network using Resilient-Propagation

```
  Randomly initialise model parameters
  while $E(\theta) > C$ do
    Forward Phase
    For the current parameters calculate derived $Z$ and input variables $v^l_{m,i}$
    Backward Phase
    Calculate model errors using the Propagation equation:
    $\delta^l_{m,i} = s'(v^l_{m,i}) \sum_{k=1}^{M^{l+1}} \delta^{l+1}_{k,i} \alpha^{l+1}_{k,m}$

    Using the model errors calculate the partial derivatives:
    $\frac{\partial E_i}{\partial \alpha^L_{m,j}} = -2\big(y_i - f(x_i)\big)\beta_m s'\big(v^L_{m,i}\big) z^{L-1}_{j,i}$   and   $\frac{\partial E_i}{\partial \alpha^l_{m,j}} = s'(v^l_{m,i}) z^{l-1}_{j,i} \sum_{k=1}^{M^{l+1}} \delta^{l+1}_{k,i} \alpha^{l+1}_{k,m}$

    Calculate change in parameter-weights along the RPROP rules:
    $\Delta w^{(r)} = \begin{cases} \eta^+ * \Delta w^{(r-1)}, & \text{if } \frac{\partial E}{\partial w}^{(r-1)} * \frac{\partial E}{\partial w}^{(r)} > 0 \\ \eta^- * \Delta w^{(r-1)}, & \text{if } \frac{\partial E}{\partial w}^{(r-1)} * \frac{\partial E}{\partial w}^{(r)} < 0 \\ \Delta w^{(r-1)}, & \text{else} \end{cases}$
    Update parameters
  end while
```

Algorithm 2 Estimating Random Forest

```
  for $b =$ 1 to $B$ do
    Draw random bootstrap sample from training data
    Grow tree $T_b$ by repeating the following steps for each terminal node:
    while Size terminal node $> m$ do
      Select $p$ variables at random
      Search for the optimal splitting condition $s$ by maximizing the decrease in impurity
      $\Delta^s_t I^s_t$ over all possible splits among the $m$ variables
    end while
  end for
```

Derivations

Partial derivatives of ANN-parameters

The updating steps from ANN follow from the partial derivatives with respect to the parameters. The derivatives with respect to α_m^l are less straightforward. Let $v_{m,i}^l = {\alpha_m^l}^T Z_i^{l-1}$, which is the input for node m before it enters the logistic function. Then, the derivative with respect to $\alpha_{m,j}^l$ can be decomposed as

$$\frac{\partial E_i}{\partial \alpha_{m,j}^l} = \frac{\partial E_i}{\partial v_{m,i}^l} \frac{\partial v_{m,i}^l}{\partial \alpha_{m,j}^l} \tag{33}$$

The first term is commonly known as the error from the current model, it is defined as $\delta_{m,i}^l = \frac{\partial E_i}{\partial v_{m,i}^l}$. Furthermore, note that the second term is equal to $z_{j,i}^{l-1}$ the output stemming from node j in the previous layer. Therefor the derivative can be rewritten to:

$$\frac{\partial E_i}{\partial \alpha_{m,j}^l} = \delta_{m,i}^l z_{j,i}^{l-1} \tag{34}$$

The value $\delta_{m,i}^l$ depends on the next values of the next layers in the network such that an obvious solution is not available. However, a solution for the final hidden layer nodes can be created by observing that equation (35) holds by substituting the formula for $f_{\text{ANN}}(X)$ into equation (12).

$$E_i = \big(y_i - \sum_{m=1}^{M^L} \beta_m Z_{m,i}^L\big)^2 \tag{35}$$

Furthermore, $Z_{m,i}^L = s\big(v_{m,i}^L\big)$. By substituting this formula into equation (35) and solving $\delta_{m,i}^L = \frac{\partial E_i}{\partial v_{m,i}^L}$, the partial derivative of the final layer nodes is derived as

$$\frac{\partial E_i}{\partial \alpha_{m,j}^L} = \delta_{m,i}^L z_{j,i}^{L-1} = -2\big(y_i - f(x_i)\big)\beta_m s'\big(v_{m,i}^L\big) z_{j,i}^{L-1} \tag{36}$$

To solve the partial derivative for the nodes in hidden layers $1 \leq l < L$, $\delta^l_{m,i}$ is decomposed using the chain rule:

$$\delta^l_{m,i} = \frac{\partial E_i}{\partial v^l_{m,i}} = \sum_{k=1}^{M^{l+1}} \frac{\partial E_i}{\partial v^{l+1}_{k,i}} \frac{\partial v^{l+1}_{k,i}}{\partial v^l_{m,i}} \tag{37}$$

The second term is solved by noting the following relationship between $v^{l+1}_{k,i}$ and $v^l_{m,i}$ in equation (38), which is the relationship of the input for node k in layer $l+1$ to the inputs of all nodes in layer l.

$$v^{l+1}_{k,i} = \sum_{m=1}^{M^l} \alpha^{l+1}_{k,m} s(v^l_{m,i}) \tag{38}$$

From this relationship, the second term of equation (37) is solved as

$$\frac{\partial v^{l+1}_{k,i}}{\partial v^l_{m,i}} = \alpha^{l+1}_{k,m} s'(v^l_{m,i}) \tag{39}$$

Finally by noting that the first term of the sum in equation (37) is equal to $\delta^{l+1}_{l,i}$, equation (37) is solved as follows

$$\delta^l_{m,i} = s'(v^l_{m,i}) \sum_{k=1}^{M^{l+1}} \delta^{l+1}_{k,i} \alpha^{l+1}_{k,m} \tag{40}$$

The equation above is known as the propagation equation. From this the closed form solution of equation (34) (partial derivative) for hidden layer nodes is derived as:

$$\frac{\partial E_i}{\partial \alpha^l_{m,j}} = s'(v^l_{m,i}) z^{l-1}_{j,i} \sum_{k=1}^{M^{l+1}} \delta^{l+1}_{k,i} \alpha^{l+1}_{k,m} \tag{41}$$