On Forecasting Project Activity Durations with Neural Networks[⋆]

Peter Zachares[1], Vahan Hovhannisyan[1], Carlos Ledezma[1], Joao Gante[2], and Alan Mosca[1]

[1] nPlan, `<first name>@nplan.io`
[2] For contributions made while employed at nPlan, `joaofranciscocardosogante@gmail.com`

Abstract. Accurately forecasting project end dates is an incredibly valuable and equally challenging task. In recent years it has gained added attention from the machine learning community. However, state of the art methods both in academia and in industry still rely on expert opinions and Monte-Carlo simulations. In this paper, we formulate the problem of activity duration forecasting as a classification task using a domain specific binning strategy. Our experiments on a data set of real construction projects suggest that our proposed method offers several orders of magnitude improvement over more traditional approaches where activity duration forecasting is treated as a regression task. Our results suggest that posing the forecasting problem as a classification task with carefully designed classes is crucial for high quality forecasts both at an activity and a project levels.

Keywords: Forecasting · Project Management · Neural Network applications

1 Introduction

Many economic endeavours require undertaking complex projects which can only be completed over long time spans [10]. In the planning phase, projects are allocated budgets which rely on the completion of the project in a similar manner to which it was planned. If a project's completion is delayed, this can lead to ballooning costs, which can have a major negative impact on the project stakeholders including failure to complete the project [9]. To mitigate the risk of delay, stakeholders of the project require an accurate time estimate of project duration. With this knowledge, stakeholders can assess the risk of delay and even execute mitigation actions to reduce this risk.

While it is impossible to estimate the duration of a complex project exactly [6], it is possible to determine which durations are more likely than others, i.e. in almost all cases a project will not take one hundred times longer to complete than planned. Consequently, it is more useful to stakeholders to forecast the duration of a project rather than to predict it.

[⋆] Supported by nPlan

Each complex project is unique, but can share many similarities with past projects. Using a data-driven modelling approach and details from past projects, it is possible to accurately forecast a project's duration providing stakeholders with useful information for mitigating the risk of delay.

Complex projects involve the completion of numerous tasks. Each task in the project has a state with respect to the current time which could be *not started*, *in progress* or *completed*. In a project, the state of a task can depend not only on the current time, but also on the state of many other tasks within the project i.e. the facade structure must be completed before facade window installation can begin. It is possible to model these state dependencies using a directed acyclic graph (DAG) [5], where nodes represent activities and links represent dependency constraints.

If provided with a duration forecast for each task in the project and an estimated start date for the project, it is possible to estimate the end date, and thus duration, of the entire project using an uncertainty propagation method such as Monte Carlo sampling [32]. This simplifies the problem of forecasting a project's duration to forecasting the durations of tasks in the project. In addition, past projects are more likely to contain similar tasks to a future project than they are to mimic the future project exactly. Consequently, past project data can provide more useful information for forecasting at a task level than a project level.

Taking the approach described above to project duration forecasting, the quality of the forecasts at a task-level will determine the quality of the forecast of project duration. Neural networks have achieved state-of-the-art performance in a number of forecasting tasks (also known as uncertainty estimation) both by combining results from ensembles of models [23] [11] [28] and applying post-hoc calibration [15].

In this paper we perform experiments to determine which training objectives and inductive biases applied to neural networks yield the most accurate task-level forecasts on project data. We perform experiments using a data set of construction projects. Our results suggest that a neural network that outputs a histogram distribution over task duration trained with a classification objective and using a domain specific binning strategy to define the classes leads to the most accurate forecasts.

The contributions of our work are threefold;

- To the best of our knowledge this is the first machine learning model used to forecast project activity durations.
- A novel problem formulation which poses task duration forecasting as a classification problem.
- A novel binning strategy incorporating domain knowledge about how humans complete tasks which leads to higher quality forecasts when used as an inductive bias to train models on project data.

Notation . We follow standard notation practices using bold lower case letters for vectors and bold upper case letters for matrices. For scalars we use both lower and upper case non-bold letters. Subscripts on vectors and matrices refer to its elements. We use superscripts to indicate indices for vectors and matrices.

2 Related Work

Forecasting is an age old problem. Early on, humans used empirical observations to forecast the weather, but now we apply complex mathematical models to forecast whether multi-billion dollar mega projects will finish on time and on budget [33]. Currently, forecasting is a very broad field of research with applications in environment sciences [21], economics, supply chain management, project controls and many other industries [31], [13]. For a comprehensive review of forecasting theory and practice we refer the reader to [29]. In this paper our focus is on forecasting time durations of activities in project schedules.

While there has been some recent work on using machine learning algorithms to forecast project durations using handcrafted features [7], [24], [25] (also see reference therein), traditionally task duration forecasting has been implemented using a blanket distribution for all tasks [8] or with added expert opinion [16], [26], [27] for individual activities. The latter, more specialised solution is, of course, more accurate, however, it is based on human inputs and inherently suffers from various biases [2].

In a recent work [17] proposes a machine learning algorithm to learn activity types in construction projects using a manually labelled database of activities. This work is closest to ours in the literature, however, it does not address the much more challenging problem of forecasting activity durations.

Our proposed method is based on the observation that task duration forecasts are inherently heavy-tailed [16] and that Gaussian distributions are not suitable to represent such distributions [35]. Hence, we pose task duration forecasting as a multi-class classification problem to train neural networks to output a histogram distribution describing the uncertainty in tasks' durations [3].

Since the majority of all activities in a project complete on time, even if the project suffers from large delays [3], splitting the activity duration range into uniform intervals for a histogram distribution creates a large class imbalance. We mitigate this well-known problem [19], [15] by proposing `equibins` - a novel binning strategy which calculates a more balanced class distribution than using uniform intervals. We also use domain knowledge about how humans accomplish tasks to further improve our models.

3 Problem Formulation

Each task in a project has an associated feature vector $\mathbf{x}$ and duration label y. We address the problem of learning the parameters $\boldsymbol{\theta}$ of a neural network

[3] 2.977 million of 5.693 million (or 52.3%) activities in our database have completed exactly on time.

$f_{\boldsymbol{\theta}} : \mathbf{x} \to P(\mathcal{Y})$ that maps a task's feature vector to a probability distribution over its duration with the objective;

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x})\bigg|_{\mathcal{Y}=y} \tag{1}$$

The objective is to maximise the likelihood of the neural network's output distribution evaluated at y. We assume that each task also has a planned duration y_{plan} and define a new quantity, completion ratio as:

$$c = \frac{y}{y_{plan}} \tag{2}$$

When actualised, each task will have a completion ratio whose magnitude will indicate the accuracy of the planner in estimating the duration of a task. We assume that all tasks will be completed within some completion ratio range $c \in [c_{min}, c_{max}),\ c_{min} > 0,\ c_{max} > c_{min}$ and separate the interval $[c_{min}, c_{max})$ into n intervals of $([c_{min}, c_1), [c_1, c_2), ..., [c_{n-2}, c_{n-1}), [c_{n-1}, c_{max}),\ i > j \to c_i > c_j)$.

We assume the associated distribution of the random variable $\mathcal{Y}$ is a histogram distribution composed of the following intervals:

$$y_{plan} \times \{[c_{min}, c_1), [c_1, c_2), ..., [c_{n-2}, c_{n-1}), [c_{n-1}, c_{max})\} \tag{3}$$

We assume that within each bin the distribution's probability density is uniform. Hence, each task's duration y will fall within one of the histogram's bins scaled by y_{plan}:

$$y \in [c_{n-1} y_{plan}, c_{max} y_{plan}) \tag{4}$$

Using this formulation, it is possible to generate a one-hot vector $\mathbf{y}$ of size n with zeros in all dimensions except the dimension corresponding to the index of the interval the duration falls in, where the vector has value 1. With these assumptions we can optimise our original objective by reposing forecasting task duration as a classification problem where $f_{\boldsymbol{\theta}}$ maps to a multinomial distribution (a vector of non-negative numbers which sum to one with n dimensions) over completion ratio ($f_{\boldsymbol{\theta}} : \mathbf{x} \to P(\mathcal{C})$) and we optimise the equivalent objective;

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x})^T \mathbf{y} \tag{5}$$

4 Method

We assume access to a data set of related tasks from past projects which is split into a training set and a validation set. Each split contains two matrices; a feature matrix $\mathbf{X} \in \mathbb{R}^{k \times m}$ and a label matrix $\mathbf{Y} \in \mathbb{R}^{m \times n}$, where k is the number of data points, m is the number of features and n is the number of classes.

We assume the mapping from task features to the task's distribution over completion ratio can be expressed by a multi-layer perceptron [30] with a softmax operation as its final layer. For completeness, we define the softmax operation below:

$$\boldsymbol{\sigma}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{l=1}^{m} e^{z_l}}, \tag{6}$$

where $\mathbf{z}$ is a vector of size n. We train our model using a variant of stochastic gradient descent [22] to minimise the negative loglikelihood (also known as cross-entropy loss) of the true class of each task in our training set i.e. if $\mathbf{X}_{batch}$ is a batch of feature vectors of k' tasks from our training set and $\mathbf{Y}_{batch}$ are the tasks' corresponding label vectors. Then we minimise the loss:

$$Loss = \sum_{i=1}^{k'} -log(f_{\boldsymbol{\theta}}(\mathbf{x}_i)^T \mathbf{y}_i) \tag{7}$$

The majority of all tasks in a project complete on time, even if the project suffers from large delays. In addition, there are certain values of completion ratio that are very common i.e. 1.0, 1.5, 2.0 in our data set (see Figure 1 for the data distribution). This indicates that when stakeholders work on a project, if a task cannot be finished on time, then they set a new deadline which is some multiple of original task's planned duration. Consequently, tasks tend to be delivered late and early in a predictable manner. This means, splitting a task's completion ratio support $[c_{min}, c_{max})$ into uniform intervals can create a large class imbalance in the data set. If intervals are all of uniform size, then an interval can contain many commonly appearing completion ratio values in the data set and consequently contain a much larger proportion of the data set than other intervals.

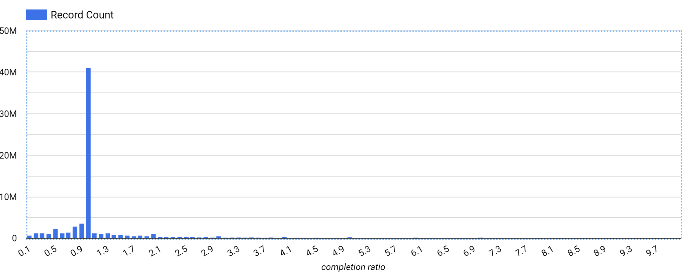

Fig. 1. Completion ratios (training labels) of all construction schedule activities in our database: over half of them have completion ratio 1, aka completed on time. There are also noticeable spikes at 0.5, 2, 3, etc points.

To mitigate this issue, we propose binning strategy called `equibins`, which helps create a more balanced class distribution. `equibins` is an iterative algorithm. Before starting, the algorithm stores a count of all completion ratio values in the training and validation set which occur more than once. A hyperparameter of the algorithm is how many bins n_{bins} the algorithm should split the support into. The algorithm starts out with all labels in training and validation sets union $c \in \mathbf{C}_{train+val}$. At each iteration the algorithm calculates how many data points would be within intervals of uniform size i.e. $n_{uniform} = \lfloor \frac{|\mathbf{C}_{train+val}|}{n_{bin}} \rfloor$. If it finds a completion ratio value c which occurs more than $n_{uniform}$ times, then the algorithm creates a bin centred at c with bounds $[c - \epsilon, c + \epsilon)$ where ϵ is another hyperparameter of this binning strategy. Then the algorithm removes the points within the defined interval from the set of completion values considered, sets $n_{bin} = n_{bin} - 1$ and repeats the steps described above until there are no completion ratio values which occur more than $n_{uniform}$ times. Then the algorithm splits the remaining intervals in the support into bins by finding the boundaries which separate the remaining data into n_{bin} equally sized quantiles. Note that n_{bin} could be less than the originally specified value depending on the number of iterations of the algorithm required to remove common completion ratio values from the set of considered values.

5 Experiments

Our experiments are designed to answer two questions:

1. Does posing task duration forecasting as a classification problem yield higher quality forecasts?

2. Does our proposed binning strategy `equibins` improve the quality of forecasts outputted by a task duration model trained as a classifier in comparison to a naive binning strategy?

We perform our experiments on real world project data from our proprietary database of over $400,000$ construction schedules from a wide range of sectors. The labels for each task correspond to the one-hot vector encodings described in section 3. We filter out any tasks from our data set which have a completion ratio less than 0.1 or greater than 10, viewing them as noisy or mislabelled. After data cleaning and preprocessing we get a data set of over $70,000,000$ activities to train and test our models. We split our data set into a training, validation and test set using split ratio $16 : 4 : 5$. Each task's feature vector is a combination of a 128 dimensional embedding vector of a textual description of the task from a pre-trained tiny-BERT [20] language model and a set of numerical features including the tasks planned duration y_{plan} and numerical features which describe the date at which the task is planned to start.

For our proposed model, we used a 5-layer fully-connected network. Each layer, except the last layer is composed of a linear layer, a batch normalisation layer [18], a Relu layer [1] and a dropout layer [34]. Figure 2 presents the model architecture. The last layer uses a softmax operation as its activation function and is not followed by a drop-out layer. For our experiments we used a dropout rate of 0.3 and the internal layers of the model had 3968 hidden units. The model was trained using ADAM [22] with a learning rate of $2.0e - 4$. The hyperparameters of the model architecture and training were determined using a

Bayesian hyperparameter search [4]. We trained our model for 64 epochs with an early stopping condition: model training terminated if the model's performance on the validation set did not improve for five consecutive epochs. For evaluation, we used the parameters for each model which resulted in the lowest validation loss. For our `equibins` binning strategy, we separated the support of the histogram into 53 bins and used an ϵ value of $1e-3$.

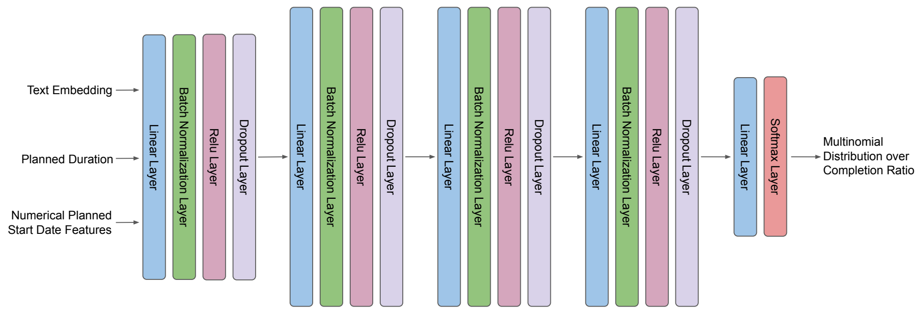
Text Embedding
Planned Duration
Numerical Planned Start Date Features
Linear Layer
Batch Normalization Layer
Relu Layer
Dropout Layer
Linear Layer
Softmax Layer
Multinomial Distribution over Completion Ratio

Fig. 2. Representative multi-layer perceptron used in our experiments.

We compare our proposed model to three baselines. All baselines share the same model architecture up to their last layer and were trained using the training procedure described above. The first baseline poses task duration forecasting as a regression task where the model outputs a Gaussian distribution. The model output is two parameters; a mean and log standard deviation. We further process the log standard deviation by performing a softplus operation on it [37] to ensure stability during training. This baseline outputs a uni-modal distribution.

If our data is inherently multi-modal, the baseline could perform poorly due to its inability to describe a multi-modal distribution. Consequently, our second baseline also poses task duration forecasting as a regression task where the model outputs a Gaussian mixture model with 3 modes. The model also outputs a log standard deviation for each mode (as opposed to standard deviation) which is processed by applying softplus operation to it. The model outputs 3 terms for the weighting terms of the Gaussian mixture model which are processed by performing a softmax operation on them to ensure they sum to one. Both regression baselines are trained by minimising the negative log-likelihood of each task's actualised completion ratio with respect to the distribution they output. The third baseline poses task duration forecasting as a classification task, but uses uniformly sized bin intervals as opposed to the `equibins` strategy we propose.

[4] https://cloud.google.com/blog/products/ai-machine-learning/hyperparameter-tuning-cloud-machine-learning-engine-using-bayesian-optimization

For our purposes, classification metrics such as F1-score are not very informative, since we do not intend to use the predicted class in any downstream task. Instead, the entire distribution outputted by the classifier is used as a forecast for the duration of a task. Consequently, we report metrics commonly used to measure forecast quality.

The first metric is the probability mass within $1e-2$ of a task's actualised completion ratio averaged over the entire test set. This metric measures local distribution quality and approximates the likelihood of sampling the correct completion ratio for a task. The second metric is the continuous ranked probability score (CRPS) [12] of the distribution outputted by our model averaged over the entire test set. This metric assesses the quality of the output distribution over its entire support, which roughly measures the overall quality of the learned forecast. The third metric we report is the expected calibration error (ECE) [14], [4], which is another rough measure of the quality of the distributions outputted by our model.

6 Results

Table 1 records the performance of our proposed model and the three baselines described above. The results show clearly that in terms of likelihood there is an order of magnitude increase in performance ($\sim 0.001 \rightarrow \sim 0.01$) when using models trained using a classification objective versus a model trained using a regression objective to forecast task duration.

A possible reason for the improved performance of models trained using classification objectives as opposed to regression objectives is that continuous distributions like Gaussian and Gaussian mixture models use distance metrics to calculate likelihood. These density functions contain an implicit locality assumption where points close to each other have similar likelihoods. In addition, Gaussian likelihoods decay exponentially as they move away from the mean of the distribution. If the underlying distribution a model is trying to learn does not meet this locality assumption or exponential decay assumption, then a Gaussian density functions will struggle to approximate it. Using a classification objective, the model learns a histogram distribution which makes fewer assumptions about the underlying distribution. Specifically, it makes a strong locality assumption within bins, but no locality assumption across bins. In our experiments, histogram distributions trained using a classification objective (which makes no locality assumption across bins) yield better uncertainty estimates from the trained model.

In addition, there is another order of magnitude increase ($\sim 0.01 \rightarrow \sim 0.1$) in performance in terms of likelihood when using the `equibins` binning strategy as an inductive bias in the model as opposed to uniformly sized bins. As explained in Section 4, `equibins` leads to a more balanced class distribution. A more balanced class distribution encourages sensitivity in our models as they see more variation in their targets during training and consequently are more likely to identify statistical relationships in their training set than models trained on a highly unbalanced class distribution.

In terms of CPRS on the test set, the combination of training the model using a classification objective and using an equibins strategy yielded the same improvement in CRPS over the three baselines ($\sim 0.060 \rightarrow \sim 0.044$). Finally in terms of ECE, there was not much of a difference between the models trained using classification objectives (~ 1.7 versus ~ 1.9), but there was a large jump in performance when comparing models trained using regression objectives as opposed to classification objectives (~ 7.0 versus ~ 1.8). To calculate the ECE for the two regression baselines, we first approximated them as histogram distributions within the completion ratio support $[0.1, 10)$.

In our experiments, the performance of the models in terms of likelihood and CRPS suggest that training models using a classification objective and incorporating an `equibins` binning strategy improves the performance of neural networks on task duration forecasting. On the other hand, the performance of models in terms of ECE suggest that only training models using a classification objective is important.

Model	Likelihood	CRPS	ECE
Unimodal Gaussian Regressor	0.0058	0.0600	7.1043
Gaussian Mixture Model with 3 modes Regressor	0.0062	0.0582	7.0310
Uniform bin size Classifier	0.0368	0.0602	1.7416
Equibins Classifier	0.3680	0.0443	1.8973

Table 1. Performance metrics on the test set of models trained to perform task duration forecasting. Higher likelihood and higher CRPS mean better forecasting power; lower ECE means better forecasting power.

Better activity level forecasts should result in better project end date forecasts. To demonstrate this, we ran Monte-Carlo simulations on various projects for which we have both initially planned and fully actualised schedules. We report the difference in performance between the two best performing models: our proposed `equibins` classifier and the uniform bin size classifier baseline on the project end-date forecasting task. We report the probability density function calculated by performing a Monte Carlo simulation on a project using the distributions outputted by these models.

We found a very consistent behaviour across all tested project, and we report visual representative results of project end date forecasting in Figure 3. Visually the results suggest that using the `equibins` classifier results in probability density functions for project end-date forecasts which place a much higher likelihood on the project's actual end-date with more mass around this date than when using a uniform bin size classifier.

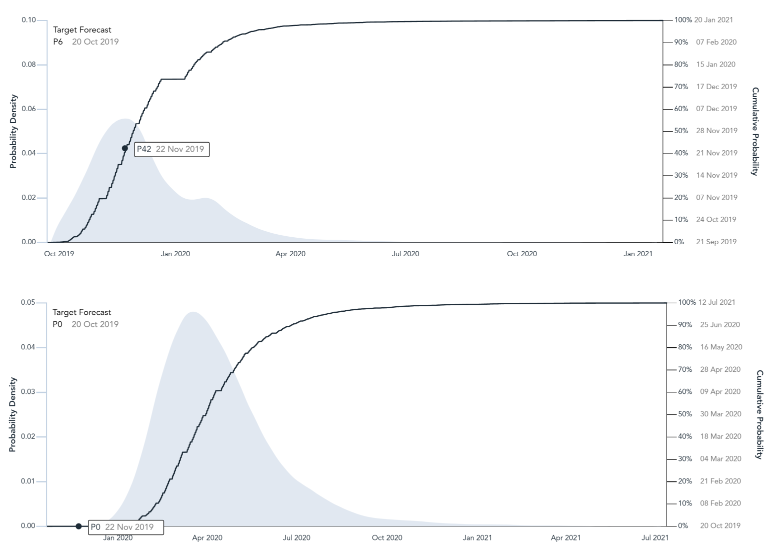

Fig. 3. Comparison of project level forecasts after running Monte-Carlo simulations using samples from a classification model with `equibin` classes (above) versus a regression Gaussian mixture model (below). The same true end date for the project is marked on each graph.

7 Discussion and Conclusions

Accurately forecasting project end dates is a valuable, yet challenging task, traditionally solved by relying on heuristics provided by experts to forecast activity durations and Monte-Carlo simulations. In recent years, the problem of forecasting task durations has gained a growing amount of attention from the machine learning community. However, there is still no consensus on what is best practice when it comes to forecasting activity durations using neural networks. In this paper, we experiment with different problem formulations and inductive biases to determine which are important when training neural networks to forecast activity durations.

Experiments on our data set of real world construction projects suggest the performance benefits of our proposed method for forecasting activity duration, as well as forecasting project end dates. We show that posing the forecasting problem as a classification task with carefully designed classes is crucial for high quality results.

The next natural step in our research is to replace the simple feed forward neural networks used here with ones that can take advantage of the underlying graph structure in project schedules. In recent years graph neural networks have been deployed in a wide range of applications from molecular biology to social sciences [36], however graph neural networks have not yet been effectively applied to project duration forecasting problems. Another interesting direction is searching for methods to train models end-to-end, meaning directly to forecast project end dates, possibly removing the need to perform Monte-Carlo sampling as part of the model when forecasting project end dates.

References

1. Agarap, Abien Fred. "Deep learning using rectified linear units (relu)." arXiv preprint arXiv:1803.08375 (2018).
2. Bhandari, Siddharth, and Keith R. Molenaar. "Using debiasing strategies to manage cognitive biases in construction risk management: Recommendations for practice and future research." Practice Periodical on Structural Design and Construction 25.4 (2020): 04020033.
3. Bishop, Christopher M. Pattern recognition and machine learning. springer, 2006.
4. Culakova, Natalia, et al. "How to Calibrate your Neural Network Classifier: Getting True Probabilities from a Classification Model." Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2020.
5. D. G. Malcolm, J. H. Roseboom, C. E. Clark, W. Fazar, (1959) Application of a Technique for Research and Development Program Evaluation. Operations Research 7(5):646-669.
6. Dubois, Didier, Helene Fargier, and Philippe Fortemps. "Fuzzy scheduling: Modelling flexible constraints vs. coping with incomplete knowledge." European Journal of Operational Research 147.2 (2003): 231-252.
7. Egwim, Christian Nnaemeka, et al. "Applied artificial intelligence for predicting construction projects delay." Machine Learning with Applications 6 (2021): 100166.
8. Fazar, Willard. "Program evaluation and review technique." The American Statistician 13.2 (1959): 10.
9. Fiori, Christine, and Molly Kovaka. "Defining megaprojects: Learning from construction at the edge of experience." Construction Research Congress 2005: Broadening Perspectives. 2005.
10. Flyvbjerg, Bent, Nils Bruzelius, and Werner Rothengatter. Megaprojects and risk: An anatomy of ambition. Cambridge university press, 2003.
11. Gal, Yarin, and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." international conference on machine learning. PMLR, 2016.
12. Gneiting, Tilmann, and Adrian E. Raftery. "Strictly proper scoring rules, prediction, and estimation." Journal of the American statistical Association 102.477 (2007): 359-378.
13. Gneiting, Tilmann, and Matthias Katzfuss. "Probabilistic forecasting." Annual Review of Statistics and Its Application 1 (2014): 125-151.
14. Guo, Chuan, et al. "On calibration of modern neural networks." International Conference on Machine Learning. PMLR, 2017.
15. Guo, Xinjian, et al. "On the class imbalance problem." 2008 Fourth international conference on natural computation. Vol. 4. IEEE, 2008.
16. Hahn, Eugene David. "Mixture densities for project management activity times: A robust approach to PERT." European Journal of operational research 188.2 (2008): 450-459.
17. Hong, Ying, et al. "Determining Construction Method Patterns to Automate and Optimise Scheduling–A Graph-based Approach." European Conference on Computing in Construction. doi: https://doi. org/10.17863/CAM. Vol. 68385. 2021.

18. Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." International conference on machine learning. PMLR, 2015.
19. Japkowicz, Nathalie and Stephen, Shaju. 'The Class Imbalance Problem: A Systematic Study'. 1 Jan. 2002 : 429 – 449.
20. Jiao, Xiaoqi, et al. "Tinybert: Distilling bert for natural language understanding." arXiv preprint arXiv:1909.10351 (2019).
21. Jolliffe, Ian T., and David B. Stephenson, eds. Forecast verification: a practitioner's guide in atmospheric science. John Wiley & Sons, 2012.
22. Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).
23. Lakshminarayanan, Balaji, Alexander Pritzel, and Charles Blundell. "Simple and scalable predictive uncertainty estimation using deep ensembles." Advances in neural information processing systems 30 (2017).
24. Mahdi, Mohammed Najah, et al. "Software project management using machine learning technique—A Review." Applied Sciences 11.11 (2021): 5183.
25. Mahmoodzadeh, Arsalan, et al. "Forecasting tunnel geology, construction time and costs using machine learning methods." Neural Computing and Applications 33.1 (2021): 321-348.
26. Mahmoodzadeh, Arsalan, et al. "Predicting construction time and cost of tunnels using Markov chain model considering opinions of experts." Tunnelling and Underground Space Technology 116 (2021): 104109.
27. Maravas, Alexander, and John-Paris Pantouvakis. "Project cash flow analysis in the presence of uncertainty in activity duration and cost." International journal of project management 30.3 (2012): 374-384.
28. Mosca, Alan, and George D. Magoulas. "Boosted residual networks." International Conference on Engineering Applications of Neural Networks. Springer, Cham, 2017.
29. Petropoulos, Fotios, et al. "Forecasting: theory and practice." International Journal of Forecasting (2022).
30. Popescu, Marius-Constantin, et al. "Multilayer perceptron and neural networks." WSEAS Transactions on Circuits and Systems 8.7 (2009): 579-588.
31. Raftery, Adrian E. "Use and communication of probabilistic forecasts." Statistical Analysis and Data Mining: The ASA Data Science Journal 9.6 (2016): 397-410.
32. Richard M. Van Slyke, (1963) Letter to the Editor—Monte Carlo Methods and the PERT Problem. Operations Research 11(5):839-860.
33. Sanderson, Joe. "Risk, uncertainty and governance in megaprojects: A critical discussion of alternative explanations." International journal of project management 30.4 (2012): 432-443.
34. Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." The journal of machine learning research 15.1 (2014): 1929-1958.
35. Taleb, Nassim Nicholas. The black swan: The impact of the highly improbable. Vol. 2. Random house, 2007.
36. Wu, Zonghan, et al. "A comprehensive survey on graph neural networks." IEEE transactions on neural networks and learning systems 32.1 (2020): 4-24.
37. Zheng, Hao, et al. "Improving deep neural networks using softplus units." 2015 International Joint Conference on Neural Networks (IJCNN). IEEE, 2015.