Politecnico di Torino

Master of Engineering and Management
Academic Year 2024/2025
Graduation Session April 2025

Thesis topic:

An Application of Deep Learning Models for Project Cost and Duration Forecasting

Supervisor:
Narbaev Timur, PhD, A.M.ASCE, Assistant Professor of Project Management,
Dept. of Management and Production Engineering, Politecnico di Torino

Co-supervisor:
Hazir Öncü, PhD, Full Professor of Supply Chain Management Academic Area,
Rennes School of Business

Candidate:
Temirbekova Zhazira, Master graduate candidate in Engineering and Management

Table of Contents

Abstract

This research explores the application of deep learning techniques to improve prediction accuracy of project cost and time at completion in response to limitations of traditional methods due to growing complexity of modern projects. An accurate forecast is key for effective project management, reducing the risk of budget overruns and schedule delays.

The proposed multilayer perceptron (MLP)-based deep learning model is compared against an established benchmark in a form of a proven high-performing machine learning model (XGBoost) and Earned Value Management (EVM). This study uses two datasets: Dynamic Scheduling Library, containing 181 projects, and 8 additional projects from Project Portfolio Dataset for more diverse context. Mean Absolute Percentage Error serve as key metric along with Normalized Root Mean Squared Error, Standard Deviation, and Mean Lags, to evaluate forecasting accuracy across different project stages: early, mid, and late.

The central hypothesis is that well-trained and tuned deep learning architectures can produce more accurate predictions of project cost and time at completion than conventional machine learning approaches and EVM, especially during the early project phases when uncertainty is the highest.

Experimental results showed that while the optimized benchmark model performed well overall, it struggled to make accurate predictions in the early stages of a project. On the other hand, the MLP model consistently delivered comparable or even better forecasts, particularly in the early and mid stages. However, in the later stages, its performance fell slightly behind the benchmark models.

The aim of the research is to contribute to the expanding body of knowledge on artificial intelligence applications in project management. The potential success of the proposed MLP models could offer improved predictive ability, enabling project managers to make better decisions.

Chapter 1. Introduction

1.1 Background

In recent years, project management has become increasingly complex, particularly in terms of cost and schedule forecasting. Organizations across construction, infrastructure, information technology, and other sectors frequently struggle with escalating project scales, tighter deadlines, and rapidly shifting resource demands. As these pressures accelerate, inaccurate forecasting can severely impair an organization's ability to achieve its objectives, leading to budget overruns, delayed milestones, and strained stakeholder relationships. [1]

Project managers have consistently relied on traditional forecasting methods in the form of Earned Value Management (EVM). Such approaches tend to assume that future trends will reflect historical patterns, which may not be accurate in today's dynamic project environments. [2] In addition, linear or near-linear models might not be able to account for the non-linear interdependencies and complex risks that affect cost and schedule variability. These limitations can lead to overly conservative or dangerously optimistic forecasts. This is especially true in industries with fast-paced technological change, fluctuating resource availability, or evolving regulatory frameworks. [3]

Recent advances in digitalization and data collection open new opportunities to solve challenges associated with forecasting. Modern projects generate extensive data [4], which has motivated both practitioners and researchers to explore machine learning (ML) and, increasingly, deep learning (DL) methods to identify correlations that traditional techniques tend to miss. Neural network architectures have the capacity to model non-linear relationships and learn from unstructured or partially incomplete datasets, providing better and more adaptive predictions. [5]

DL adoption for project cost and duration forecasting is still in its early stages. Researchers still need to solve issues with data quality, interpretability, and the demand for significant computing resources. [6] In spite of these obstacles, as industry practices evolve toward more data-driven project management, there is an increasing awareness that advanced analytics, especially when combined with solid domain knowledge, can provide a competitive advantage. Organizations with reliable forecasting capabilities are better equipped to effectively allocate resources, anticipate risks, and make decisions that keep projects within budget and on schedule. [5]

This study explores how DL models can improve project forecasting accuracy, precision, and stability. The research aims to bridge the gap between traditional methods and emerging analytical paradigms by experimenting with neural network architectures and established performance metrics. This approach can pave the way for more resilient, data-centric project management practices.

1.2 Problem Statement

Traditional forecasting methods, including EVM, continue to be widely used in project management. However, the evolving complexity of modern projects has posed challenges for these techniques in dynamically adapting to modern project ecosystems. Some of these techniques often assume fixed starting points and a straightforward timeline, which could be misleading since the evolving requirements and new risks can appear during the project. Consequently, project teams could potentially mitigate cost overruns and time delays by employing more reliable early warning signals that take into consideration dynamically changing nature of a project. [3]

Emerging studies suggest that, unlike traditional methods, ML can enhance forecasting accuracy by identifying non-linear correlations that are often overlooked. [6] Within the ML domain itself, there is a gap between simpler methods and advanced DL architectures. If used effectively, these architectures may reveal subtle, higher-order patterns that basic ML models miss. [7]

Despite the impressive capabilities of DL, its application to forecasting project cost and duration is currently limited, with few well-known standards and guidelines. Researchers have yet to establish guidelines on neural network architectures that are best suited for varying types of project data. [7]

Thus, this study shows the need to take advantage of the potential of DL given the increasing complexity and data-richness of modern projects. This study clarifies when and how DL provides a real advantage by examining neural network performance across various project datasets and comparing it to established ML baselines. The goal is to guide project managers, data scientists, and academic researchers toward more accurate and adaptive forecasting solutions that leverage the rich data available in modern project environments.

1.3 Research Aim and Objectives

The aim of this research is to determine the extent to which DL models can further improve the accuracy and reliability of project cost and time at completion forecasting beyond what established ML approaches can offer. By experimenting with different neural network architectures and comparing their performance to recognized methods (EVM and ML), the study seeks to provide insights into which predictive methods are most effective.

To reach this goal, the research is focused on the following objectives and key considerations:

- **Review current forecasting approaches.** Examine the strengths and limitations of traditional EVM methods and modern machine learning techniques.
- **Design and implement Deep Learning Models.** Develop and configure suitable neural network architectures for cost and time at completion forecasts.
- **Benchmark against baseline.** Compare the predictive accuracy of these DL models with the traditional EVM method and widely used ML algorithms, focusing on XGBoost as a robust reference point.

By addressing these objectives, the research aims not only to expand the academic understanding of applying DL to project forecasting but also to offer practical strategies for organizations seeking to enhance their predictive capabilities in cost and schedule management.

1.4 Scope and Boundaries

This research focuses on developing and evaluating DL models for accurately predicting the total project expenses and completion times. It places particular emphasis on comparing their performance to traditional approaches. To ensure the study remains both achievable and coherent, several delimitations and constraints have been established to focus the research on the most relevant and manageable aspects of the problem:

- **Data Sources.** The data used in the analysis primarily comes from two of EVM-focused datasets: the Dynamic Scheduling Library (DSLIB) [8] and a smaller Project Portfolio (Australian) Dataset [9]. These datasets were selected since they all have extensive time-phased cost and schedule information which fits perfectly with the EVM metrics. It is important to note that the study does not include private company records or specialized industry reports, which limits its scope.
- **Predictive Techniques.** We evaluate multiple different DL architectures, however, the focus of our study is on the Multilayer-perceptron (MLP), which has been preferred based on the nature of the data as well as optimal complexity trade-offs. Other architectures, such as Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN), are mentioned but not tested. The dataset's short length and heterogeneity limit the effectiveness of sequential models, which typically require longer and more homogeneous data sequences to perform well. This choice balances the complexity of the model with the dataset's characteristics and structure.

1.5 Significance of the Study

This study investigates the application of DL for project cost and duration forecasting, offering both academic and practical contributions:

- **Academic Contribution.** From a scholarly perspective, the research contributes to AI literature in project management by benchmarking performance of neural networks against other predictive models such as XGBoost. Though extensive research has been done covering DL in image processing and speech recognition, fewer studies have rave truly evaluated the advantage of DL in forecasting project metrics. [10]
- **Practical Implications.** An efficient DL model that provides better performance compared to traditional methods can enhance resource allocation, risk mitigation, and project success. Even when DL does not outperform simpler models, understanding its limitations allows project managers to adapt forecasting strategies to their operational constraints. [11] By describing model-building steps such as data preprocessing, feature engineering, and hyperparameter tuning, the research provides a framework that practitioners can adapt to their own project environments.

Collectively, these contributions highlight the strategic value of modern AI methods in enhancing decision-making and efficiency in project management. Traditional approaches struggle to capture intricate or rapidly changing conditions, particularly in the early phases of a project. On the other hand, well-tuned DL models give invaluable foresight that helps avert expensive disruptions and lead to a new era of proactive, data-informed decision-making.

1.6 Thesis Structure

The remainder of this thesis is organized into the following chapters:

- Chapter 2: Literature Review. Presents an exploration of existing project forecasting methods. It begins by discussing traditional techniques (EVM) and then examines more advanced ML approaches. The chapter concludes with an overview of DL architectures.
- Chapter 3: Methodology. Details the research design and used tools (software), data selection process, feature engineering, and preprocessing steps. The chapter also describes the model setup, training procedures, and evaluation metrics.
- Chapter 4: Results. Reports the empirical results of the study, including performance metrics for both the benchmark models and the DL architectures. The results are further analyzed by project stage (early, mid, late) to highlight where each model type excels or struggles.
- Chapter 5: Discussion. Interprets the results in the context of the research objectives, comparing DL outcomes to ML baseline, and explores reasons for observed performance differences.
- Chapter 6: Limitations and Future Research Directions.

Chapter 2. Literature review

2.1 Earned Value Management (EVM)

Project forecasting forms the backbone of successful project management. It informs decisions about budgeting, the allocation of resources and schedule changes, and influences a project's likelihood of meeting cost and timeline targets. EVM and regression/time-series modeling have been the main traditional methods for project forecasting. EVM integrates cost, schedule, and scope to provide a comprehensive view of project performance, while regression/time-series modeling uses statistical techniques to predict future project outcomes based on historical data. [12, 13, 14] While these methods have shown their worth, the rapidly evolving complexity of modern projects exposes certain limitations that call for more advanced approaches.

EVM has evolved into a global best practice applied across various industries and nations; consequently, its basic principles and use in practice have been comprehensively described in many sources. The Project Management Institute issued a guidebook on EVM. [2] To summarize, EVM combines cost, schedule, and scope into a single framework that enables project managers to monitor performance in a more systematic manner than simple budget-vs-actual comparisons. Through key metrics, EVM enables project teams to derive performance indices (Table 1) that help identify early variances and trends. [1]

Table 1. Key EVM metrics

Key EVM metrics	
Planned Value (PV)	The planned value is often called the budgeted cost of work scheduled (BCWS). It is the time-phased budget baseline and an immediate result of the schedule constructed from the project network.
Earned Value (EV)	It is often called the budgeted cost of work performed (BCWP). It represents the amount budgeted for performing the accomplished work by a given point in time. EV is the cumulative measure of the work performed.
Actual Cost (AC)	The actual cost is often referred to as the actual cost of work performed (ACWP). It represents the cumulative actual cost spent at a given point in time.
Budget at Completion (BAC)	The total agreed budget of the project.
Planned Duration (PD)	The total agreed project duration.
Performance measures	
Cost Performance Index (CPI)	Assesses how efficiently the project's BAC is utilized. $$CPI = EV/AC \quad (1)$$ A CPI of 1.0 indicates that cost performance is on target; more than 1.0 indicates the project is running under budget, and less than 1.0 indicates cost overruns.
Cost Variance (CV)	Difference between EV and AC $$CV = EV - AC \quad (2)$$

Schedule Performance Index (SPI)	Shows how effectively time is spent. $SPI = EV/PV$ (3) A value greater than 1.0 suggests that the project is ahead of schedule, while a value below 1.0 reveals delays.
Schedule Variance (SV)	Difference between EV and PV $SV = EV - PV$ (4)

The main advantage of EVM is the early warning capability; if CPI or SPI deviates far from 1.0, it serves as a signal for corrective actions, providing objective information instead of subjective estimates. [15] Since EVM examines work accomplished in the context of cost and time, it offers a more complete picture than methods that focus solely on budget or schedule alone. [16]

Although EVM has been set up to follow up both time and cost, the majority of the research has been focused on the cost aspect. [1] This resonates with the idea that EVM was originally developed for cost management, with less emphasis on forecasting a project's duration. [15]

The Earned Schedule (ES) method extends the projection of EV onto the PV curve. It addresses limitations of traditional EVM schedule indicators, as SV and SPI provide unreliable time forecasts near project completion, offering a time-based alternative. [17, 18] ES is found by comparing the cumulative EV earned to the performance baseline. The time associated with EV is derived from the PV S-curve. The ES metric translates the monetary value of EV into a time-based value. The cumulative value for the ES is found by using the EV to identify in which time increment t of PV the cost value for EV occurs. [19, 20]

$$ES = c + \frac{EV - PV_c}{PV_{c+1} - PV_c} \tag{5}$$

where,
c – time instance such that $PV_c \leq EV < PV_{c+1}$

Performance indicators can then be calculated using ES:

$$SV(t) = ES - AT \tag{6}$$
$$SPI(t) = ES/AT \tag{7}$$

where,
AT – actual time

The SPI(t) is similar to the traditional SPI formula, but this new performance measure is reliable over the complete horizon of the project.

EVM has been widely adopted, however, it assumes a stable baseline and consistent reporting of actual data. [21] If a project is subject to scope creep, changing requirements, or sudden disruptions, EVM metrics may become disconnected from reality. Traditional EVM calculations assume linear progress, making them unreliable when project tasks overlap or risks skew the conventional 'planned vs actual' approach to risk management. [2] Such limitations emphasize the necessity of more flexible forecasting models considering that current digital project environments generate more complex and frequent data changes. [22]

2.2 Machine Learning in Project Forecasting

ML is an expanding domain within AI that has garnered significant attention in project management research and practice due to its ability to enhance forecasting accuracy and adapt to dynamic project conditions. Unlike traditional statistical approaches that rely on static assumptions and linear models, ML learns from historical data, recognizes patterns, and adapts to changing project conditions. This flexibility makes ML particularly effective for project forecasting, as it can adapt to dynamic environments, process vast datasets, and handle unexpected disruptions. [6]

ML encompasses a broad set of algorithms, including classical regression for trend analysis, decision trees for decision-making processes, neural networks for complex pattern recognition, and clustering approaches for grouping similar project data, each offering unique advantages for analyzing and predicting project outcomes. [23] The unifying principle across these ML models is that they learn by example; they detect correlations and patterns within training datasets and then use that knowledge to forecast outcomes such as project cost or schedule duration. [5] As projects generate more data, ML's data-driven nature processes this information efficiently. By doing so, ML can continuously improve its forecasts through iterative training, thus offering greater resilience to all the uncertainties and complexities present in many project environments. [6]

A wide spectrum of ML algorithms addresses the varying needs of project forecasting. For example:

- Random Forests and Gradient Boosting employ ensembles of decision trees, combining multiple weak learners to improve predictive accuracy and manage messy or partially missing datasets. [5]
- Artificial Neural Networks (ANNs) are inspired by the human brain and consist of layers of connected nodes (neurons) that recognize patterns and make predictions for complex data. [25]

ML models can combine EVM metrics like CPI and SPI with other project data to improve estimates. [22] By addressing the assumption of linear cost growth in EVM, ML-based 'growth modeling' improves forecasts by capturing the complexities of actual project dynamics. It clarifies how real-world performance indicators are expected to change during project execution. This approach highlights a common trend in technology adoption: rather than replacing existing methodologies completely, ML enriches EVM by offering more accurate and data-driven insights. [6]

In line with its role in improving estimates, ML applications can span various phases of the project lifecycle, including planning, execution, and monitoring. In the early stages, it enhances the accuracy of initial cost estimates, reducing the risk of underbidding or overestimating the project's requirements. As a project progresses, ML-based forecasting tools can integrate real-time data such as updated costs and schedule metrics, to refine predictions and assess potential risk factors. [24] Researchers have reported that ML applications can reduce budget overruns in construction and optimize sprint planning in software projects by providing precise cost analysis and accurately predicting resource needs. [5]

By analyzing historical data and identifying patterns, ML can bring an evidence-driven approach to predicting delays or cost increases, enabling proactive measures to be taken. This analysis allows project managers to take proactive measures, such as reallocating resources or adjusting timelines, before minor delays escalate into significant budget overruns.

Despite the significant benefits ML offers for project forecasting, it is important to consider several challenges that may limit its effectiveness and adoption.

- Many organizations do not have comprehensive project databases. This limitation reduces ML's effectiveness, as models trained on insufficient data do not generalize well. [1]
- The black box nature of certain algorithms, particularly neural networks, presents additional complications, as project managers may have difficulty knowing how or why a model reached a certain cost predictor. Interpretation difficulties may undermine the practical adoption of ML solutions. [4]
- Additionally, some ML techniques require significant computational resources and careful hyperparameter tuning, which demand specialized expertise. [10] Hyperparameter tuning adjusts parameters to prevent underfitting, where the model fails to capture patterns, and overfitting, where it is excessively tailored to the training data. This aspect will be examined thoroughly in the Methodology chapter, establishing our benchmark ML model.

Despite these constraints, ML is a promising tool for predicting cost and schedule more accurately than traditional EVM methods, allowing for the development of specialized architectures that capture complex project dynamics. ML often offers a better alternative to EVM when handling modern project complexities. However, as project environments continue to produce increasingly voluminous and unstructured data, some practitioners and researchers are turning to DL to uncover higher-order dependencies and further improve forecasting precision. [5]

2.3 Deep Learning for Project Cost and Duration Forecasting

2.3.1 Evolution and Core Concepts of Deep Learning

The history of neural networks dates back to the 1940s, when the first models were developed to simulate the way the human brain processes information. The initial limitations of small training datasets and slow processing speeds of early computing hardware also hindered progress in DL. These restrictions were progressively transcended with the growth of larger datasets and faster processing technologies. In 2006, researchers proposed new approaches to train deep neural networks that tackled the vanishing gradient issue: the introduction of activation functions, such as ReLU, and optimization methods, like Adam, both of which improved the training process. Since then, DL has been progressing rapidly, propelled by advancements in algorithms and the availability of large compute resources. These breakthroughs rekindled the interest of researchers in neural networks, leading to their adoption in many fields over the subsequent years, including computer vision (image recognition), natural language processing (machine translation), and time series forecasting (stock market prediction), to name just a few. (Figure 1) [26]

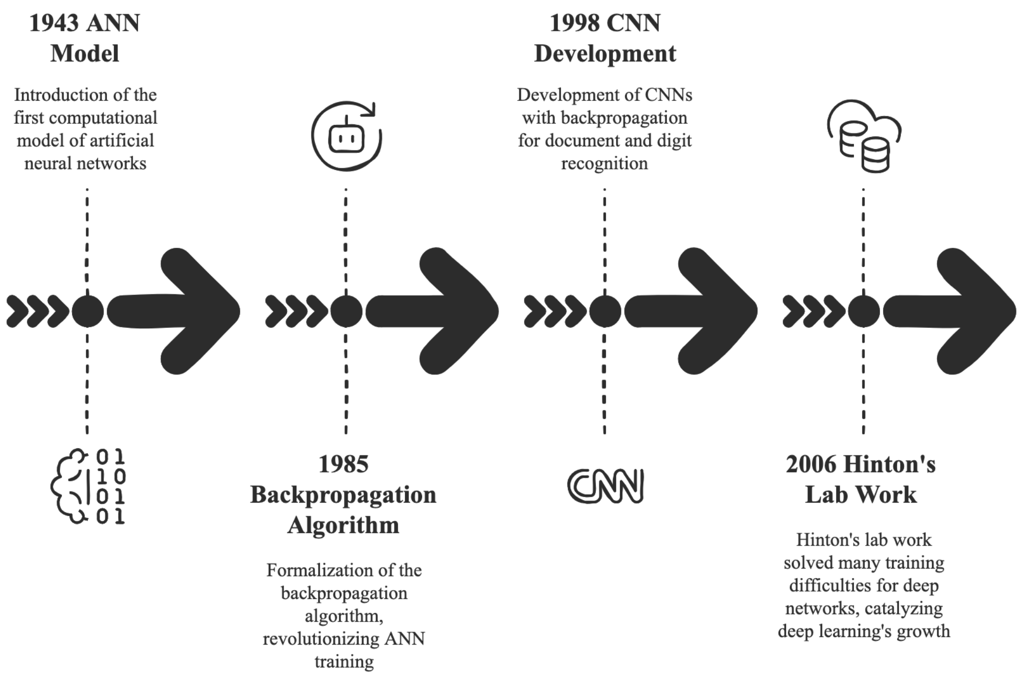

Figure 1. Key breakthroughs in Deep Learning

At its core, DL relies on the concept of ANNs, which are inspired by the way neurons communicate and transmit signals in the human brain. Traditional ANNs typically have a single hidden layer, which limits their ability to model complex patterns. In contrast, deep neural networks utilize multiple layers, allowing them to capture intricate patterns and relationships in data, making them more suitable for complex tasks. Each neuron calculates the sum of its inputs, adjusts them by weights, and then applies a non-linear activation function, such as ReLU, sigmoid, or tanh. This process allows the network to learn complex patterns and better understand the input data. (Figure 2) [27]

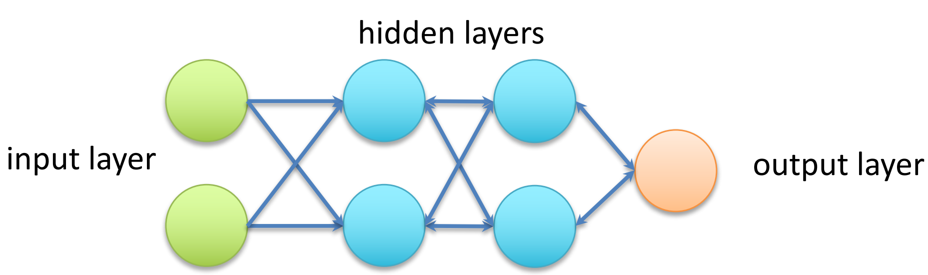

Figure 2. Artificial Neural Network base architecture

The key to this multilayered approach is the backpropagation algorithm, which adjusts neuron weights based on the errors between the network's predictions and the actual target values. During this process, deep neural networks learn to associate inputs with desired outputs, such as predicting project costs or estimating delivery times. This training process is enhanced by large datasets, which enable the layers to identify patterns and characteristics in the data. [28]

A defining trait of DL is its ability to automate feature extraction, reducing manual engineering and helping the model discover complex patterns in raw data. Instead of depending on domain experts to specify which attributes to use, deep neural networks learn directly from raw data, identifying the features that are most effective for making accurate predictions. The versatility of DL allows for flexibility with various input formats, including numerical time series, unstructured text, and images. [27]

As promising as DL is, its multi-layer architecture presents specific challenges. Training deep networks entails high computational costs and specialized hardware (e.g., GPUs or TPUs). [7] The proliferation of parameters in these networks can lead to overfitting if data is scarce or if hyperparameters are poorly chosen. While DL excels at identifying hidden correlations, it can be difficult for end users to understand. The complexity of deep learning models raises concerns about result interpretability and stakeholder confidence. [25]

2.3.2 Types of Deep Learning Approaches

DL includes various architectures and learning methods, each designed to meet specific data characteristics and domain needs. These architectures are categorized based on their learning paradigms: supervised learning for labeled data, unsupervised learning for unlabeled data, and semi-supervised learning for partially labeled data. Furthermore, these approaches vary in their treatment of temporal and spatial relationships within the input data, which influences their application in different contexts. [26]

Supervised Learning

Supervised learning involves training models on labeled datasets, where each input sample is paired with a corresponding output (or target). In DL, supervised learning is particularly effective when there is an abundance of well-structured and accurately annotated historical data, such as true cost values or project completion times. During training, a supervised model maps input features to their known outputs, measures errors, adjusts weights to minimize them, and learns the patterns and relationships between input features and known outputs. As such, supervised DL forms the backbone of many forecasting and classification tasks, where both the predictions (cost, schedule) and input data are well defined. [26]

In project management, labeled datasets often include detailed descriptions of completed projects, such as interim milestones, budget entries, and final outcomes. When combined with an appropriate supervised DL architecture, these records form the basis for forecasting models that are more accurate than simpler methods. Deep Neural Networks (DNNs) perform well with structured, multi-dimensional data. In contrast, Convolutional Neural Networks (CNNs) are most effective when the input can be mapped to spatial or grid-like formats. Additionally, Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) networks, are advantageous in situations where sequential updates or event logs significantly influence the final outputs. (Table 2) [29]

Table 2. Popular Deep Learning models for supervised learning

Deep Neural Networks (DNNs)	A foundational supervised architecture commonly used for tabular data, DNNs consist of multiple layers of interconnected neurons, each transforming input data into a more abstract representation. With every pass of the backpropagation algorithm, these networks adjust their weights to reduce errors.
Convolutional Neural Networks (CNNs)	Initially designed for visual tasks, CNNs use small filters, called convolutions, which gradually detect patterns such as edges or geometric shapes. This process continues through multiple layers, resulting in more advanced feature maps that reflect higher-level abstractions.
Recurrent Neural Networks (RNNs)	RNNs effectively tackle challenges such as sentiment analysis in text, speech recognition in voice commands, and predicting stock prices in time-series data. They have an internal memory of previous inputs, which is stored as hidden states that are transferred from one time step to the next. This internal memory enables the network to seamlessly connect past information with future predictions. LSTM networks enhance this concept by adding gating mechanisms, which help to solve the problem of vanishing gradients. They can retain relevant signals from several previous time steps.

Semi-Supervised Learning

A semi-supervised learning method combines elements of both supervised and unsupervised learning by using a small amount of labeled data to guide the learning process while leveraging a larger pool of unlabeled data to improve model performance. In this setting, part of the training data is labeled while the other part is unlabeled. The model's task is to leverage both labeled and unlabeled samples to learn robust representations and make accurate predictions even when limited ground-truth information is available. This is especially useful when annotated data is scarce or expensive to generate but unlabeled data may be abundant. [26]

The core challenge in semi-supervised scenarios is to learn as much as possible from unlabeled data without steering the model in the wrong direction. In semi-supervised learning, a DL model initially learns basic relationships between input and output functions using the labeled subset. It then refines these mappings by analyzing unlabeled samples to identify patterns that complement the labeled data, enhancing its predictive capabilities. This iterative process allows the model to generate more confident predictions on unlabeled data, effectively broadening its understanding of the entire dataset. [26]

Table 3. Popular Deep Learning models for semi-supervised learning

Generative Adversarial Networks (GANs)	The main components are a generator and a discriminator. The generator learns to create synthetic examples that replicate real data, and the discriminator attempts to distinguish between these synthetic and authentic samples. Both the generator and discriminator are continuously improved via a dynamic training process.
Deep Reinforcement Learning (DRL)	Unlike conventional ML that rely on a large set of explicit input-output pairs, DRL is based on learning by taking actions and receiving feedback from the environment. The model gradually improves by updating parameters of the deep network.

Unsupervised Learning

In unsupervised learning, the algorithm identifies patterns in a dataset without access to labeled data or guidance on the correct output. In project forecasting scenarios, such methods can be extremely valuable when detailed labeled information (such as actual final costs or completion times) is missing. [26] These approaches analyze the data to uncover hidden patterns, identify latent representations, detect outliers, and reduce dimensionality. This helps project managers or analysts draw conclusions that may be obscured in a setting where labeled data is unavailable. [23]

The model is fed raw data on which it learns without being supervised. It reveals subtle correlations within the data and simplifies complex measures such as multidimensional relationships. In unsupervised learning, algorithms perform feature extraction by analyzing the data's internal structure and distribution patterns. Unsupervised learning serves as a suitable preprocessing step or anomaly detector by identifying discrepancies in cost or schedule logs that may suggest overruns or other issues. [26]

Table 3. Popular Deep Learning models for unsupervised learning

Auto-Encoders (AEs)	Best for reducing dimensions by identifying relevant features and encoding the input data into a lower dimensional latent space, thus preserving key information. AEs are also great for detecting anomalies.
Restricted Boltzmann Machines (RBMs)	Energy-based models composed of visible and hidden layers, and links within a layer. RBMs learn a joint probability distribution over input variables by iteratively refining weights between these layers, hence applied to collaborative filtering and feature learning.
Deep Belief Networks (DBNs)	Multi-layer generative models that successively learn hierarchical representations of the input. Each layer refines the abstraction gleaned from the previous one, capturing increasingly complex data structures.

Chapter 3. Methodology

This chapter presents the research design and methodology followed to develop, train, and evaluate the DL models for project cost and duration prediction. It starts by describing the high-level research approach before describing the data sources and preprocessing steps, model architectures, and evaluation metrics. These details provide a clear, reproducible framework that other researchers or practitioners can use to replicate or build upon the experiments outlined in this research.

3.1 Research Design and Tools

Given the aim of this thesis, which is to compare the performance of a DL model against a conventional machine learning benchmark, the study adopts the following research design (Figure 3). The key steps in the research design include:

- **Data Acquisition.** Gathering project data from two publicly available, EVM-focused datasets.
- **Data Preprocessing and Feature Engineering.** Ensuring consistency across datasets by cleaning, transforming, and augmenting feature sets.
- **Model Development.** Constructing the DL architecture and configuring the benchmark model for fair comparison.
- **Model Training and Validation.** Using standardized procedures to assess predictive performance.
- **Performance Comparison.** Employing relevant metrics to compare predictive accuracy, precision, and stability at different project stages (early, mid, and late).

By applying the same data and evaluation metrics to different forecasting methods, this study ensures that any performance differences can be attributed primarily to differences in the modeling techniques.

By incorporating hyperparameter tuning and feature engineering, the method refines models in a systematic manner.

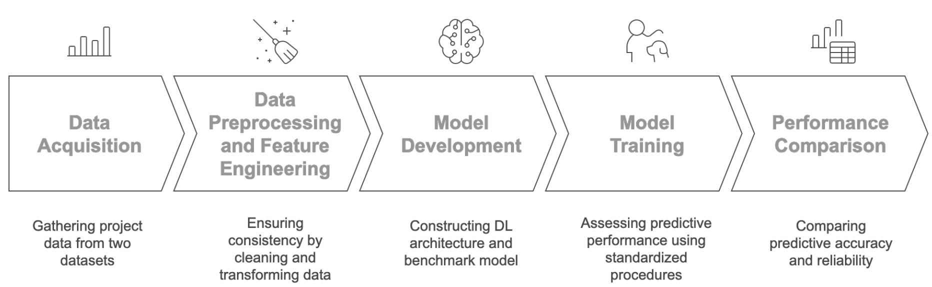

Figure 3. Thesis research design sequence

Although the datasets are domain-specific, the methodology for data preprocessing, model training and evaluation can be adapted to other project contexts if comparable data is available.

All experiments in this research were conducted using the Python programming language (version 3.10.7). Python's extensive ecosystem of libraries provides robust capabilities for data analysis, ML, and DL. Python scripts in Jupyter notebooks were primarily run in a local development environment (Microsoft Visual Studio Code), with occasional use of cloud platforms (Google Colab) for sharing and additional computational resources. Below are the key tools and modules employed (Table 4).

Table 4. Key tools and modules

Data Handling and Manipulation	
Pandas	reading, merging, transforming, and cleaning tabular data files [30]
Numpy	handling numerical arrays and performing vectorized computations [31]
Os	directory and file operations (e.g., navigating file paths) [32]
Re	pattern matching and data cleaning tasks involving regular (text) expressions [32]
Scipy.interpolate	interpolating intermediate data points, used in calculating regularity indicator [33]
NetworkX	used for any potential analysis or visualization of project task dependencies in a network structure, used in calculating series/parallel indicator [34]
Visualization and Exploratory Analysis	
Matplotlib	static plots and figures to illustrate data distributions and results [35]
Plotly (graph objects, plotly express)	generates interactive charts and dashboards, offering dynamic data exploration [36]
Seaborn	provides advanced statistical graphics and correlation heatmaps [37]
Phi_K correlation matrix	assists in identifying both linear and non-linear relationships [38]
Machine Learning (Scikit-learn modules)	
Optuna	employed for automated hyperparameter tuning [39]
Xgboost Regressor	gradient boosting library for regression tasks, serves as the baseline ML model [40]
Cross_val_score and **Kfold**	systematic model evaluation and data partitioning [41]
OneHotEncoder and **StandardScaler**	transforming categorical features and normalizing numeric variables [41]
Metrics (r2_score, mean_absolute_percentage_error, root_mean_squared_error)	model performance measurement [41]
Deep Learning (Keras with TensorFlow as a backend)	
Sequential, Model	build various neural network architectures (MLP, LSTM, CNN layers) [42]
Dense, LSTM, Dropout, Conv1D, MaxPooling1D	layers for creating and experimenting with feed-forward or recurrent models [42]
Optimizers (Adam)	manage the learning process [42]
Callbacks (ModelCheckpoint, EarlyStopping)	automatically saving the best model and early stopping during training [42]

3.2 Data Sources and Description

The Dynamic Scheduling Library (DSLIB) consists of 181 real-life project datasets explicitly designed for scheduling, project control, and EVM research. As highlighted in prior studies [8], each dataset includes comprehensive details. (Table 5)

Table 5. Overview of the parameters used to classify the empirical projects of DSLIB

Completeness	indicates the completeness of the baseline schedule, schedule risk analysis and project control information
Authenticity	authenticity of all project progress data for each tracking period
General Information	number of activities in the project, planned duration and planned costs (budget at completion) of the project schedule
Resource	details whether the resource information is available, renewable and/or consumable
Tracking Data	updates on project status, including intermediate milestones or performance metrics
Network Topology	includes serial/parallel indicator, activity distribution indicator, length of arcs indicator, topological float indicator, regularity indicator and its classification index
Time, Cost and Resource Sensitivity	contains information about cruciality indices. Indicators show how changes in schedule or expenses might impact overall performance
Performance Metrics	actual duration and cost incurred, as well as EVM-based measures such as EV, PV, AC, CPI, SPI, CV, SV, p-factor
Project Control Data	comprehensive EVM performance measures recorded at each tracking period, enabling a granular analysis of project progression

In contrast to DSLIB, the Project Portfolio Dataset (Australian dataset) comprises eight project data subsets, each fully describing a completed project's original baseline plan and actual progression. [9] Smaller in size than DSLIB, these datasets do not offer as much comprehensive information, but still contain crucial information about project timelines, budgets, and outcomes:

- Essential EVM Metrics – EV, PV, AC, BAC alongside planned and actual duration data.
- Baseline cost estimates, schedule breakdowns, and final performance outcomes for each project.

The dataset offers an alternative viewpoint on project evolution under different organizational or regional norms. Inclusion of these additional cases gives the research comparative depth and model testing robustness.

While they are well-suited for EVM-based research, both the DSLIB and Australian datasets have their own unique challenges:

- **Incomplete or Missing Fields.** Not all projects in DSLIB or the Australian datasets provide the full range of possible attributes (e.g., some may lack detailed resource data or intermediate progress information).
- **Non-Uniform Metrics.** DSLIB has duration represented in terms of days, while the Australian projects have duration measured in terms of tracking periods with different durations. To address these discrepancies, it is crucial to implement extensive data cleaning and to combine all the data into a common format before performing any comparative analysis.

These limitations emphasize the need for a systematic pre-processing step that takes care of unit conversions, renaming attributes, and potentially adding derived variables (for example, converting from periods to days or the other way around). These steps guarantee that both datasets are comparable and can be passed into the same modeling pipeline seen in sections 3.6 Data Preprocessing and 3.9 Model Development.

3.3 Data Filtering (Complete and Authentic Projects)

In this study, accurate project forecasting relies heavily on the quality of the data provided. [24] To ensure the data used in this study is both complete and valid, we have adopted a filtering method.

All eight project subsets in the Australian collection are complete and authentic, reporting all baseline plans and ongoing progress for each project from start to finish, including EV, PV, AC, and BAC). [9]

In the DSLIB collection, Completeness is defined by having data in the Real Duration and Real Cost columns, as well as Project Control parameters (EVM indicators for all tracking periods).

The Project and Tracking columns are used to validate authenticity, ensuring that each dataset contains actual, not simulated, project records. [8]

Applying the criteria of completeness and authenticity reduced the total number from 181 to 98 valid projects, comprising 54% of the original library. Since all Australian projects met the same requirements, they were included in the final dataset, resulting in a combined total of 106 projects (98 from DSLIB + 8 from the Australian dataset).

All Australian projects belong to the Construction (civil) sector. Figure 4 illustrates the sector distribution of the filtered projects from DSLIB, highlighting the construction sector as the largest contributor.

Sector	Count
Construction (residential)	38
IT	19
Construction (civil)	16
Construction (commercial)	8
Construction (institutional)	7
Construction (industrial)	3
Event Management	3
Mobility	2
Engineering	1
Education	1
TOTAL	**98**

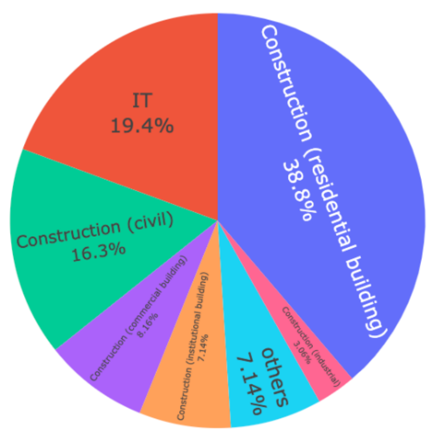

Figure 4. Sector distribution of filtered projects from DSLIB

Furthermore, it is important to note that, sectors containing only one, two, or three projects complicate the process of achieving a balanced split for training and testing subsets. This issue will be addressed in detail in the relevant section (3.6 Preprocessing).

3.4 Feature Selection and Feature Engineering

3.4.1 Feature Selection

Accurate prediction models for project cost and duration forecasting rely on well-chosen and systematically engineered features representing various project performance factors. [25] Overly narrow feature sets may exclude relevant variables, which can lead to underfitting due to the model's inability to capture the underlying patterns in the data. Conversely, too many or unfiltered features may inject noise or redundant information into the model, which can obscure the true relationships in the data and lead to unreliable predictions.

We formed a combined dataset of 1 252 tracking periods by merging the filtered DSLIB and Australian project datasets, which are significant due to their comprehensive coverage of project performance metrics, providing a robust sample size for analysis. Each project-level and tracking-period attribute was evaluated for its relevance to cost and duration forecasting. This evaluation resulted in a refined set of features, summarized in Table 6, which details core project descriptors and dynamic EVM metrics. Together, they enable the models to incorporate cost baselines, schedule performance, and activity structures into their predictive framework.

Table 6. Selected features

Project-Level Features	
code	unique project ID, facilitates merging and cross-referencing of data across multiple tables or worksheets (not used as a predictive variable)
name	descriptive name of the project, useful for documentation (not used as a predictive variable)
sector	industry or domain to which the project belongs (e.g., Construction, IT), potentially relevant for modeling if certain sectors exhibit unique cost/schedule patterns.
bac (Budget at Completion)	the total planned budget for the project, used for cost normalization and feature engineering
pd (Planned Duration)	the initially scheduled timeline for project completion, used as the basis for time-based normalization
sp (Serial/Parallel Indicator)	reflects the degree to which project activities are arranged sequentially versus concurrently, calculated from project network and baseline schedule data
ri (Regularity Indicator)	measures the consistency or “smoothness” of the activity network, indicating how uniformly tasks are distributed
Tracking-Period Features	
period	time-step reference indicating the specific tracking interval
pv (Planned Value)	the budgeted value of work scheduled at the given tracking period
ev (Earned Value)	the budgeted value of work actually completed by the tracking period
ac (Actual Cost)	the real cost incurred at the given tracking period
cpi (Cost Performance Index)	a ratio of EV to AC, indicating cost efficiency at each tracking period
spi (Schedule Performance Index)	a ratio of EV to PV, reflecting schedule efficiency over time

spi_t (Schedule Performance at Period t)	a time-based schedule performance measure derived from Earned Schedule principles
cv (Cost Variance)	EV − AC, expressing how much the project is under or over budget at each period
sv (Schedule Variance)	EV − PV, indicating how far ahead or behind schedule the project is in cost terms
sv_t (Schedule Variance at Period t)	a variant of schedule variance that incorporates time-based factors
Target Features	
ceac (Cost Estimate at Completion)	total projected cost upon project completion
teac (Time Estimate at Completion)	total duration by which the project will finish

3.4.2 Data Harmonization Across Sources

The DSLIB dataset contained extensive EVM indicators, whereas the Australian datasets were missing many advanced metrics such as CPI, SPI, Serial/Parallel Indicator (SP), and Regularity Indicator (RI). Manual calculation and feature engineering techniques were used to backfill missing fields.

One of the main challenges in this process was to maintain consistency across all the datasets, particularly while recomputing and validating the features derived from DSLIB. The proprietary software used in DSLIB, ProTrack (inaccessible at the time of conducting research at https://www.protrack.be/), often produced results that deviated from standard theoretical formulas (see Section 2.1 Earned Value Management) and outputs generated by the Earned Schedule Calculator [43]. Furthermore, the Earned Schedule Calculator itself was unable to process the numerous tracking periods on large DSLIB projects. In order to standardize calculations and minimize discrepancies that may occur between tools, all performance metrics of interest were recalculated using base EVM equations based on the metrics EV, PV and AC. This standardization ensured consistency across both data sources, reinforcing the validity of the forecasting models.

Another major consideration in feature engineering (which was already mentioned in section 3.2 Data Sources and Description) was non-uniform measurement units across datasets. As an example, DSLIB usually describes durations as a number of days, whereas Australian projects describe durations in tracking periods that are of differing lengths. The differences highlighted the need for extensive data cleaning and conversion of units into a common format – the first step to allow for comparative analysis. We maintained the consistency of cost and schedule performance indicators across various project environments by going through this preprocessing.

3.4.3 Graph-Based Features (SP, RI)

In this research, Serial/Parallel Indicator (SP) and Regularity Indicator (RI) were integrated into project-level features, in addition to the standard cost and schedule metrics. These indicators provide structural insights into project execution patterns and require graph-based analyses of baseline schedules. Directed acyclic graphs were constructed to model project task dependencies, and algorithms were implemented to quantify network seriality (SP) and schedule uniformity (RI). [19]

The RI measures scheduling irregularities, quantifying how much a project's PV curve deviates from a perfectly linear trajectory. The calculation process is as follows: [19, 44]

$$RI = \frac{\sum_{i=1}^{r} m_i - \sum_{i=1}^{r} a_i}{\sum_{i=1}^{r} m_i} \tag{8}$$

where,
m_i – maximal possible deviation from a perfectly linear PV curve
a_i – actual deviation of the project's PV-curve from a perfectly linear curve
r – number of equidistant evaluation points

$$m_i = \max\{BAC - PV_{lin_i}, PV_{lin_i} - 0\} \tag{9}$$
$$a_i = |PV_{real_i} - PV_{lin_i}| \tag{10}$$

where,
PV_{lin_i} – linear PV curve at point i
PV_{real_i} – actual PV curve at point i

The r is determined by the PV curve rather than the number of tracking periods. As PV trajectories become more complex, larger values of r are necessary to accurately characterize these trajectories. [19] In this study, r ranged from 50 to 120 across projects. Figure 5 illustrates an example of RI calculation for one of the analyzed projects.

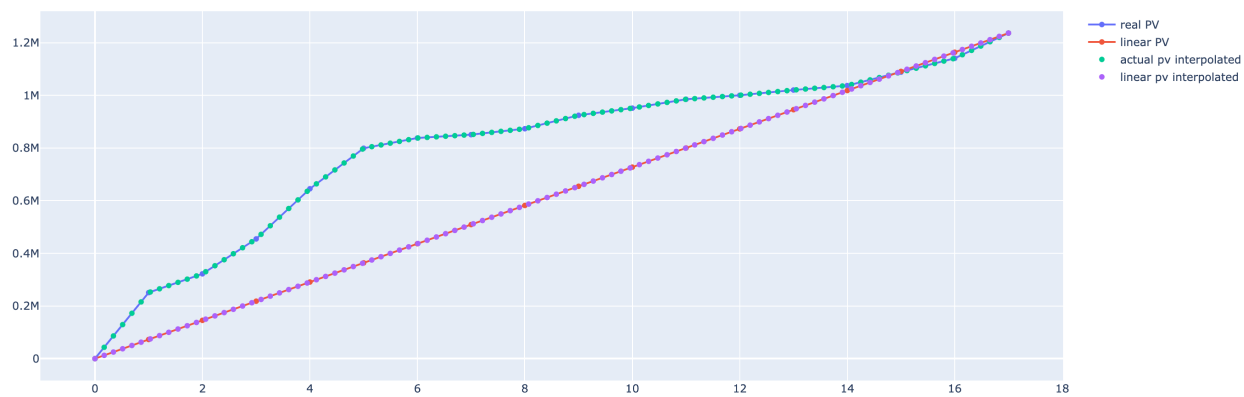

Figure 5. Example of calculating Regularity Indicator for one of the projects

The SP is used to measure the extent of task parallelization in a project schedule. The calculation is as follows: [19]

$$SP = \frac{n_s - 1}{n_t - 1} \tag{11}$$

where,
n_s – maximum number of subsequent activities in the network (or the maximum progressive level)
n_t – total number of activities

This measure reflects how closely the structure of a project resembles a strictly serial or parallel workflow. Figure 6 demonstrates an example of SP computation using the NetworkX module. [34]

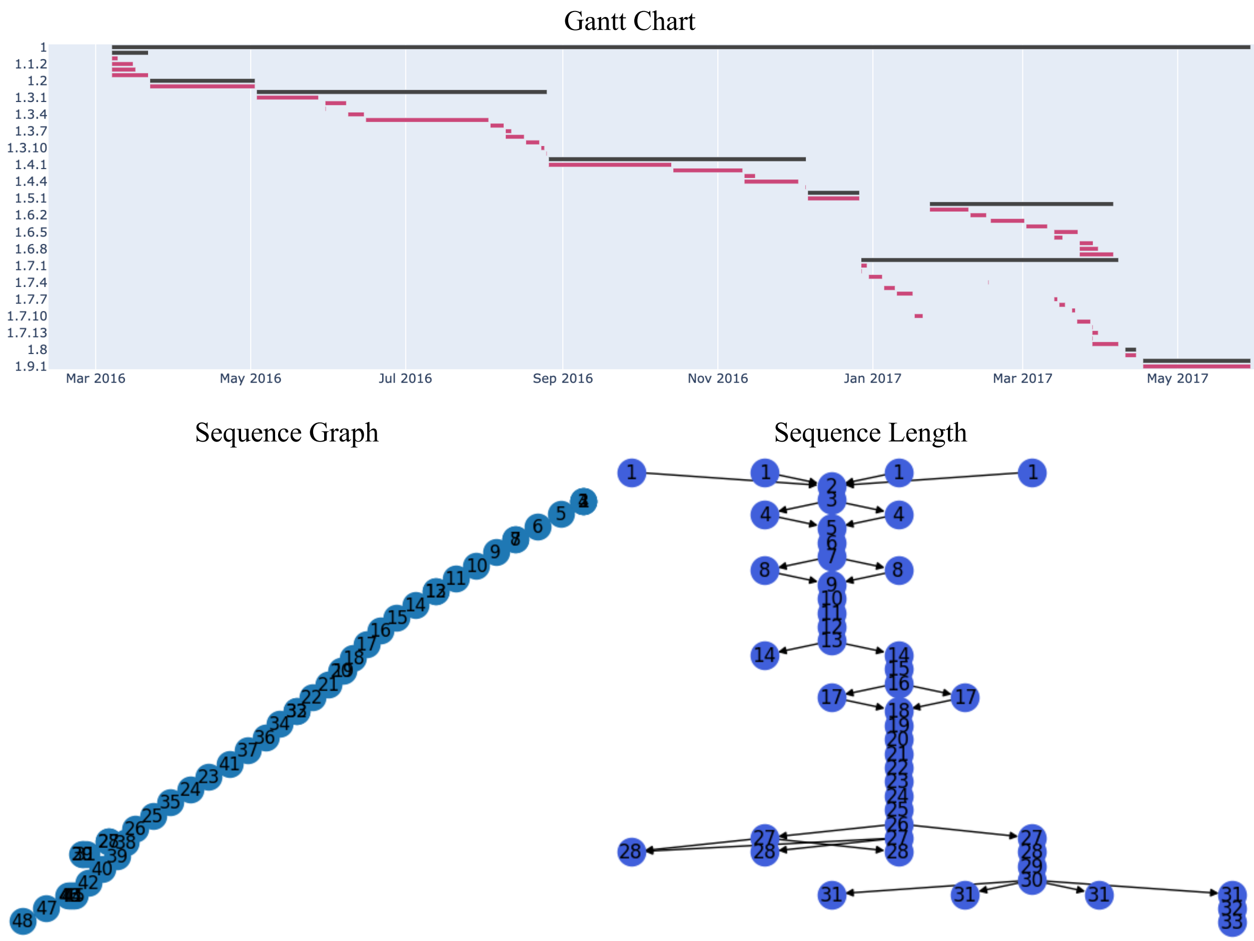

Figure 6. Example of calculating Series/Parallel Indicator for one of the projects using NetworkX module

The calculation of SP and RI was computationally intensive, as it involved parsing, validating, and measuring the baseline network of each project. The complexity of these computations is part of the reason other potentially useful metrics (e.g., topology network, p-factor) were excluded from the study's focus. Despite their computational demands, SP and RI provide fundamental insights into structural dynamics. They reveal patterns of task parallelization and schedule uniformity that directly impact cost and duration risks.

By applying uniform formulas and preprocessing techniques across both data sources, we successfully integrated DSLIB and Australian datasets into a single, coherent dataset for modeling. This approach ensured:

- Consistency across projects.
- Elimination of data inconsistencies specifically from different practices of data collection or software-extracted outputs.
- Uniform contribution of tracking periods to model training and validation, preventing bias in predictive outputs.

3.5 Outlier Detection and Normalization

The next step of the process was to refine the data further by removing outliers with statistical approach. This process helps ensure that abnormally high or low values do not disproportionately affect the forecasting models, and that cost and time metrics are comparable across projects. [11]

3.5.1 Outlier Detection

The distributions of the BAC and PD of each project were reviewed to detect potential outliers. Using histograms and box plots, the range of project values was visualized to determine thresholds for outlier detection (Figure 7). The analysis identified projects with budgets exceeding 3.5 million and planned durations surpassing 475 days as outliers.

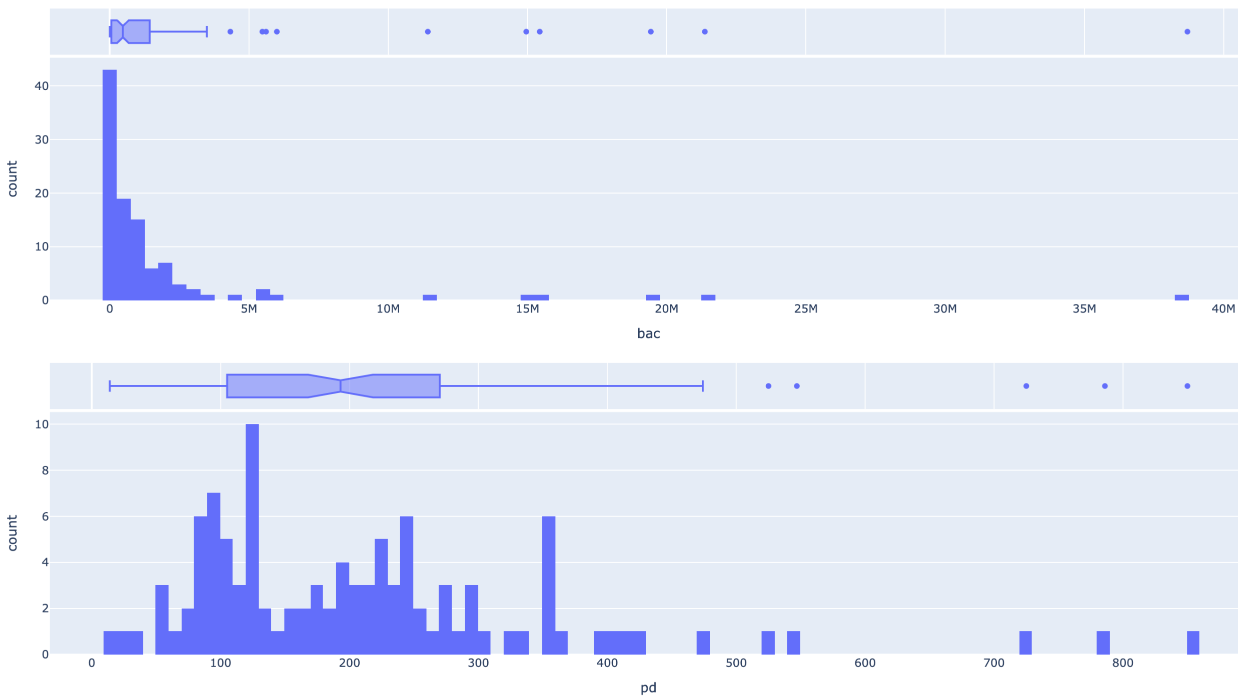

Figure 7. Box plots and histograms highlighting BAC and PD outliers

From these visualizations, 11 projects (accounting for 431 tracking periods) were identified as falling outside the statistically determined boundaries for budget and duration. All of these outliers originated from the DSLIB dataset. After their removal, the dataset was reduced to 95 projects, consisting of 87 projects from DSLIB and 8 from the Australian dataset.

3.5.2 Correlation Analysis with Φk (phi k) Matrix

To further analyze feature relationships, a Φk (phi k) correlation matrix was generated to detect both linear and non-linear dependencies, as illustrated in Figure 8: [38]

- Strong clustering among cost-related features (BAC, AC, EV, PV, CEAC) indicates high degree of interdependence.
- High correlations between time-based variables (PD, TEAC) reflect the interconnected nature of project scheduling.
- The dominance of BAC and PD in multiple correlations suggests that larger projects with higher budgets could overshadow smaller projects if left unscaled.

The findings emphasize the need to normalize cost and time variables to ensure fair comparisons across projects of different sizes. [11]

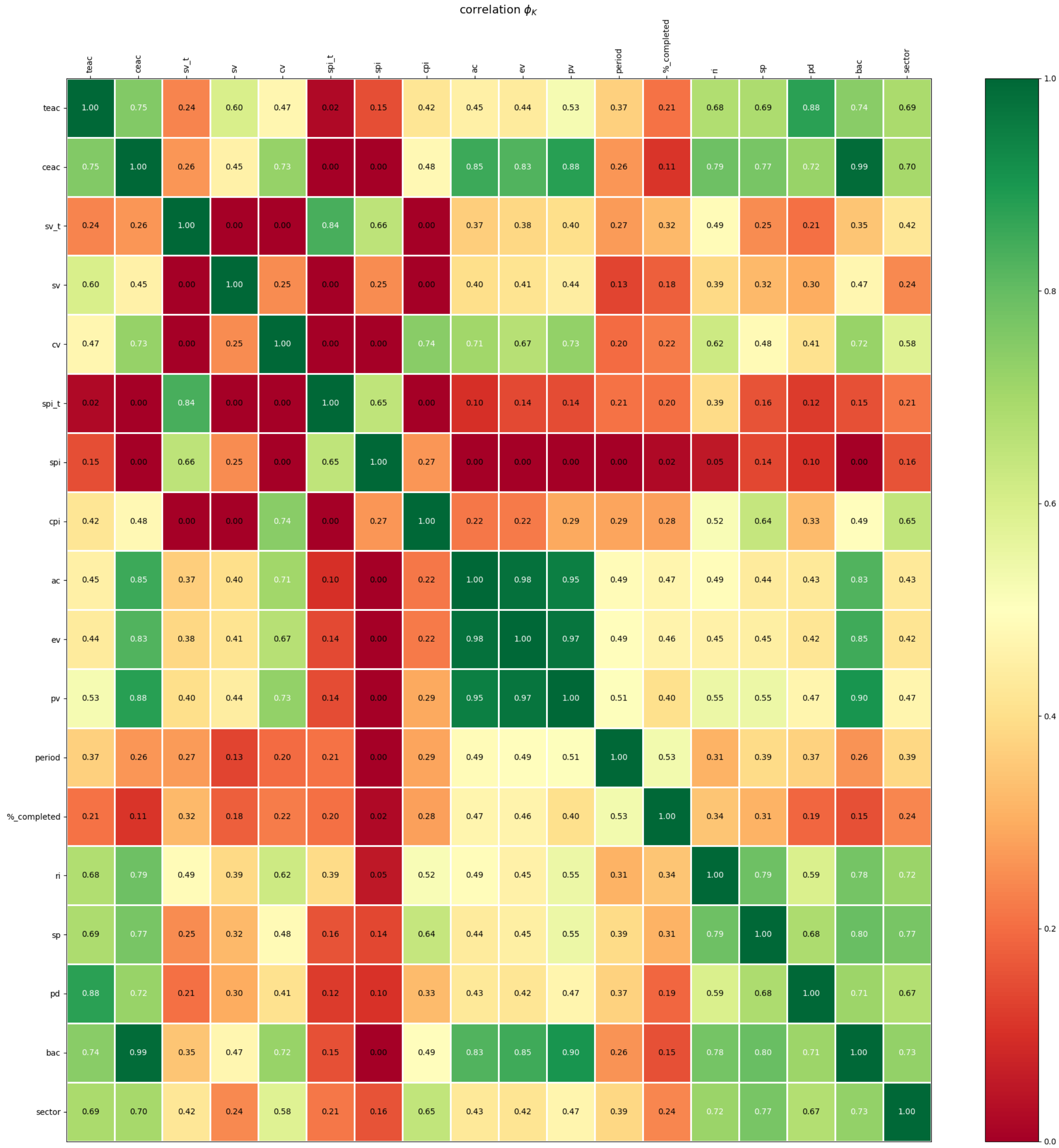

Figure 8. Φk correlation matrix before normalization

3.5.3 Normalization Strategy

To address the impact of variations in project size and duration, we implemented a structured normalization approach. For cost normalization, key financial metrics such as CEAC, PV, EV, and AC were adjusted by dividing them by each project’s BAC. This transformation normalized each project's planned budget to a baseline value of 1, effectively eliminating discrepancies caused by varying project sizes. After using BAC as a scaling reference, it was removed from the dataset to avoid redundancy.

Similarly, for time normalization, all time-related features were standardized by dividing them by each project's PD. This process aligned the planned schedule of each project to a uniform scale of 1, so it allowed for meaningful comparisons across projects with different timeframes. As PD was used exclusively as a normalization factor, it was also excluded from the dataset after processing.

This dual-step approach reduced the focus on large-scale projects while maintaining clear relationships between cost and time variables. After normalization, the updated Φk correlation matrix (Figure 9) revealed:

- A continued strong correlation of target features with sector, SP and RI.
- TEAC exhibiting strong correlation with SV and period.
- CEAC maintaining strong associations with CV, CPI, and AC.

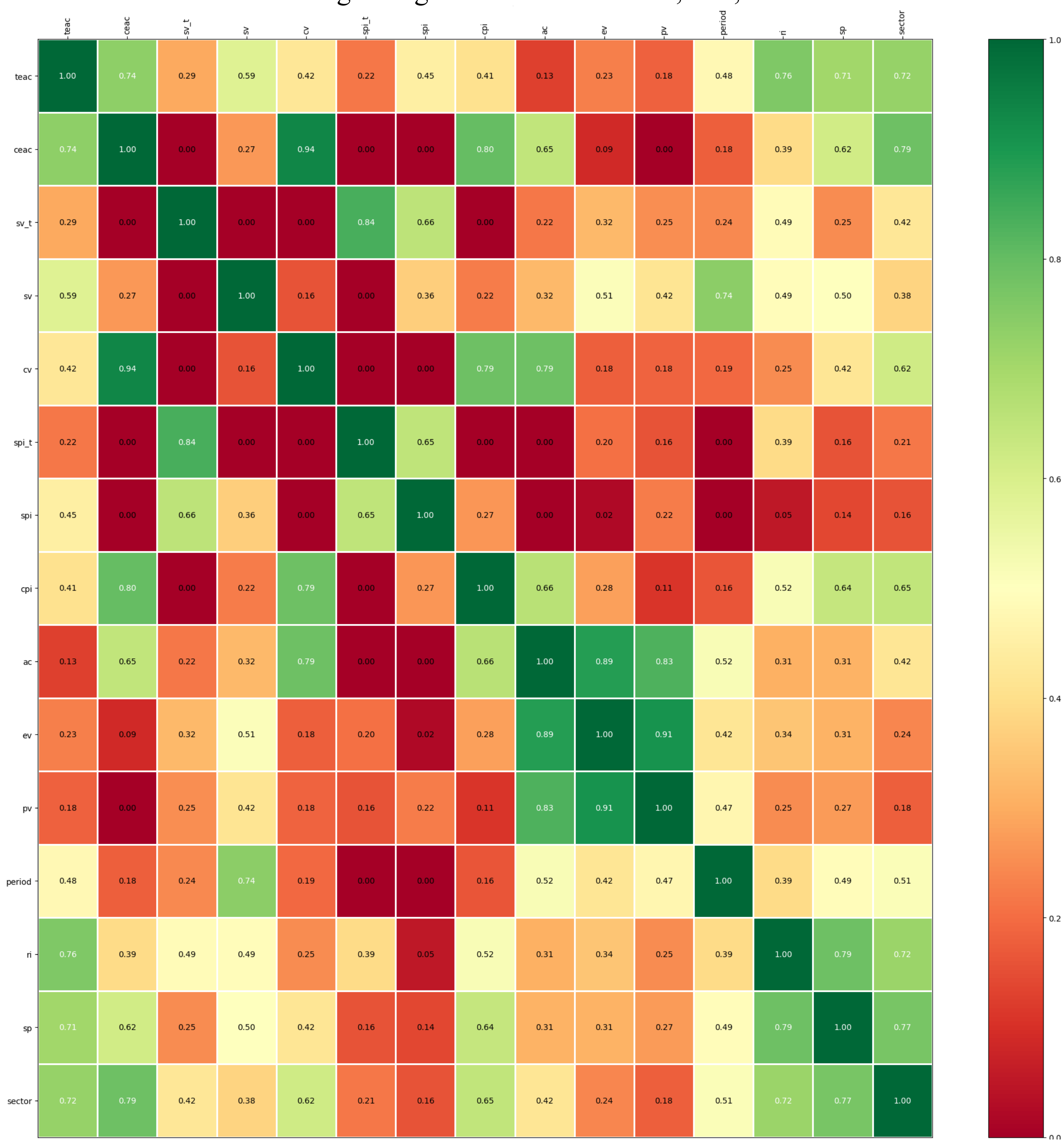

Figure 9. Φk correlation matrix after normalization

3.6 Preprocessing

After normalizing cost and time variables, the last step of data preparation consists of splitting the dataset into train and test subsets, as well as encoding and scaling procedures. These steps ensure that all models are trained and evaluated in the same way to enable direct comparison amongst scenarios.

3.6.1 Train-Test Split

Because the dataset merges both DSLIB and Australian projects, it was important to split the merged dataset into train and test subsets in a balanced manner, such that exactly 2 out of 8 Australian projects were included in the test subset.

Several constraints guided the train-test split: (Table 7)

- A pseudo-random sampling approach was taken using a pre-determined random state parameter to ensure reproducibility of these experiments. [45]
- Each sector, described in section 3.2 Data Sources and Description, was guaranteed at least one representative in the training set to assure that all sector-specific characteristics were available for model learning.
- Tracking periods within each project were not separated from each other between train and test subsets, ensuring the model encounters an entirely new project in the test subset rather than partial data from already-seen projects in the train subset.
- The test set size was constrained to range between 20% to 25% of the unified dataset.
- Due to the small size of the dataset, a separate validation subset was omitted. Instead, the model evaluates its performance using cross-validation on the training set, so robust evaluation occurs without needing to split the data further.

Table 7. Train-test split distribution

	Train	Test
Projects	75 (6 Au)	20 (2 Au)
Tracking Periods	628 (76,49%)	193 (23,51%)

The dataset contains two target features – CEAC (cost) and TEAC (time). Although modern deep neural networks can predict multiple outputs simultaneously, these targets were modeled separately to improve interpretability and isolate cost-forecasting from schedule-forecasting performance. Models with the same architecture will be trained twice: once for cost and once for time.

3.6.2 Encoding and Final Scaling

The single categorical feature – sector, which identifies each project's domain (e.g., Construction, IT), is significant for understanding domain-specific patterns. OneHotEncoder was used to convert this categorical variable into a numerical one. This approach creates binary columns for each unique sector and assigns it a value 1 if the project belongs to that sector and 0 otherwise. [46] OneHotEncoder avoids creating an artificial ordinal relationship that could mislead the model. [47]

Although cost and time variables were normalized with respect to their BAC and PD (see section 3.5 Outlier Detection and Normalization), DL models also benefit from additional standard scaling, which adjusts each numerical feature to have a mean of zero and a standard deviation of one, thus, reducing biases in neural network training (particularly in the initial layers). [26]

Most importantly, encoder and scaler were fitted on the training subset, and only then applied to the test subset separately, ensuring that the test data remained unseen during the fitting process and preventing data leakage. [46]

3.7 Baseline models

Before developing the DL model, it is essential to establish reliable reference points that will serve as a baseline for a comparing traditional, easily interpretable methods with more complicated ones. EVM was selected as a foundational method due to its widespread use. Additionally, XGBoost serves as a ML benchmark, known for its strong performance. [24] By measuring how well deep neural networks can outperform these two baselines, the research will demonstrate whether DL really represents a step forward in project forecasting.

3.7.1 EVM Forecasting

Despite the limitations noted in Section 2.1, EVM remains one of the most widely adopted techniques for project control and performance monitoring. It has a rather straightforward implementation, requiring only most common periodic measurements of EV, PV, and AC. With a few simple formulas, EVM can calculate forecast estimates. The method enhances transparency and helps position more complex approaches, such as XGBoost and deep neural networks, within a familiar project management framework.

EVM cost forecast formula: [4]

$$CEAC = AC_t + \frac{BAC - EV_t}{PF} \tag{12}$$

where,
$CEAC$ – cost estimate at completion
AC_t – actual cost at tracking period t
BAC – budget at completion of the baseline schedule
EV_t – earned value at tracking period t
PF – performance factor, capturing assumptions about how future progress will evolve

Time-based forecasts follow a similar logic:

$$TEAC = t + \frac{PD - ES_t}{PF} \tag{13}$$

where,
$TEAC$ – time estimate at completion
t – tracking period at question
PD – planned duration pf the project in the baseline schedule
ES_t – earned schedule at tracking period t
PF – performance factor, used to adjust the remaining duration

The choice of performance factor (PF) has a critical impact on cost and time prediction. PF determines the extent to which past performance trends are expected to persist in the future project phases. The common options for PF are:

- $PF = 1$ – assumes that past problems or opportunities will not affect future work, meaning that the project will revert to a baseline plan for the remaining tasks. According to the studies, this choice is the best for time forecasting, and will be used in this research. [48]
- $PF_{cost} = CPI$ and $PF_{time} = SPI(t)$ – uses observed performance to indicate future efficiency or inefficiency. This scenario best matches real-world conditions, when the project team would continue to work at approximately the same efficiency with which it had been working up to that point. This option is the best choice for cost forecasting. [24]
- $PF = CPI * SPI(t)$ – assumes a strong interdependence between cost and schedule performance, suggesting that past variances in either will compound and affect the project's future.

3.7.2 XGBoost

A central objective of this research is to compare the performance of DL methods with a robust, well-regarded ML benchmark. Previous studies have repeatedly identified XGBoost as one of the top-performing algorithms for structured data tasks, including project cost forecasting. It outperforms many traditional models such as linear regression, random forests, and simple time-series approaches. [24] Consequently, XGBoost was selected as the baseline mode.

Establishing a high-quality baseline is essential for providing a meaningful reference point against which the performance of proposed models can be measured. [48] By using XGBoost, a proven model, the study sets a high standard that DL models must exceed to show real advantages.

While XGBoost often performs well in its default configuration, hyperparameter tuning can significantly improve predictive accuracy. [23] In this research, an advanced optimization framework Optuna systematically explores the hyperparameter space to achieve a balance between model complexity and overfitting prevention. [39] Through a sequence of 100 trials for each target variable (cost and time), Optuna evaluates candidate configurations using tenfold cross-validation on the training subset. At each iteration, it collects performance metrics and refines its estimates of the best hyperparameter set (Table 8).

Table 8. Key hyperparameters tuned for XGBoost

Hyperparameter	Description
learning_rate	Controls speed of the model's learning. A smaller value results in slower convergence, but improves overall precision.
n_estimators	Specifies the number of boosting iterations (trees). Increased values can improve accuracy, but they may also lead to overfitting.
min_child_weight	Sets the minimum sum of instance weights required in a leaf, which helps control overfitting by ensuring that leaves have enough instances, thus preventing the model from learning overly specific patterns.

These parameters were chosen because they significantly impact model performance, especially for imbalanced or multi-dimensional datasets. [40]

For the baseline model, domain expertise plays an especially pivotal role, given that conventional ML algorithms do not perform automated feature extraction as neural networks do. [5] A curated set of essential features, namely PV, EV, AC, period, and sector, was selected based on established project management practices and prior research indicating their predictive value. [22] By focusing on these high-impact variables, the XGBoost baseline can concentrate on proven cost and schedule drivers, potentially raising the performance bar for the DL approaches even higher.

3.8 Neural Network Setup

3.8.1 DL Type Consideration

MLP was chosen as the principal DL architecture for this study due to its suitability for tabular datasets. Unlike specialized networks such as CNNs, which excel at image-like data, or RNNs, which are geared toward sequential information, an MLP provides a more straightforward but still powerful approach for cost and duration forecasting in project management. [29]

Early exploration included testing LSTM networks, which are designed for longer, uniformly spaced sequences. However, the current dataset is not well-suited for a recurrent-only approach, as some projects contain as few as three tracking periods, while many others have a relatively short duration. These factors negate the advantages of LSTM or similar RNN variants. [10]

MLP design emerges as a simpler and more robust solution for project forecasting in this context. Although specialized networks can excel in domains with rich sequence data, the sparse and irregular time steps in the present dataset favor the MLP's ability to combine static project-level features with dynamic performance metrics. Consequently, MLP, with its fully connected layers and ability to capture non-linear interactions, strikes a practical balance between simplicity and predictive capacity. [10]

3.8.2 Core Architecture Decisions

Three key decisions dictate the configuration of MLP architecture: how many hidden layers, how many neurons in each layer, and whether or not to enable a dropout layer to combat overfitting. While other hyperparameters (e.g., activation functions or optimizers) also influence performance, they typically have well-established default values or recommended practices. [7]

- **Hidden Layers.** The term "deep" learning usually means multiple hidden layers, in contrast to "shallow" approaches with one or zero hidden layers. Issues with high abstraction like computer vision or long time-series sequences, may require three or more hidden layers. [26] A few hidden layers (0, 1, or 2) are often sufficient for moderate complexity of project data. Configurations with up to 20 hidden layers were also tested to explore the benefits of deeper representations, while still being cautious of overfitting when working with relatively small datasets.
- **Number of Neurons in Each Layer.** Neurons in the hidden layers critically determine the MLP's representational power or "width". The network must have enough capacity to model the patterns governing cost and duration but should not become so large as to risk overfitting or prolonged training times. A typical heuristic suggests setting the number of neurons in the first hidden layer to approximately the sum of the input features and output targets, although this value often adjusts based on domain knowledge and preliminary experiments. Too few neurons can miss important signals in the data, leading to underfitting, whereas too many can needlessly inflate model complexity and training duration. [50]
- **Dropout Layer.** Dropout regularization selectively disables a fraction of neurons during training. This process reduces their co-adaptation and decreases the risk of overfitting. A dropout rate of 0.2 was tested as part of the architecture grid, a common value in research and industry. [51]

Table 9 outlines the core components of the MLP architecture explored in this study, detailing the number of neurons and activation functions for each layer, as well as the choice of optimizer and training parameters. [7]

Table 9. Core components of the MLP used in the research

Input Layer	
Number of neurons	The input layer typically contains a neuron for each input feature, plus one additional bias node.
Input Dim	Refers to the dimensionality of the input feature vector, ensuring the network correctly interprets the number of input features.
Hidden Layers	
Layer Count	Parameter defines the depth of a model. In addition to investigating shallow models, alternative setups with up to 20 hidden layers were examined allowing for a comprehensive analysis of how depth affects model performance.
Number of neurons	"Width" of a model. The number of neurons in each hidden layer was systematically varied from 10 to 200, which influences the model's capacity to learn complex patterns and generalize from the data.
Activation Functions	The Rectified Linear Unit (ReLU) was selected as the activation function because it is robust and helps prevent the vanishing gradient problem seen in other functions.
Dropout	Optionally inserted between layers to randomly "turn off" 20% of neurons (dropout rate set to 0.2) during training, which helps in reducing overfitting by preventing the model from becoming too reliant on any particular set of neurons, thus promoting robustness.
Output Layer	
Number of nodes	Contains 1 node, because the target variable (either cost or time at completion) is a single value.
Activation Function	Since cost and time are continuous variables, the final layer uses a linear activation.
Other parameters	
Kernel initializer	Weights in each layer are initialized from a small Gaussian distribution centered around zero ("normal"). This method, combined with ReLU activations, encourages smoother gradient flow, and avoids extremes, facilitating more stable and efficient training.
Optimizer	The Adam optimizer adaptively adjusts learning rates (default settings), offering advantages such as efficient handling of sparse gradients and requiring less tuning compared to other optimizers.
Number of epochs	Each configuration of parameters was permitted to train for a maximum of 1000 epochs.
Early stopping	This callback stops the training if the validation score fails to improve for a certain number of consecutive epochs (set to 200). This method prevents unnecessary further training, reducing overfitting and computational complexity.

Validation split	A 20% cross-validation split during the training is used to ensure that adjustments to hyperparameters do not bias performance results.
Batch size	Optimal batch size (32) balances frequency of gradient updates (to capture subtle patterns) and stability of gradient estimates (to avoid erratic swings in weight updates). If the batch size is too big, the risk of missing subtle variations in the data increases, conversely, small batch sizes can inflate training variance.
Model checkpoint	The utility saves best configurations whenever the validation score improves, allowing for the reuse of optimal model states without recalibrating from scratch. This approach is especially important given the random nature of DL models, such as weight initialization, which can produce different outcomes each time the training runs.

3.8.3 Hyperparameter Tuning and Architecture Search

A structured grid-like approach enabled the exploration of varying parameters of the MLP structure, reflecting both manual experimentation and a systematic enumeration of key parameters. We explicitly tested each combination of layer counts and neuron quantities with/without dropout layers to evaluate their effect on predictive performance, rather than relying on automated hyperparameter optimizers. [49] The variations included:

- **Layer Depth** – 0, 1, 2, 3, 5, 8, 10, 15, and 20 hidden layers.
- **Number of Neurons** ranges from 10 to 200 per hidden layer (10, 12, 14, 15, 16, 18, 20, 24, 30, 40, 50, 60, 70, 80, 90, 100, 200).
- **Dropout** is either present with a fixed 0.2 dropout rate or absent.

Values for other hyperparameters are discussed in Table 9.

While conventional ML models allow for deterministic seeding (assigning a pseudo-random generator), DL often introduces additional sources of variability that are harder to control.

The experiment tested 306 unique MLP configurations several times for both cost and duration targets. Results for each unique configuration were averaged to obtain a more stable and representative measure of performance. Many more preliminary exploratory trials, which helped refine the training pipeline and identify pitfalls, were also conducted in the early stages of the research.

Although taking a lot of time, this grid-like search provided clear insights into which architectures performed well.

3.9 Model Evaluation

3.9.1 Metrics Overview

Once all models are trained and fitted, they must be assessed with consistent and meaningful criteria to determine their suitability for EVM forecasting. Metrics provide assessment of model performance, but their significance varies with domain factors, data quality, and project complexity. [11] To form a balanced picture of model effectiveness, this study employs five key metrics:

- **Mean Absolute Percentage Error (MAPE)** – a widely used accuracy metric for measuring how closely forecasts match actual values, often considered primary due to its ability to express errors as a percentage, making it easy to interpret. [8]
- **Normalized Root Mean Squared Error (NRMSE)** – an alternative accuracy measure that highlights the magnitude of errors in the same units as the target variable, providing a clear sense of scale and making it useful for comparing different models. [24]
- **Standard Deviation (SD)** – an indicator of precision in model evaluation, demonstrating how consistently the model's errors cluster around the mean, thus reflecting the reliability of the model's predictions. [13]
- **Mean Lags** – a measure of stability that quantifies the extent to which forecasts shift between consecutive predictions, indicating the consistency of the model over time. [52]

While these metrics complement one another, none of them can be considered perfect. MAPE can skew when actual values approach zero. R^2 can inflate performance for complex models. NRMSE is sensitive to outliers, which can distort the overall error measurement. SD focuses on the spread of errors, potentially missing systematic bias. Mean lags measure the amount of fluctuation over time but do not consider whether a prediction was correct. By relying on multiple metrics, we mitigate limitations presented by individual metric.

Models are also evaluated for timeliness at three different project stages, which are defined by the percentage of work completed (the ratio of AC to BAC), as summarized in Table 10.

Table 10. Breakdown of project phases

Early stage	Mid stage	Late stage
1% – 29%	30% – 69%	70% – 95%

This breakdown recognizes that traditional forecasting methods often perform best in the mid to late stages of the project, because project data tends to stabilize during these stages, whereas early forecasts are more volatile. Managers often depend on early and mid-stage predictions to adjust budgets or refine project scopes. [24] By applying the chosen metrics to each segment separately, the study offers nuanced insights into which approaches excel or struggle at various points in a project's lifecycle.

Assessing performance through multiple lenses reduces the risk of overemphasizing one dimension of model quality.

3.9.2 MAPE

In this study, MAPE is utilized as the primary indicator of forecasting accuracy. It expresses deviations between predicted and actual values in percentage terms, which facilitates easy interpretation. Mathematically, it sums the absolute deviations expressed as percentages of the real data points, then divides by the number of data points, resulting in a single figure, a percentage, that tells us, on average, how "off" a model is in relative terms.

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \widehat{y_i}}{y_i}\right| \tag{14}$$

where:
y_i – actual value
$\widehat{y_i}$ – predicted value

A lower MAPE value implies greater accuracy, since it means the model's predictions stay closer A lower MAPE value implies greater accuracy, since it means the model's predictions stay closer to actual project data (cost or duration). A higher MAPE suggests that the model misjudges project outcomes, which can be costly or disruptive in EVM, where financial and scheduling decisions rely heavily on accurate estimates. [4]

MAPE provides a clear, percentage-based snapshot of where forecasts stand, helping EVM practitioners quickly spot and address potential inaccuracies. Because MAPE is relative, it tends to be intuitive for managers who want a quick read on how closely their forecasts align with actual results. However, MAPE has notable flaws. When actual values are extremely small or zero, MAPE produces disproportionately large error percentages. Additionally, MAPE does not account for whether predictions are higher or lower than actual value, only how large the difference is.

3.9.3 NRMSE

While the MAPE spotlights relative forecasting accuracy, the NRMSE offers a clear measure of prediction errors, normalized to for easier interpretation. By keeping the units consistent (e.g., cost in euros or time in days) and scaling the error relative to actual values, NRMSE helps stakeholders understand how large typical deviations can be, without overemphasizing projects with larger budgets or longer durations.

NRMSE is an adaptation of the RMSE that normalizes the error term, usually by dividing it by the mean of the actual data. A common formula is:

$$NRMSE = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y_i})^2}}{\bar{y}} \tag{15}$$

where,
y_i – actual value
$\widehat{y_i}$ – predicted value
$\bar{y}$ – normalization method, for this research it is corresponding BAC

A lower NRMSE signifies smaller deviations relative to the typical scale of the data, signaling better overall accuracy. In EVM, projects vary widely in size and complexity. A one-million-dollar budget overrun means something very different for a billion-dollar infrastructure project than for a smaller IT implementation. By expressing errors as a ratio, NRMSE ensures that the measurement of error scales appropriately to each project's scope. This aspect is crucial when comparing multiple models or projects, as it prevents large-scale projects from dominating the analysis simply due to their absolute numerical magnitudes.

NRMSE squares larger errors (carries over from RMSE), making it sensitive to outliers. In addition, the choice of normalization method can influence the results, potentially complicating direct comparisons if different methods are used.

3.9.4 SD

In this research, SD is a measure for precision of prediction. SD informs about the consistency of the model performance by taking the square root of variance (average of the squared differences between absolute percentage error and MAPE). [13]

$$SD = \sqrt{\frac{\sum_{i=1}^{n}(APE_i - MAPE)^2}{n}} \tag{16}$$

where,
APE_i – absolute percentage error
$MAPE$ – mean absolute percentage error

SD indicates how spread are the errors around their mean. A higher SD means greater variation among errors, resulting in unpredictable swings in forecasts and instability in project planning. On the other hand, a low SD shows that the model predictions are more consistent (where it predicts the same output for the same input) even if there is an overall skew or bias in one direction. Although SD clarifies how spread out errors are, it does not indicate whether the forecasts are systematically too high or too low (i.e., whether there is an overall bias). A model can consistently overshoot actual costs yet still maintain a small SD if it does so by a nearly uniform margin each time.

3.9.5 Mean Lags

Mean Lags measures stability by looking at how much a forecast changes from one period to the next. For each pair of adjacent time steps, the metric calculates how much the forecasted value has shifted and expresses it relative to the previous forecast. Summing these deviations across the entire forecasting horizon, then dividing by the number of intervals, yields a measure of how "smooth" the model's predictions tend to be. [52]

$$\frac{1}{T}\sum_{i=2}^{T}\frac{|\hat{y}_i - \hat{y}_{i-1}|}{\hat{y}_{i-1}} \tag{17}$$

where,
$\hat{y}_i$ – predicted value

A lower Mean Lags value suggests more stable forecasts, meaning the model's estimates do not swing from one tracking period to another. This consistency can be valuable in EVM. A high Mean Lags indicates frequent or large changes between consecutive forecasts, which may complicate management decisions, even if the model is ultimately accurate over the long run.

Because Mean Lags focuses on temporal fluctuations, it does not consider whether those forecasts are correct in an absolute or relative sense. A model might be very stable but consistently off-target, or it could occasionally produce large errors yet maintain minimal shifts between successive predictions.

Chapter 4. Results

This chapter presents the results of the forecasting models developed in this research. Building on the methodologies detailed in Chapter 3, this chapter examines how both the baseline methods and the proposed MLP architectures perform on two specific forecasting targets: cost at completion and time at completion.

Results, based on metrics established in Section 3.9, are reported both overall and segmented into early, mid, and late stages of work completed. The chapter's organization parallels the two forecasting targets and offers a thorough comparison of baseline (EVM and XGBoost) and MLP methods for each.

Tables 11 and 12 present a summary of the final outcomes for cost and time forecasts. The full tables contain 310 rows and 22 columns, only the results of EVM and XGBoost baseline models, the best shallow MLP, the best deep MLP, and the worst MLP are shown in this chapter. Full details of all 308 model configurations appear in the appendices for readers desiring complete technical reference.

Each MLP configuration is labeled in the format "layer count – neuron count – T/F for dropout", where "layer count" indicates the number of layers, "neuron count" specifies the number of neurons per layer, and "T/F for dropout" denotes whether dropout is applied (True or False). Alongside every model in the condensed table, its ranking among all 308 models indicates how well it fares relative to the entire set of tested configurations.

4.1 Cost Estimate At Completion

Before evaluating model performance, it helps to first understand the overall cost behavior in the dataset. A simple but informative metric is the Mean Deviation of cost at completion (MD_{cost}), comparing each project's actual cost at completion to its budget (BAC). This metric highlights how much, on average, projects deviate from their planned budgets. Formally, MD_cost is defined as: [24]

$$MD_{cost} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - BAC_i|}{BAC_i} \tag{18}$$

where,
y_i – actual value (cost at completion)
BAC – budget of the project

The training set shows an MD_{cost} of 15,05%, whereas the test set yields a noticeably lower 8,47%. This discrepancy implies that the randomly selected test subset, on average, contains projects with smaller cost overruns. From a methodological point of view, such a difference may pose a challenge, as the test data only partially align with the more pronounced overruns seen in training, potentially affecting the generalization of the models' learned patterns.

Despite the difference in cost profiles, the baseline models EVM and XGBoost, both achieve MAPE scores under 10% (6,68% and 5,72%, respectively), as shown in Table 11. However, the best shallow MLP (a simpler architecture) and the best deep MLP (with multiple layers) outperform these baselines, attaining 4,98% and 4,35% MAPE, respectively. They also surpass EVM and XGBoost on virtually every metric at each project stage, with the sole exception of the late-stage predictions.

Even the worst-performing MLP tested maintains a 10,13% MAPE score, slightly above the 10% threshold, which is often considered competitive in management. In the overall cost-forecast ranking, the EVM baseline positioned in the 154th place, whereas XGBoost sits at 53rd, meaning that out of 306 different configurations of MLP, 52 performed better than ML benchmark, and 153 – better than EVM benchmark. MLP configurations clearly demonstrate the added value of using deeper neural network architectures, which can capture complex patterns and interactions in the data, in predicting cost at completion.

Table 11. Final ranking table for cost prediction

	EVM (154)	**XGB (53)**	**MLP shallow 1-12-T (13)**	**MLP deep 5-16-T (1)**	**MLP worst 10-40-F (308)**
MAPE	**6,68%**	**5,72%**	**4,98%**	**4,25%**	**10,13%**
early	13,83%	10,00%	6,65%	6,33%	8,79%
mid	9,95%	7,05%	5,46%	4,88%	11,73%
late	2,31%	3,42%	4,03%	3,26%	10,45%
RMSE	**0,1259**	**0,0946**	**0,0730**	**0,0664**	**0,1438**
early	0,2100	0,1570	0,1019	0,0963	0,1197
mid	0,1480	0,1032	0,0768	0,0700	0,1661
late	0,0364	0,0505	0,0553	0,0488	0,1472
SD	**10,31%**	**6,93%**	**5,23%**	**4,96%**	**8,92%**
early	15,98%	10,46%	8,11%	7,39%	8,92%
mid	9,88%	7,08%	5,21%	4,86%	9,68%
late	2,57%	3,73%	3,35%	3,45%	8,55%
MeanLags	**8,56%**	**6,03%**	**5,76%**	**5,08%**	**9,00%**
early	16,74%	16,90%	7,42%	5,92%	10,25%
mid	7,71%	5,45%	4,82%	3,78%	8,66%
late	8,29%	4,81%	6,33%	5,94%	10,11%

4.2 Time Estimate At Completion

Similar to cost forecasting, it is instructive to begin by examining the overall time patterns in the dataset provides a foundational understanding necessary for analyzing the Mean Deviation of time at completion (MD_{time}). A formal definition reflects that of cost deviation:

$$MD_{cost} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - PD_i|}{PD_i} \tag{19}$$

where,
y_i – actual value (time at completion)
PD – planned duration

MD_{time} for the training set is 17,81%. The test set displays a near-identical value of 17,63%. This implies that, in contrast to cost forecasting, the sampled test projects are broadly consistent with the overall distribution of schedule deviations in the training data.

Despite this relative alignment, Table 12 reveals that baseline and MLP models still vary in their predictive power. The EVM approach forecasts final durations with a 12,41% MAPE, while XGBoost performs slightly better with MAPE of 11,78%. Yet both baselines are surpassed by the best shallow MLP (1–15–T), which attains an 8,22% MAPE, and the best deep MLP (20–10–F) at 8,50%. This performance gap suggests that even basic neural architectures can more effectively capture the complex patterns inherent in real-world scheduling data, compared to baseline models.

Despite being the worst MLP tested, model with 15 layers and 30 neurons in each layer still achieves MAPE of a 17,41%, which is worse than the baselines and best MLPs, but nonetheless remains within the 20% threshold. Examining the stage-specific columns in Table 11 reveals that MLP-based forecasts generally outperform both EVM and XGBoost in early and mid stages, reflecting better adaptability to partial project progress data. However, in some late-stage scenarios, the baselines remain competitive, possibly because cost or time overruns become more predictable as projects near completion.

Collectively, these findings show that, while baseline methods offer a solid reference for time estimation, MLP models, whether shallow or deep, often provide more accurate predictions, particularly when uncertainty is still high as in early-stage forecasting. The near-matching MD_{time} values for train and test sets highlight that all models are learning and evaluating on a relatively uniform set of schedule conditions, increasing confidence that these gains are not solely a product of the test subset's composition.

Table 12. Final ranking table for time prediction

	EVM (162)	XGB (129)	MLP shallow 1-15-T (1)	MLP deep 20-10-F (4)	MLP worst 15-30-T (308)
MAPE	**12,41%**	**11,78%**	**8,22%**	**8,50%**	**17,41%**
early	9,40%	10,03%	8,11%	8,04%	11,11%
mid	12,27%	9,40%	8,60%	9,23%	23,32%
late	12,37%	11,25%	8,29%	8,18%	18,16%
RMSE	**0,2396**	**0,2303**	**0,1802**	**0,1966**	**0,2985**
early	0,1860	0,1872	0,1527	0,1745	0,2187
mid	0,2457	0,2300	0,2219	0,2455	0,3943
late	0,2473	0,2339	0,1832	0,1912	0,2825
SD	**13,71%**	**14,29%**	**11,02%**	**11,44%**	**19,61%**
early	10,96%	12,34%	13,65%	13,02%	16,76%
mid	11,50%	11,80%	11,74%	13,04%	25,57%
late	14,21%	13,81%	10,14%	10,31%	15,47%
MeanLags	**10,55%**	**15,64%**	**10,93%**	**13,62%**	**10,00%**
early	8,32%	14,42	2,84%	9,90%	8,21%
mid	8,84%	11,37%	7,40%	10,90%	7,43%
late	13,02%	15,49%	15,94%	17,81%	12,00%

Overall, the final results indicate that MLPs equipped with 1, 2, or 5 hidden layers, each containing around 14 to 30 neurons, consistently yield the most accurate cost and time forecasts. This finding suggests that project management scenarios do not necessarily require extremely deep or large networks for effective forecasting; moderate architectures suffice to capture the underlying complexities of cost and schedule data. By contrast, models with zero hidden layers, essentially linear approaches, performed noticeably worse, landing in the bottom half of all tested configurations and underscoring the need for some level of non-linearity to address the varied dynamics of real-world projects.

Chapter 5. Discussion

In our research, we found that medium-sized MLPs (2 or 5 hidden layers with 14-30 neurons per layer) are consistently good predictors of both cost and time at completion, hitting an ideal compromise in both cases. By simply providing the network with a single hidden layer, populated with enough neurons, it was able to learn subtle, non-linear trends in the data. They also didn't scale up so much as to overfit the training set or require too much computational power.

Although MLPs improved accuracy for both cost and time estimates consistently, we saw some distinct differences in performance between the two targets. Some MLP configurations were particularly effective in predicting final cost, whilst others appeared to be more adept at handling schedule elements. This is perhaps due to the fact that cost data and schedule data have different forms and patterns.

Another key observation is that the early stages of a project, when little information is available and uncertainty runs high, tend to pose the greatest difficulties for all forecasting methods. Here, MLPs really shine, consistently outperforming baseline models and providing a noticeable edge in proactive risk management. For instance, if a network can flag potential overruns or schedule slips at 20% completion, teams can make far-reaching corrections well before problems spiral. However, by the late stages, when the project has stabilized and much of the major spending or scheduling hurdles are behind it, traditional baselines like EVM or XGBoost kept pace with or even surpassed neural networks. In that context, with fewer unknowns in play, the advantage of capturing non-linear patterns becomes less critical, and simpler methods can hold their own just fine.

Another important factor beyond accuracy is forecast stability, in other words, how smoothly predicted values move from one reporting period to another. This is especially desirable as we saw by looking at Mean Lags and the Standard Deviation of model errors. In many cases, MLPs of moderate depth and neuron count tended to produce relatively smooth updates without the erratic jumps sometimes seen when models overfit to small sets of data. Lower Mean Lags meant that revisions were more gradual between stages, and smaller Standard Deviation of errors showed that the network consistently landed close to its targets from step to step, thus solidifying project managers' faith in the forecasts.

Additionally, we observed that some models even though achieving excellent accuracy metrics, can still introduce higher volatility if a model aggressively recalibrates on fresh inputs. Decision-makers therefore need to strike a balance between raw accuracy and forecast steadiness, recognizing that a perfectly precise model is not always the best fit if it causes the project team to see-saw on important decisions.

One of the most valuable takeaways from this research is the systematic process we developed for configuring MLPs, deciding on the number of layers, selecting how many neurons go into each layer, and fine-tuning core training hyperparameters. While our efforts targeted EVM specifically, that same approach can easily carry over into other fields that also demand a careful balance between accuracy, interpretability, and efficiency. Organizations that focus on resource planning, supply chain optimization, or demand forecasting can follow the same fundamentals, start with a moderate network size, incrementally adjust depth and neuron counts, and fine-tune hyperparameters until they start to build a configuration showing sturdy predictions. By adopting these methods, teams across various industries can make more informed, data-driven decisions, ultimately extending the framework's real-world impact well beyond EVM.

Chapter 6. Limitations and Future Research Directions

The data itself may not represent the entire range of complexity that all project types can have in the real world, as projects can differ significantly in size, domain and data quality. That begs the question of how well does the model generalize to extreme or special and rare scenarios.

Second, while our tuning framework balances depth and width effectively for most data conditions, we did not experiment extensively with novel architectures, which might capture certain dependencies more effectively. Future research could compare these architectures with the moderate MLP approach, especially when faced with more heterogeneous or large-scale project datasets.

Finally, although we explored hyperparameter tuning in a systematic yet manual fashion, advanced optimization techniques (e.g., Bayesian search or population-based training) could further refine network configurations and potentially uncover combinations we did not exhaustively investigate.

Moving forward, there is ample room to extend this work to more diverse project environments, to integrate domain-specific knowledge directly into the model, and to experiment with semi-supervised or unsupervised approaches for cases where labeled EVM data is limited. By continuing to expand the training data, exploring next-generation neural architectures, and refining evaluation methods, future research can further advance our understanding of how deep learning can improve project cost and time forecasting in EVM and well beyond.

Reference List

1. A Guide to the Project Management Body of Knowledge (PMBOK Guide) -- Seventh Edition and The Standard for Project Management. ISBN: 978-1-62825-664-2 Published by: Project Management Institute, Inc

2. Practice standard for earned value management / Project Management Institute. -- 2nd ed. ISBN 978-1-935589-35-8 (pbk. : alk. paper) 1. Project management--Standards. I. Project Management Institute. Published by: Project Management Institute, Inc.

3. Soetjipto, J. W., Ratnaningsih, A., Arifin, S., Adinanda, D. A., & Wicaksono, K. H. (2024). Improving the effectiveness of project scheduling by using Earned Value Management and Artificial Neural Network. Revista Ingeniería De Construcción, 39(2), 161–173. https://doi.org/10.7764/RIC.00112.21

4. Mario Vanhoucke, 2023. "The Illusion of Control," Management for Professionals, Springer, number 978-3-031-31785-9, December. https://doi.org/10.1007/978-3-031-31785-9

5. Uddin, S., Yan, S., & Lu, H. (2024). Machine learning and deep learning in project analytics: methods, applications and research trends. Production Planning & Control, 1–20. https://doi.org/10.1080/09537287.2024.2320790

6. Uddin, S., Ong, S., Lu, H., & Matous, P. (2023). Integrating machine learning and network analytics to model project cost, time and quality performance. Production Planning & Control, 35(12), 1475–1489. https://doi.org/10.1080/09537287.2023.2196256

7. Sarker, I.H. Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. SN COMPUT. SCI. 2, 420 (2021). https://doi.org/10.1007/s42979-021-00815-1

8. Batselier, J. and Vanhoucke, M. (2015) Construction and Evaluation Framework for a Real-Life Project Database. International Journal of Project Management, 33, 697-710. http://dx.doi.org/10.1016/j.ijproman.2014.09.004

9. Brett Thiele, Michael Ryan, Alireza Abbasi, Developing a dataset of real projects for portfolio, program and project control management research, Data in Brief, Volume 34, 2021, 106659, ISSN 2352-3409, https://doi.org/10.1016/j.dib.2020.106659.

10. Shiri, F.M., Perumal, T., Mustapha, N., Mohamed, R. (2024). A comprehensive overview and comparative analysis on deep learning models. Journal on Artificial Intelligence, 6(1), 301–360. https://doi.org/10.32604/jai.2024.054314

11. Ottaviani FM, De Marco A, Narbaev T, Rebuglio M. Improving Project Estimates at Completion through Progress-Based Performance Factors. Buildings. 2024; 14(3):643. https://doi.org/10.3390/buildings14030643

12. Narbaev, T., & Marco, A.D. (2014). Combination of Growth Model and Earned Schedule to Forecast Project Cost at Completion. Journal of Construction Engineering and Management-asce, 140, 04013038. https://doi.org/10.1061/(ASCE)CO.1943-7862.0000783

13. Narbaev, T., & Marco, A.D. (2014). An Earned Schedule-based regression model to improve cost estimate at completion. International Journal of Project Management, 32, 1007-1018. 10.1016/j.ijproman.2013.12.005

14. Pewdum, W., Rujirayanyong, T. and Sooksatra, V. (2009), "Forecasting final budget and duration of highway construction projects", Engineering, Construction and Architectural Management, Vol. 16 No. 6, pp. 544-557. https://doi.org/10.1108/09699980911002566

15. Mario Vanhoucke, ISBN 978-1-4419-1013-4 DOI 10.1007/978-1-4419-1014-1 Springer Dordrecht Heidelberg London New York

16. Batselier, Jordy & Vanhoucke, Mario. (2015). Empirical Evaluation of Earned Value Management Forecasting Accuracy for Time and Cost. Journal of Construction Engineering and Management. 141. 05015010. 10.1061/(ASCE)CO.1943-7862.0001008. http://dx.doi.org/10.1061/(ASCE)CO.1943-7862.0001008

17. Lipke, Walt. PMI Oklahoma City Chapter, Introduction to Earned Schedule

18. Lipke, Walt. (2003). Schedule is different. The Measurable News. Summer. 31-34.

19. Batselier, J., Vanhoucke, M. Project regularity: Development and evaluation of a new project characteristic. J. Syst. Sci. Syst. Eng. 26, 100–120 (2017). https://doi.org/10.1007/s11518-016-5312-6

20. Walt Lipke, Ofer Zwikael, Kym Henderson, Frank Anbari, Prediction of project outcome: The application of statistical methods to earned value management and earned schedule performance indexes, International Journal of Project Management, Volume 27, Issue 4, 2009, Pages 400-407, ISSN 0263-7863, https://doi.org/10.1016/j.ijproman.2008.02.009.

21. T. Kose, T. Bakici and Ö. Hazir, "Completing Projects on Time and Budget: A Study on the Analysis of Project Monitoring Practices Using Real Data," in IEEE Transactions on Engineering Management, vol. 71, pp. 4051-4062, 2024, http://dx.doi.org/10.1109/TEM.2022.3227428

22. Tolga İnan, Timur Narbaev, Öncü Hazir, A Machine Learning Study to Enhance Project Cost Forecasting, IFAC-PapersOnLine, Volume 55, Issue 10, 2022, Pages 3286-3291, ISSN 2405-8963, https://doi.org/10.1016/j.ifacol.2022.10.127

23. Karadimos, P., Anthopoulos, L. A taxonomy of machine learning techniques for construction cost estimation. Innov. Infrastruct. Solut. 9, 420 (2024). https://doi.org/10.1007/s41062-024-01705-0

24. Narbaev, T., Hazir, Ö., Khamitova, B., & Talgat, S. (2023). A machine learning study to improve the reliability of project cost estimates. International Journal of Production Research, 62(12), 4372–4388. https://doi.org/10.1080/00207543.2023.2262051

25. Kyriazos T, Poga M. Application of Machine Learning Models in Social Sciences: Managing Nonlinear Relationships. Encyclopedia. 2024; 4(4):1790-1805. https://doi.org/10.3390/encyclopedia4040118

26. Alom MZ, Taha TM, Yakopcic C, Westberg S, Sidike P, Nasrin MS, Hasan M, Van Essen BC, Awwal AAS, Asari VK. A State-of-the-Art Survey on Deep Learning Theory and Architectures. Electronics. 2019; 8(3):292. https://doi.org/10.3390/electronics8030292

27. Jason Brownlee, What is Deep Learning?, Machine Learning Mastery, Available from https://machinelearningmastery.com/what-is-deep-learning/, accessed December, 2024

28. Jason Brownlee, Difference Between Backpropagation and Stochastic Gradient Descent, Machine Learning Mastery, Available from https://machinelearningmastery.com/difference-between-backpropagation-and-stochastic-gradient-descent/, accessed December, 2024

29. Jason Brownlee, When to Use MLP, CNN, and RNN Neural Networks, Available from https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/, accessed December, 2024

30. McKinney, W., & others. (2010). Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51–56). https://doi.org/10.25080/Majora-92bf1922-00a

31. Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., … Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585, 357–362. https://doi.org/10.1038/s41586-020-2649-2

32. Van Rossum, G. (2020). The Python Library Reference, release 3.8.2. Python Software Foundation. http://web.mit.edu/course/18/18.417/doc/pydocs/lib.pdf

33. Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17, 261–272. https://doi.org/10.1038/s41592-019-0686-2

34. Hagberg, A., Swart, P., & S Chult, D. (2008). Exploring network structure, dynamics, and function using NetworkX. http://dx.doi.org/10.25080/TCWV9851

35. Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90–95. http://dx.doi.org/10.1109/MCSE.2007.55

36. Inc., P. T. (2015). Collaborative data science. Montreal, QC: Plotly Technologies Inc. Retrieved from https://plot.ly

37. Waskom, M., Botvinnik, Olga, O'Kane, Drew, Hobson, Paul, Lukauskas, Saulius, Gemperline, David C, … Qalieh, Adel. (2017). mwaskom/seaborn: v0.8.1 (September 2017). Zenodo. https://doi.org/10.5281/zenodo.883859

38. Baak, M., Koopman, R., Snoek, H., & Klous, S. (2020). A new correlation coefficient between categorical, ordinal and interval variables with Pearson characteristics. Computational Statistics & Data Analysis, 152, 107043. https://doi.org/10.1016/j.csda.2020.107043

39. Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In KDD. https://doi.org/10.1145/3292500.3330701

40. Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785–794). New York, NY, USA: ACM. https://doi.org/10.1145/2939672.2939785

41. Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B., Grisel, O., … others. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825–2830. https://doi.org/10.48550/arXiv.1201.0490

42. Chollet, F., & others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras

43. Walt Lipke, Earned Schedule Excel calculators, Available at https://www.earnedschedule.com/Calculator.shtml, Accessed December 2024

44. Paulo André de Andrade, Annelies Martens, Mario Vanhoucke, Using real project schedule data to compare earned schedule and earned duration management project time forecasting capabilities, Automation in Construction, Volume 99, 2019, Pages 68-78, ISSN 0926-5805, https://doi.org/10.1016/j.autcon.2018.11.030.

45. Jason Brownlee, How to Get Reproducible Results with Keras, Available from https://machinelearningmastery.com/reproducible-results-neural-networks-keras/, accessed December, 2024

46. Jason Brownlee, Why One-Hot Encode Data in Machine Learning?, Available from https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/, accessed December, 2024

47. Jason Brownlee, 3 Ways to Encode Categorical Variables for Deep Learning, Available from https://machinelearningmastery.com/how-to-prepare-categorical-data-for-deep-learning-in-python/, accessed December, 2024

48. Barrientos-Orellana, A., Ballesteros-Pérez, P., Mora-Melia, D., González-Cruz, M.C. and Vanhoucke, M. (2022), "Stability and accuracy of deterministic project duration forecasting methods in earned value management", Engineering, Construction and Architectural Management, Vol. 29 No. 3, pp. 1449-1469. https://doi.org/10.1108/ECAM-12-2020-1045

49. Jason Brownlee, How To Get Baseline Results And Why They Matter, Available from https://machinelearningmastery.com/how-to-get-baseline-results-and-why-they-matter/, accessed December, 2024

50. Jason Brownlee, How to Grid Search Hyperparameters for Deep Learning Models in Python with Keras, Available from https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/, accessed December, 2024

51. Jason Brownlee, Dropout Regularization in Deep Learning Models with Keras, Available from https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/, accessed December, 2024

52. Wauters, M., & Vanhoucke, M. (2015). Study of the stability of earned value management forecasting. JOURNAL OF CONSTRUCTION ENGINEERING AND MANAGEMENT, 141(4). https://doi.org/10.1061/(ASCE)CO.1943-7862.0000947

53. Chicco, D., Warrens, M. J., & Jurman, G. (2021). The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation. PeerJ. Computer science, 7, e623. https://doi.org/10.7717/peerj-cs.623