Guidelines for Applied Machine Learning in Construction Industry—A Case of Profit Margins Estimation

Muhammad Bilal, Lukumon O. Oyedele

Abstract

The progress in the field of Machine Learning (ML) has enabled the automation of tasks that were considered impossible to program until recently. These advancements today have incited firms to seek intelligent solutions as part of their enterprise software stack. Even governments across the globe are motivating firms through policies to tape into ML arena as it promises opportunities for growth, productivity and efficiency. In reflex, many firms embark on ML without knowing what it entails. The outcomes so far are not as expected because the ML, as hyped by tech firms, is not the silver bullet. However, whatever ML offers, firms urge to capitalise it for their competitive advantage. Applying ML to real-life construction industry problems goes beyond just prototyping predictive models. It entails intensive activities which, in addition to training robust ML models, provides a comprehensive framework for answering questions asked by construction folks when intelligent solutions are getting deployed at their premises to substitute or facilitate their decision-making tasks. Existing ML guidelines used in the IT industry are vastly restricted to training ML models. This paper presents guidelines for Applied Machine Learning (AML) in the construction industry from training to operationalising models, which are drawn from our experience of working with construction folks to deliver Construction Simulation Tool (CST). The unique aspect of these guidelines lies not only in providing a novel framework for training models but also answering critical questions related to model confidence, trust, interpretability, bias, feature importance and model extrapolation capabilities. Generally, ML models are presumed black boxes; hence argued that nobody knows what a model learns and how it generates predictions. Even very few ML folks barely know approaches to answer questions asked by the end users. Without explaining the competence of ML, the broader adoption of intelligent solutions in the construction industry cannot be attained. This paper proposed a detailed process for AML to develop intelligent solutions in the construction industry. Most discussions in the study are elaborated in the context of profit margin estimation for new projects.

Keywords: Applied Machine Learning, Profit Margin Forecasting, Construction Simulation Tool, Interpretable Machine Learning, Predictive Modelling

1. Introduction

Machine learning (ML) has been a renaissance in the IT industry. Today sectors like manufacturing are utilising ML algorithms from design to delivery of great consumer products (Manyika, 2017). This success has lured other sectors of the economy like construction industry into the adoption of intelligence algorithms in their enterprise software (Gomber et al., 2018). The latest progress in ML algorithms has enabled the automation of those non-trivial tasks which, a decade ago, were deemed impossible (Luong et al., 2015; Toshev & Szegedy, 2014; Bau et al., 2019). However, ML is not a silver bullet. These algorithms are far fetched to achieve human-level performance on many tasks (Mendelson, 2019). Merely, embarking on ML projects without the knowledge about the strengths and limitations of ML algorithms often lead to failures (Ng, 2018). Even ML on simple tasks becomes difficult without a structured roadmap; hence leads to unfavourable outcomes as we witnessed in predictive policing (Shapiro, 2017). Seemingly perfect ML models fail during the production that leads to investment loss, increased scepticism and lack of future support. Unless a systematic methodology is exercised for training and operationalising models, ML projects will tend to fail frequently. In true essence, ML has enormous potential to transform the construction industry by automating many perennial problems related to project planning and control (Bilal et al., 2016).

Since majority of ML solutions tend to either automate or facilitate the decision-making process, it is imperative for application users to trust the capabilities of ML models (Doshi-Velez & Kim, 2017). It requires diverse knowledge of the domain, the data and ML algorithms to train models that are reliable, transparent and trusted. So the scope of Applied Machine Learning (AML) goes beyond merely prototyping ML models. Currently, ML engineers undertake most modelling decisions based on their domain knowledge expertise and intuitions, which are occasionally proven to be wrong and biased. AML emphasises to engaging the industry experts at early stages of the process where the domain is progressively learned from the data through ML models and corroborated by the industry experts (Rudin & Vahn, 2014; Mullainathan & Spiess, 2017). This early engagement of users into the AML process will provide the domain experts enough time to understand the strengths and weaknesses of algorithms, whereas will also enable the ML engineers to learn the domain as the process progresses. AML is, therefore, a more involved task where ML models are developed iteratively by gradually incorporating domain knowledge from industry experts. Such ML models not only have great predictive accuracy and generalizability but also key to understand non-trivial relationships between large volumes of data (Bose & Mahapatra, 2001).

The profit margin estimation for construction projects is an important decision that business development teams in construction firms undertake during early design stages. However, accurately forecasting profit margins is difficult since many factors influence this choice (Leon et al., 2017). At the moment, project teams decide profit margins based on intuitions or uniform rates, which are unreliable methods (Bagies & Fortune, 2006). Margins often erode as projects progress. The projects started with planned margins end up with entirely different margins at completion (Banaitiene & Banaitis, 2012). The profitability performance of firms can be significantly improved if margins are rightly set based on project-specific details; thereby reducing the overall bankruptcy ratio of construction firms (Rui et al., 2017; Akintoye & Skitmore, 1991). Most firms posses data about historical projects under the compliance for the Building Information Modelling (`BIM`). This data reside in `relational`, `CSV` or `XLS` data sources. Most importantly, this data contains every detail of when and how margins change in projects. ML algorithms can be utilised to understand the relationship between project attributes and profit margins. Such models have the potential to improve the accuracy of profit margin estimation significantly.

This paper aims to propose the AML process for guiding the development of useful ML models for the construction industry. The process is gleaned from leading the development of the Construction Simulation Tool (CST), which is project analytics software, utilising data to its fullest for facilitating the decision making across numerous activities involved in the construction project lifecycle. The process provides a roadmap for executing ML projects in the industry. Most discussions are focused for construction IT folks. In this article, the AML process is broken down into tasks, and the description of each task along with technical details, are provided. The aim is to ensure robust sanity checks are exercised before rolling out ML models into the production. These guidelines are general in the sense that these can be used to develop ML models for the majority of construction industry tasks. However, discussions are contextualised with a real-life example of profit margin estimation to solidify the concepts and showcase how various design choices are undertaken and how domain knowledge is progressively learned as the process progresses.

This paper is organised as follows: Section 2 describes literature review on the need for profit margin estimation in construction projects. Section 3 and Section 4 present the process and guidelines for the AML with practical demonstrations from profit margin estimation. Lastly, Section 5 provides concluding remarks, discusses lessons learnt and future directions of the work.

2. Literature Review

The construction industry is a highly competitive landscape and continually fluctuating industry. Firms strive to deliver projects with great profitability performance. This zeal puts these firms in constant pursuit of maximising profits on awarded contracts, whereas lowering margins on new bids to get more work. Getting the profit margins right is paramount to making many decisions at various project lifecycle stages. Several

Table 1: List of ML Models used in Profitability Performance Estimation

Sr.#	Tool Name	Approach or Methodology Employed	Total Projects	Key Limitations	Literature Source(s)
1	Project Profit Forecasting System	This system uses a hybrid approach where fuzzy clustering and genetic algorithm are used for classifying projects, and support vector regressor (SVR) is employed for training the ML model for profit margin forecasting. System outperformed other ML algorithms like generalised regression neural networks (GRNN), radial basis function neural network (RBFNN) and back propagation neural network (BPNN).	Twenty-five (25) projects	Model is hard to generalise as high-level project attributes of fewer projects are used for training.	(Chang et al., 2013)
2	SVM Forecasting System	Support vector machine (SVM)-based algorithm is used to develop a model for forecasting profit margins. R DTREG package is used for training the model. R^2 and Mean Absolute Percentage Error (MAPE) are used for model evaluation.	Twenty-six (26) projects	Model is specific to geographic location and hard to generalise.	(Petruseva et al., 2016)
3	InMES	ANN is used to develop profit forecasting model where de-compositional KT-1 algorithm to generate rules for the neural network. RMSE is used to evaluate model performance.	Thirty (35) projects	Model contains one layer which clearly can't capture non-linearity between project attributes and profit margins.	(Li & Love, 1999) (Li et al., 1999)
4	DBID	DBID uses ANN to forecast the profit margins by harnessing ANN with GA. Monte Carlo simulation is used for sensitivity analysis to assess the winning probability concerning variability in profit margins.	Twenty (20) projects	Fewer data used in training and evaluation method is not robust.	(Moselhi et al., 1993a)
5	Bid Expert	Bid Expert integrates FaRM with cost estimation model for predicting margins. Tool used past data of firm and of its competitors.	Not available	It is hard to use this sytem as firms can rarely access competitors' data.	(Moselhi et al., 1993b)

external, organisational and project related factors affect profit margins (Rui et al., 2017). The contractors generally set higher margins on courtesy bids or if a project involves more risks. The relationship between profitability performance and project-specific factors is not clearly understood. Estimators adjust margins based on their intuitions or use a uniform rate to guide their decisions, which are sometimes wrong or misleading. Therefore, a project started with a planned margin, eventually finishing with entirely another margin. The adverse implications of poorly designed projects are enormous as one such project can ruin the cash flow of the firm, and can lead to its bankruptcy (Kim et al., 2008). Devising a systematic approach to explore the variability in profit margins by project attributes is indispensable. This knowledge is crucial to enable estimators to understand the profitability performance across an array of projects and estimate margins correctly for new projects.

The estimators are hard to convince to use ML for profitability forecasting outrightly as they have their reasons. While they are keen to see ML is useful for their tasks, they usually seek clarifications about the strengths and weaknesses of ML algorithms before they trust or let these algorithms to influence their decisions. ML interpretability for adoption of intelligent systems in construction industry is crucial. In profitability forecasting, estimators hold many assumptions about how project specific attributes drive profit margins. Some estimators believe that projects in specific regions or from particular clients shall always be estimated at higher rates (Rui et al., 2017). We need to confirm these assumption using ML. The estimators mostly claim their assumptions stem from their experience. However, not all assumptions are right or wrong unless proven otherwise. Besides, most estimators follow heuristics such as uniform rates for overhead costs and profit margins (Akintoye & Skitmore, 1991). One limitation of these approaches is their inability to reflect variations that can be induced by project-related risks or contingencies. Profit margins shall be derived to include the negative as well as the positive impact of project specific attributes (Kim et al., 2008). Such provisions call for advanced digital technologies to support estimators to testify such assumptions quickly. ML algorithms have a huge role to play to enable comprehension of large amounts of data as well as to train highly reliable predictive models. If ML is not devised with interpretability in mind, it will be hard to operationalise such algorithms regardless of whatever ML algorithm we employ or whatever accuracy we claim. The industry has a record for the moderate intake of emerging technologies.

Several researchers have developed ML models to forecast the profitability performance of construction projects. A major drawback of these models is their only inclusion for high-level project attributes which significantly affect their predictive accuracy for new projects. These algorithms fall into two categories. The first category uses ML models for classifying projects into Bid/No-Bid classes (Dulaimi & Shan, 2002; Lin

& Chen, 2004; Enshassi et al., 2010; Shash, 1993; Wanous et al., 2003, 2000; Christodoulou, 2010). The second class harnesses ML to regress profit margin for new projects (Chang et al., 2013; Petruseva et al., 2016; Li et al., 1999; Li & Love, 1999; Moselhi et al., 1993a; Dikmen & Birgonul, 2004; Moselhi et al., 1993b). Table 1 shows popular ML works used in the industry in the domain of profitability performance. It is obvious that these approaches use rudimentary ML approaches and used fewer data for training. This paper demonstrates the development of an ML model for profit margin estimation. While we presented a significant performance increase through better training algorithms and more data, the main focus of this work is set to highlight key steps for developing robust ML models. These steps are presented as a unified Applied Machine Learning (AML) process. A main caveat is that we do not expect ML folks to have prior domain knowledge as it can bias their judgment, and lead to wrong modelling decisions. They acquire knowledge progressively as the process advances.

3. Applied Machine Learning Process

The guidelines for applied machine learning (AML) are described in this section. These guidelines are arranged into a structured process, consisting of several tasks. Fig. 1 shows an overview of the proposed AML process. Various tasks in the process are gleaned from our experience of developing numerous ML models in the CST. We chose those tasks which are often found useful in producing highly robust ML models. In most cases, the final model obtained by following these steps was a better fit for production deployment in the CST.

The AML process consists of six well-defined tasks (see Fig. 1). After defining the ML problem and establishing the performance metrics, access to the data is obtained, which are then integrated and curated for robust ML training. Next, several standard data pre-processing activities are recommended to get the data into the right format. After that, the baseline model is trained using various design strategies, which, in our example of profit margin estimation using the random forest, are subsampling and hyperparameter tuning. Once we attain a reasonable level of accuracy for the baseline model, we then move onto the interpretable machine learning step. This step is crucial as it enables ML engineers not only to understand the domain progressively but also better explore the competencies of the baseline model. Several revisions are made to the data and predictive models during this step. Critical insights related to Trust, Bias, Transparency, Confidence and Interpretability of ML are investigated. The early engagement of domain experts enables them to trust these models by knowing their capabilities and limitations. Finally, other ML algorithms are also used to train models for the given problem to see if a better alternative can be found. Once the best learning algorithm is decided, we train the final model using all the data and using the optimal hyperparameters. The resulting model is then deployed in the enterprise solution for real-life scoring. These AML tasks are further explained in the subsequent sections.

3.1. ML Task Definition & Data Selection

According to Mitchell (1997), ML programs learn from experience (`E`) with respect to task (`T`) and performance measure (`P`) if its performance (`P`) improves at task in (`T`) with experience (`E`). In the above formalism, terms (`T`), (`P`) and (`E`) are the first things to define in the AML process.

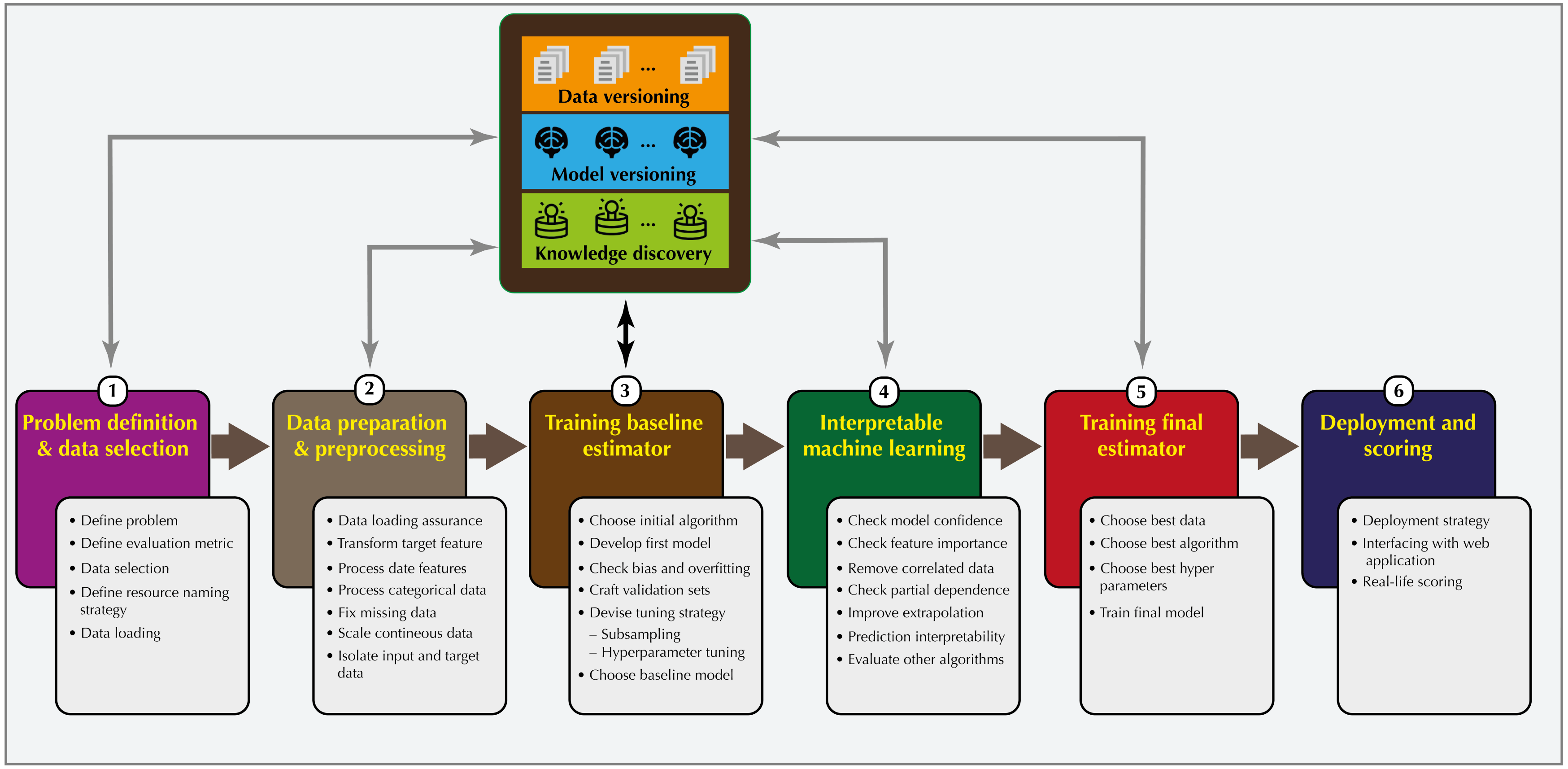


Figure 1: An Overview of Applied Machine Learning Process

3.1.1. Defining Task & Evaluation Metrics

The first step and arguably the most important aspect of ML is to clearly define the ML task and its intended use. The results of ML model would be worthless if the task is wrongly defined. It involves describing the problem and listing all assumptions. More importantly, we need to explain intended use of ML, including its implications for productivity and efficiency for target users. In the case of CST, we set the goal of ML task to predict the profit margin for new opportunity based on project attributes (see Table 2). As discussed briefly, profit margin estimation is tricky task as many factors influence its choice. Most estimators determine margins based on their intuition or uniform rate which are not reliable approaches. The intended use is to employ this model to predict profit margin during early estimation stage. We assume that detailed design has been completed and we have detailed project specific information for all attributes to make such prediction.

The next step in the process is to specify performance metric for evaluating ML models. A performance metric reveals how effectively a model captures the relationship between input and the target features in the given data. It provides a uniform basis for comparing different ML algorithms. The type of ML task influences the type of evaluation metric employed. Supervised ML tasks fall into two categories (Friedman et al., 2001). A regression task involves the prediction of a continuous value, whereas a classification task aims to predict a label. We first determine the ML task type and then specify the performance metric. In the CST case, our task is multivariate regression problem where ML will predict profit margins, i.e. a continuous value. The most common metric for regression tasks is Root Mean Squared Error (`RSME`), which computes the difference between the predicted value and actual value. In the case of CST, we used Root Mean Squared Log Error (`(RMSLE)`, which is the log difference between actual and predicted profit margins. The reason behind this choice is the fact that we don't want the performance metric to penalize huge differences in the predicted and true values when these values are huge numbers. The profit of $10,000$ from $100,000$ worth project is same as the profit of $100,000$ from $1,000,000$ worth project. In such scenarios like profit estimation, only the percentage differences matter, so instead of `RMSE` we selected `RMSLE` to observe percentile differences. Eq. 1 shows the formula for `RMSLE`.

$$RMSLE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\Big(log(\hat{y}+1) - log(y+1)\Big)^2} \tag{1}$$

We also used R^2 to keep track of the accuracy of the ML model. R^2 explains the ratio of the accuracy of the predictive model (`SSR`) to that of baseline model `SST`. Eq. 2 shows the formula for R^2.

$$R^2 = 1 - \left(\frac{SSR}{SST}\right) \tag{2}$$

One useful trick we found handy is to define a function `print_score()` that takes the model as input and prints a tuple containing `Training Error`, `Validation Error`, `Training Accuracy`, and `Validation Accuracy`. This function is generally invoked each time we revise our model to see its performance in response to a change in the data or hyperparameters. The usability of `print_score()` greatly depends on naming conventions used for the datasets. The function will not be useful if dataset names change frequently.

3.1.2. Data Selection, Integration, Naming & Loading

The last part of our formalism is the experience `E` that means data for training and evaluating ML models. The availability of enough quality data is crucial for training reliable ML models (Jordan & Mitchell, 2015). With recent advances in Big Data, firms have started to store all sorts of data. In the construction sector, firms store data to comply with Building Information Modelling (`BIM`), but these data reside in different data sources. One of the challenging tasks in AML is to identify reliable sources of data. Once sources are identified, we integrated them into a `Common Data Environment (CDE)` where technologies like `Hadoop` are vastly

Table 2: List of Project Attributes for Profit Margin Estimation

Sr.#	Attribute Name	Attribute Description
1	Opportunity ID	Unique code attributed to the opportunity (8 characters long)
2	Opportunity title	Unique code attributed to the opportunity (8 characters long)
3	Client	The customer whom the opportunity is from
4	Market	Workstream category to which the opportunity belongs to
5	Region	Part of the country opportunity belongs to as defined by BB
6	kV	Voltage (s) along the route of opportunity
7	km	Total length of the route of the opportunity
8	Project type	The type of opportunity based on BB defined categories
9	Start date	Proposed construction start date of the opportunity
10	End date	Proposed construction finish date of the opportunity
11	Duration (days)	Proposed total duration of the opportunity
12	Outage (s)	Total duration of all proposed outage (s) on the opportunity
13	Key milestones	Key milestones on the project
14	Duration (days)	Duration of key milestones on the project
15	Termination	Points of termination at each end of the route
16	Contract type	Client's chosen contract
17	ITT date	Date/Proposed date which we were invited to tender for the opportunity
18	Tender submitted	Date/Proposed date that the tender is submitted to the cl
19	Contract awarded	Date/proposed date that the client offers the opportunity
20	Mobilisation date	Date/proposed date for initial mobilisation on to project
21	BB role	The role of BB in this partnership
22	**Scope & complexity**	**Scope of the project**
23	Scope	Scope of the project
24	% of work	% of BB work within scope
25	Ground type	Ground conditions for the route area
26	Route length	Length per ground type
27	HDDs in river	HDDs in river
28	Total length in river (m)	Total length in river (m)
29	HDDs in rail	HDDs in rail
30	Total length in rail (m)	Total length in rail (m)
31	HDDs in road	HDDs in road
32	Total length in road (m)	Total length in road (m)
33	HDDs through utilities	HDDs through utilities
34	Total length in utilities (m)	Total length in utilities (m)
35	HDDs in other	HDDs in other
36	Total length in other (m)	Total length in other (m)
37	Joint bays	No of points where two cable ends are jointed
38	**Material cost**	**Material attributes that influence project profitability**
39	No of circuits	Number of circuits
40	No of phases	Number of phases
41	Conductor	The type and size of conductor within the power cable
42	Cable length (m)	Total length of the power cable throughout the route
43	Duct size	Power cable duct size
44	Duct length	Power cable duct length
45	Fibre cable length	Fibre cable length
46	Duct size	Fibre cable duct size
47	Duct length	Fibre cable duct length
48	Pilot cable	Pilot cable length
49	Duct size	Pilot cable duct size
50	Duct length	Pilot cable duct length
51	Straight joints	Total amount of joint accessory between the same cables
52	Transition joints	Total amount of joint accessory between two different ca
53	Tiles	No of tiles for the entire route of the opportunity
54	JB tiles	No of JB tiles for the entire route of the opportunity
55	Tapes	No of tapes for the entire route of the opportunity
56	Cable markers	No of cable markers for the entire route of the opportunity
57	Link boxes	No of link boxes on the entire route of the opportunity
58	**Supplier**	**The attributes below are associated with materials supply managem**
59	Power cables	Desired supplier (s) for power cable (please choose client if
60	Joints	Desired supplier (s) for cable joints (please choose client if free
61	Terminations	Desired supplier (s) for the terminations used at each end of
62	Backfill materials	Desired supplier (s) for the backfill materials (please
63	Subcontractors	The attributes below are associated with subcontractor work
64	Scaffold	Desired subcontractor (s) for scaffold works
65	Labour	Desired subcontractor (s) for agency labour
66	HDD	Desired subcontractor (s) for Horizontal directional drilling
67	Testing	Desired subcontractor (s) to carry out route testing
68	Access	Desired subcontractor (s) for access works
69	**Resource**	**The attributes below are associated with the key resources on a project**
70	Bid manager	Proposed bid manager for the opportunity
71	CM Proposed	commercial manager for the opportunity
72	Estimator	Proposed estimator for the tender stage
73	PM	Proposed project manager for the delivery stage
74	QS	Proposed quantity surveyor for the delivery of the opportunity
75	Planner	Proposed planner for the for the delivery of the opportunity
76	PDM	Proposed design manager for the opportunity
77	PE	Proposed project engineer for the delivery of the opportunity
78	SA	Proposed safety advisor for the delivery of the opportunity
79	**Others**	**Other attributes of a project**
80	Variations/Compensation events	Any disruption to the agreed workflow which is not the contractor's fault and has cost implication
81	Risks	Potential mishaps during the delivery of the project
82	Opportunity/innovation	An instance where an estimated cost is partially reduced or totally avoided

used. In the case of CST, several sources were identified including `Oracle Financials`, `Business Objects`, `Google Earth`, `Keyhole Markup Langauge (KML)`, `Oracle CRM`, and many non-standardised `Excel` and `CSV` files. We used `SQL` for data curation and integration. The data is first stored into staging tables in `Hadoop` and then transported to the relational database. Data elements for the same project across different data sources were joined using unique project identifier which stayed immutable in all systems. The data contained $437,000$ construction projects, completed in the last 20 years. We used all data for training except for the last six months that are used to create validation set.

Once data is available, the next step is to load it into the analytical toolbox of your choice. During the process, data is split into various datasets like one for training models and one or more for validation to see how model performs on unseen data. These datasets shall be prepared in the same way. One caveat is to follow resource-naming strategy and stick to it throughout the AML process. We will see that several copies of data are created during the process, and if we do not adhere to naming conventions, it leads to strange modelling errors. In this study, we used `Anaconda` as analytical toolbox, where `Jupyter` is used for writing code scripts for data processing and model development. We used Python programming language and employed `read_csv()` method from `pandas` library to load our data from `Hadoop` into `X_raw` data frame. We split `X_raw` into two data frames `X` and `y`. The data frame `X` holds the input features, whereas `y` holds the target attribute. These data frames are further split into training and validation sets. We used terms `X_train`, `X_valid`, `y_train` and `y_valid` for these data frames. Test data is generally advised to set aside at the beginning in `X_test` data frame until the final model gets ready, and use it for final evaluation. The files on `Hadoop` are named as `Train.csv`, `Test.csv`, and `Valid.csv`. After preprocessing is performed on data, it is good practice to store data into python `feather` format for later use.

3.2. ***Data Preparation and Pre-Processing***

Data plays the most crucial role in ML projects and would rarely be in the format required by ML algorithms. Oftentimes, data shall be cleansed, filtered, normalised, sampled, and transformed in various ways before algorithms can leverage it for learning. ML engineers spend most of their time and efforts in data preparation (Hulten, 2018). It is always recommended to glance through the data and check for both its format (i.e. structure and data types) and contents (Witten et al., 2016; Friedman et al., 2001). Data may not be what you expect even if you have read its descriptions. Many transformations are normally recommended before ML models are trained. Following sections explain essential tasks that are performed on most datasets.

3.2.1. Data Consistency Verification

Several issues may arise when data is moved between the systems. It is essential to ensure that you are working on a consistent copy of the data. Therefore, after data is loaded into data frames, we shall check its consistency. It is always suggested to explore data and at least reconcile the total number of columns and rows. This verification shall not be confused with Exploratory Data Analysis (EDA), which is suggested in most of the ML textbooks (Cox, 2017). Experts in ML recently noticed that we shall avoid EDA before predictive modelling as it can bias our judgement. ML engineers make decisions like adding or discarding attributes based on EDA are sometimes found wrong & misleading. They shall progressively acquire domain knowledge during the AML process and make critical modelling decisions informed by their implication on model performance, not judgments that are build through the EDA and intuition. In the case of CST, we loaded data into `X_raw` data frame and checked it's structure, columns, data types and number of rows using python commands `display(X_raw)`, `X_raw.columns()`, `X_raw.dtypes()` and `len(X_raw)`. We compared the output of these commands with statistics from actual data stored on `Hadoop`. There were no inconsistencies found in our data frame. In addition, we also checked the top and bottomed 10 rows to ensure `character encodings` of the columns have no issues.

3.2.2. Target Attribute Transformation

The first data transformation begins with modifying the target attribute (`y`) based on the chosen performance metrics. As explained in the earlier section, performance metric plays a vital role in producing reliable ML models. We can transform (`y`) on the fly through data augmentation that is computationally intensive due to repeated processing in each epoch or use data preprocessing once before we start training our models. A good practice is to avoid compute-intensive operations to save time and resources. Therefore, the `y` attribute shall be transformed once at the source than repeatedly inside the cost function during training, which is a computationally-intensive and infeasible approach.

In our CST example, we intend to use `RMSLE` as the performance metric, which is the log difference of actual and predicted profits on projects. We performed this computation at once on the `y_train` and `y_valid` data frames. We used the `log()` function from `numpy` library for this transformation. This way we speed up various tasks of the AML process through intelligent design choices.

3.2.3. Feature Extraction

Feature engineering is one of the most crucial tasks in the AML process. It involves a series of transformations to our data to enable the algorithm to quickly learn underlying insights. Our approach to feature engineering is slightly different. We barely perform feature engineering at this stage of the AML process except for the `date` and `geospatial` attributes. This section explains commonly used transformations for `date` attributes for enabling the model to understand temporal dependencies in the data. We usually transform `date` attributes into several derived attributes, including `day`, `week`, `month`, `year`, `quarter`, `month_first_day`, `month_last_day`, `quarter`, `weekday`, `weekend`, `quarter_start_day`, `quarter_end_day`, `week_number`, and `elapsed_time`. Similar transformations are also desirable for geospatial attributes to derive attributes like distances, etc. Once attributes are derived and appended, original attributes are dropped from the data frames.

In our CST example, we created derived attributes from several `date` attributes such as `project_start_date` and `project_end_date`. For these two attributes, 26 derived attributes are appended to `X_raw` data frame. We omitted time attributes like `hour`, `minute` and `second` as our problem does not require granularity up to that level. In addition, there were no `geospatial` attributes in our dataset, so no `geospatial` transformations are performed. We created several python functions such as `expand_date_attr()` and `expand_gis_attr()` to automatically extract derived columns for input `date` and `geospatial` attributes.

3.2.4. Data Types Conversion

The next step in the AML process is data types transformation. Data types describe the domain of attributes. In ML, there are generally two main types of data. 1) `Categorical data` that stores values corresponding to discrete categories for a columns. For example, `Workstream` attributes is categorical and holds values like `Cabling`, `Substation`, or `Overhead lines`. 2) `Numerical attributes` store numerical values, which can be integers or real numbers like `costs`, `margin` and `net sales value (NSV)`. When we load our data, most analytical toolboxes interpret numerals properly. However, these tools often misinterpret categorical attributes into character data. ML engineers shall manually transform character data back to categorical data for the model to harness it for learning underlying patterns and insights.

In our CST example, we noticed several attributes like `Region` and `Workstream` were stored as character objects in `X_raw` data frame. We created a python function `to_cats()` for automatically converting character attributes into `pandas` library `category` objects. This function uses python utility function `as_type('category')` for data types conversion. For ordinal categories, we can specify the order of values inside the categorical attributes if it is necessary. By default, `pandas` maps textual descriptions with numerical categories in `Jupyter` but it treats these columns as numerals internally. Optionally, these attributes can be replaced with numerical codes outrightly. However, it will adversely affect ML engineers result-interpretation abilities since they would be required to remember all categories behind these codes.

3.2.5. Fix Missing Data

A common problem in almost every dataset is the issue of missing data. There are numerous reasons—ranging from a human error to incorrect sensor readings to a software bug that causes a value to be missing.

In some cases, missing data is absolutely no issue since it will get populated at a later stage in the business process. A simple solution to deal with missing values is to delete all rows containing nulls. However, it will drastically shrink the dataset size if the majority of rows constitute nulls. Sometimes, the presence of nulls can be a pattern of interest and can provide additional insights to ML algorithms for learning the relationship between input and the target attributes. Therefore, missing values must be dealt with great caution. Some analytical libraries like `pandas` in python automatically create an additional category of -1 for null entries if we change an attribute data type to `category`. However, ML engineers need to fix missing values in all continuous attributes. The most vastly used approach for dealing with nulls is `imputation`, which replaces nulls with an estimate. A common imputation technique is `Mean imputation` where `mean` of an attribute substitutes all nulls in that column.

In our CST example, we used `mean imputation` to populate nulls in all continuous attributes. In addition, we also included additional columns to the data frame `x_raw` for all columns containing nulls to retain insights that can be revealed from the pattern of nulls in those attributes. These new attributes are named by suffixing column name with text `_na`. For example, the nulls in `route_length` attribute are substituted with mean of `route_length`, and a new column `route_lenght_na` is appended to the data frame, storing the digit 1 for all rows with nulls otherwise 0. We shall also preserve these mean values as we will need these in future for fixing nulls in the validation as well as test sets. Otherwise, the validation and test sets will not be fit for evaluating the ML models.

3.2.6. Scale Transformation of Continuous Data

Another useful data preparation technique is scaling or normalisation of all continuous attributes in the dataset. Most ML engineers feel confused with scaling and normalisation as both transform our data. The key difference is that scaling modifies the range of data to make it fit a given scale like $0 - 100$ or $0 - 10$. Normalisation, on the other hand, shifts data distribution such that it can be described as the normal distribution. Scaling is a good choice for techniques like `support vector machines (SVM)` or `k-nearest neighbours (KNN)`. Normalisation works best with `t-tests`, `ANOVA`, `linear regression`, `linear discriminant analysis (LDA)` and `Gaussian naive Bayes`.

In our CST example, we did not apply any scaling due to our choice of the algorithm, i.e. `random forest`. The tree-based ML algorithms have no requirements as such of the data to be normally distributed or normalised. However, we applied `Box-Cox` transformation for all continuous attributes when we fitted the deep neural network for profit margin estimation.

3.2.7. Encoding the Categorical Attributes

The categorical attributes contain non-numerical values, which sometimes are not suitable for most of the ML algorithms. We can transform categorical attributes in two ways. Firstly, `integer-encoding` where unique numbers are assigned to categories as we did during the data type conversion task. This approach suits scenarios where categories contain inherent order like regions. `Integer-encoding` does not work for categorical attributes with no ordinal relationship. We are likely to get unexpected results from our analysis if we treat nominal attributes as ordinal. Secondly, `one-hot encoding` is another alternative for categorical transformation, where one binary attribute is added to the dataset for each category in the column and a value of 1 is recorded for respective category in the row otherwise 0. These newly created attributes are sometimes called dummy attribute.

In our CST example, we introduced the idea of a threshold `max_num_cats` to guide the algorithm to employ `integer encoding` if categories are above the `max_num_cats` otherwise use `one-hot encoded` for categorical encoding. After some experimentations, we found `max_num_cats` of five (5) a good threshold for choosing between categorical encoding options for this dataset. The value of `max_num_cats` shall be stored for future use to transform the validation and test datasets. All basic data preparation steps end here. We usually split our data at this stage into an input matrix `X` and a target vector `y`. All data shall be in the numerical format and stored in feather format using python `to_feather()` function. Feather is

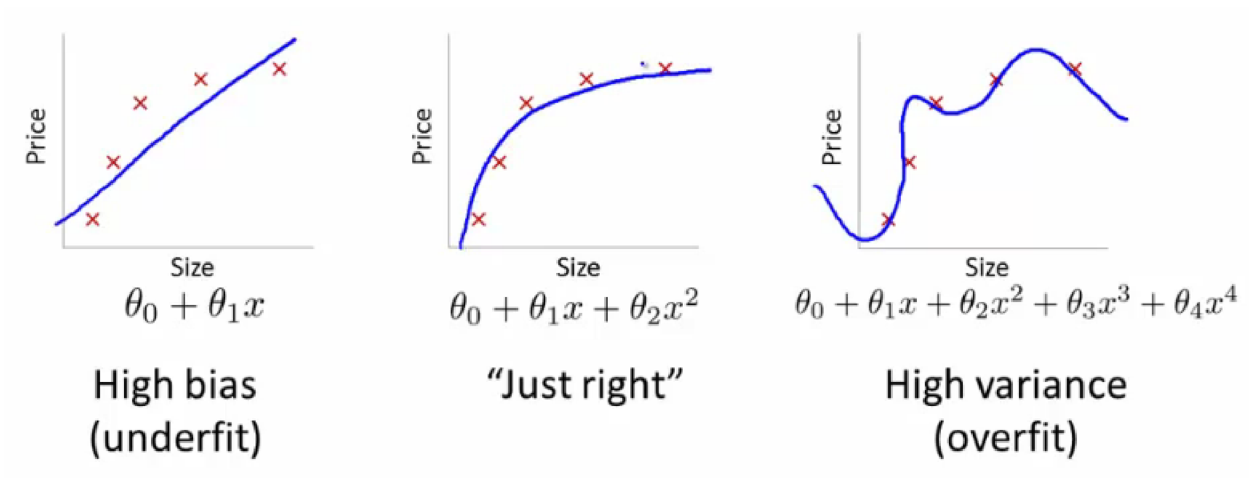


Figure 2: Underfitting and overfitting of estimator

fast on-disk format for data frames and is used by ML engineers for data exchange and interoperability of datasets between analytical tools.

*3.3. **Training Baseline Estimator***

So far, all categories in the data are replaced with numerical codes, nulls are imputed, and our data is split into the input and target attributes. The next step of the AML process is to train the baseline estimator. This section provides guidelines for creating baseline estimators. Though it involves extensive experimentation to find one with reasonable accuracy. The baseline estimator sets the stage for the next AML task, where we use ML to guide engineers in making critical design decisions in an informed way. More importantly, feature engineering where baseline estimator informs ML engineers what to include or drop from the feature list, based on the model's accuracy. The following sections explain these steps in much detail.

3.3.1. Choose First ML Algorithm

Our first step to developing the baseline estimator is to decide the kind of ML algorithm we would employ. A vast majority of ML tasks can be modelled using two main classes of algorithms. Firstly, the `Decision Tree Ensembles (DTE)`, including Random Forests or Gradient Boosting Machines. DTEs are suitable for tasks involving structured data representing different facts like a relational table. Secondly, the `Deep Neural Networks (DNN)` trained with Stochastic Gradient Descent (SGD) which work great for tasks involving unstructured data like audio, vision, and natural language.

In our CST example, we chose the random forest algorithm for training our baseline estimator. A random forest is a bunch of decision trees, containing a bunch of decisions to classify data into several clusters. We used popular python library `sklearn` for this purpose and imported the `RandomForestRegressor` class to start training our model. We began with all data and random forest with ten decision trees. The results were remarkable. The accuracy (R^2) for the model was 96.72%. However, we do not know whether the estimator is good, or it is merely overfitting our data which is a key challenge for ML engineers. This issue can be elaborated using Fig. 2 where three models are depicted. The first model fits the data using a straight line (linear & biased case), and the third model fits through every data point, making a curved line (variance). These two models are not considered reliable as they are either unable to learn the relationship due to underfitting or instead memorised the data entirely due to overfitting. Such models perform poorly on the unseen real-life data and are not desirable. The second model though looks better in terms of learning abilities and generalisability, so we intend to get models with similar characteristics.

3.3.2. Crafting Great Validation Sets

The issue of bias and variance of a model leads us to the issue of creating great validation sets. There is no way to know the learning abilities of an estimator without a good validation set. We shall not confuse validation sets with test sets. Validation sets are generally carved out of the training set whereas test sets are suggested to set aside at the beginning of the AML process and shall not be used during training phase until we are ready to test our final estimator. This strategy is to ensure two-staged evaluation of estimators and has been found phenomenal to achieve several benefits. More importantly, it enables ML engineers to understand the capabilities of models in terms of overfitting and underfitting. The role of validation sets is also crucial to find out the most optimal hyperparameters of an ML algorithm for the given dataset. We usually merge training and validation sets before training our final estimator. So, carving out a robust validation set is a key stage in the AML process. A simple caveat is to choose those data that resemble closely to the real-life scenarios if model will be deployed in the enterprise solutions. We usually create several validation sets and then perform some experimentation to decide about a good validation set.

In our CST example, we created three validation sets from our training data. The first validation set was a random sample of 1% projects from our training data. The second validation set entails all projects that were completed during the last month in our data. The last one involved projects which were completed in the last quarter in the data. We then trained several models and checked their performance on these validation sets. We choose the validation set that holds a linear relationship between the training and validation scores for all models. In our CST example, the validation set containing projects from the last month demonstrated linear relationship between training and validation scores. This validation set is chosen for subsequent experimentation in our study. As discussed earlier, we always stick to the naming conventions for data sets since they are used internally by our generic `print_score()` function to check the errors and accuracies of models during our experimentation.

3.3.3. Devising Performance Tuning Strategy

The next step in the AML process is to formulate a tuning strategy to improve the performance of our initial estimator. There is no single best strategy to tune the estimator. However, a structured approach to performance tuning is key to conduct an effective experimentation. Most modelling decisions at this task depends on our choice of ML algorithms. We chose the random forest in this study. Two areas require careful consideration during performance tuning of random forests. These include `subsampling` to quickly conduct experimentation and `hyperparameter tuning` to understand what options would enhance the estimator learning abilities on the given task.

i). **Subsampling:** It is generally not advisable to utilise all training data for the baseline estimator at first. Otherwise, our experimentation will take considerable time, which is not good to carry out an effective exploration of different design strategies. In the case of a random forest with ten decision trees (estimators), it will take considerable amounts of time to train each estimator with all data. So, we pick different random subsets of data to construct each decision tree during our training phase. In addition to reducing training time, this strategy also introduces randomness by picking different subsets of data for estimators, which is key for random forest algorithm to learn complex domains. The more the trees vary in a forest, the better would be the predictive accuracy of the model. This strategy of using a small subset of data during training is generally called subsampling. In our CST example, the `scikit-learn` library does not provide any provision to customise its default mechanism which uses entire data during training the model. We wrote the following code to override the default behaviour in `scikit-learn` library to achieve subsampling:

```
forest._generate_sample_indices = (lambda rs, n_samples:
        forest.check_random_state(rs).randint(0, n_samples, n))
```

Where `n` is the number of random samples used to train each tree. We performed a bit of experimentation to obtain a good sample size for this data set. Our experimentation revealed `n` of size 50,000 a good option. Using the subsampling, the accuracy of baseline estimator (R^2) went to 87.93% and 86.19% for training and validation scores. This model is more reliable than our first model with training and validation accuracies (R^2) of 97.13% and 86.04%, respectively. There is a huge drop in performance of the earlier baseline estimator due to overfitting.

ii). **Hyperparameter Tuning:** Hyperparameters are the knobs for ML engineer to adjust ML algorithms for the given task. Their values cannot be learned from data; instead, ML engineers need to decide these values before the training begins. Their right choice can greatly boost the learning abilities of ML models. The whole idea of hyperparameter tuning is to explore the search space for various combinations of parameters that would yield superior performance for ML models. It can be extremely computation-intensive operation if we check for all permutations. Approaches like Grid Search and Randomised Search are vastly used techniques by ML engineers. Over the period, engineers get familiar with these hyperparameters and mostly know when they would work. Hyperparameters vary from one algorithm to another. Once we find out good parameter values, we start training more ML models and perform extensive feature engineering until we get to our final model. In our CST example, we tuned the model for `n_estimator`, `max_depth`, `min_samples_leaf`, `max_features` and `oob_score`. These hyperparameters are only relevant to the random forest algorithm. The `n_estimator` specifies the number of trees in the random forest. The key idea behind random forest to combine several weak estimators to get one powerful estimator. So it always worth exploring different number of trees and see what would work. The values like 1, 5, 10, 25, 40, 80, and 100 are tried. The `n_estimators = 80` achieved model's performance for training and validation accuracies to 95.80% and 88.37%, respectively. The `min_samples_leaf` specifies the minimum number of samples required at each node. The values of 1, 3 and 5 are tested, and model with `min_samples_leaf = 3` gained better training and validation accuracies of 97.20% and 90.19%, respectively. The `max_depth` specifies maximum depth of the tree in random forest. While we tried several values, this parameter didn't contribute much to the accuracy of our model. The `max_feature` specifies the maximum features to be included at each split for the best fit. Values of `none`, `0.5`, `sqrt`, and `log2` are tested. The training and validation performance of 97.02% and 90.66% respectively is achieved with `max_feature = 0.5`.

3.3.4. Choosing Baseline Estimator

This is the last step in training our baseline estimator task. We carried out extensive experimentation on hyperparameter tuning and subsampling to learn which configurations of the random forest will yield us the best estimator. Once these steps are complete, we train our baseline estimator and evaluate its predictive performance. It is the baseline estimator obtained from best design strategies and hyperparameters is used for interpretable machine learning. In our CST example, we trained baseline model using random forest for profit margin estimation with subsampling (`n = 50,000`), `n_estimators = 80`, `max_features = 0.5`, `min_samples_leaf = 3`, and `oob_score = True`. The baseline estimator has the training and validation accuracies of 97.02% and 91.67%, respectively.

3.4. ***Interpretable Machine Learning***

Machine learning (ML) is a great technology, but industry folks will never take it for our word. They ask a series of questions out of their curiosity to understand ML capabilities and weaknesses as intelligent solutions are getting deployed in their line of work software. They will not use ML with confidence until they fully trust it, which requires significant verifications. A piece of simple advice for ML folks is to let the model corroborate what industry folks already know. People in the industry will start trusting our models even if they do not know much about ML technology. For this reason, our applied machine learning (AML) process put special emphasis on explaining our models which go beyond just making good predictions. It is about enabling models to answer the critical question of the end-users of the system. Otherwise, the wider adoption of ML into the construction industry will not be achieved. This field of exploring the strengths and weaknesses of ML models by analysing their internals is called interpretable machine learning, which will be the focus of this section.

A common misconception about contemporary ML is the assumption that these models are black boxes. Nobody knows what they learn and how they make predictions. Their predictions are great, but what logic drove that prediction. Surprisingly, most ML engineers are barely aware of the approaches for analysing ML models. The entire field of interpretable machine learning is to understand the competencies of our ML models. This field is vast. We intend to cover a few techniques to answer some key questions asked by the industry folks. The chosen questions are the ones asked by most of the industry folks whenever we deployed ML algorithms as part of the CST. These questions include (but are not limited to):

a. *How confident are you of the predictions?*
b. *What attributes drive the predictions?*
c. *How attributes interact with others to drive the predictions?*
d. *How much attributes contribute to the predictions?*
e. *How good the model can extrapolate unseen data?*

Most tasks in interpretable machine learning require ML engineers to work with domain experts actively during the AML process. It enables the ML engineers to progressively learn the domain and make informed decisions to revise data and models. Tasks in this section are vital for many ML operations ranging from debugging models to feature engineering to future data collection to facilitate human decision making and building trust. Interpretable machine learning is the cornerstone of our proposed AML process. We will see several interpretable machine learning techniques for answering the questions above.

Algorithm 1: Predictions Confidence Assessment Algorithm

```
   Data: Validation Set & Baseline Estimator
   Result: Confidence Score for Categorical Attributes
 1 Make Predictions for All Projects in the Validation Set ;
 2 Retrieve Tree-Level Predictions for Each Project in the Validation Set ;
 3 foreach Categorical Attribute in Validation Set do
 4     foreach Category in Categorical Attribute do
 5         Compute Actual Margin, Average Predicted Margin, & SD of Tree-Level Predictions;
 6         Calculate the Confidence Score by Dividing the SD of Predictions over the Actuals ;
 7         Tabulate Confidence Scores for the Category
 8     end
 9     Plot the Data Distribution of Categorical Attribute;
10     Correlate SD of Categories with Number of Rows in the Validation Set ;
11 end
```

3.4.1. Checking Predictions Confidence of Baseline Estimator

A common aphorism in statistics is that all models are wrong, but some are useful. Nobody can claim their model is 100% accurate regardless of what data or algorithm we choose. This inability is partly because real-world phenomena have many perspectives which are hard to capture in one model holistically. This aphorism brings us to an important inquiry which we mostly received from end-users. They want to know the competencies of our ML models. In our CST example, we were often asked about how the confidence of models varies across different types of projects. To elaborate on this analysis, we will first explain the concept of confidence in random forests. Random forest is an ensemble of trees where trees make predictions, and random forest aggregates their predictions into the outcome. So, if each tree predicts a slightly different profit margin, it indicates higher confidence. Otherwise, huge disagreement in the predictions from trees is a sign that the model is less confident. The only reason predictions of trees vary in random forest is if projects, we are predicting for, are either non-existent in our data, or are sparse. Random forest has no rules to make consistent predictions; thereby, those projects end up in entirely leaf nodes in different trees. This variation of individual tree predictions can be analysed using standard deviation (SD). We can figure out the confidence for our models from that ratio of SD of predictions. The model will be confident of its

Table 3: Confidence of Estimator by Project Size

Project Size	**Confidence Score**
Large	0.034508
Small	0.029514
Medium large	0.027730
Medium	0.026984
Mini	0.026122

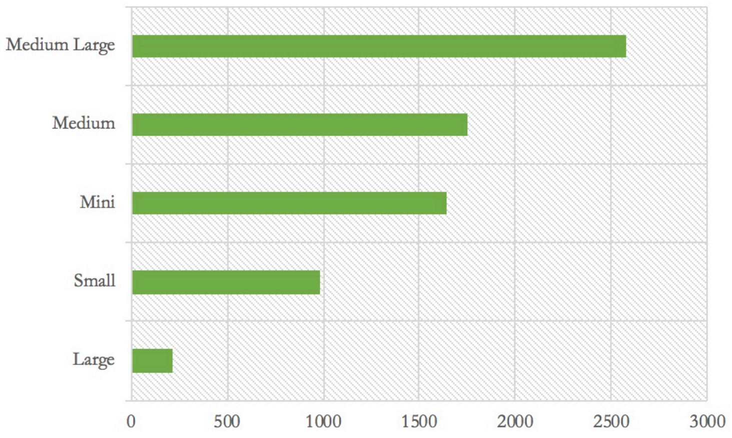


Figure 3: Data Distribution by Project Size

predictions if SD across all tree predictions is low otherwise less/no confidence. There is no built-in library to check the predictions confidence of ML models. We performed several steps, as shown by the algorithm 1, to perform this analysis.

In our CST example, we explored the predictions confidence of our baseline estimator for different types of projects. For the sake of brevity, we will only discuss this analysis for one categorical attribute, i.e. `project size`. In real-life, we analyse predictions confidence across all categorical attributes to truly understand the strengths and weaknesses of our models. Confidence score in the algorithm is the ratio of the standard deviation of tree-level predictions over the predicted profit margin. Table 3 shows the confidence score for all categories in the `project size` attribute. The higher the confidence score, the lesser the confident the model is on predictions for that category. In this case, `large` projects category has the confidence score of 0.0345 which shows that our baseline estimator is less confident for predictions on `large` projects. There are several reasons for this lack of confidence. One major reason is the uneven distribution of projects in our data set.

Fig 3 shows the distribution of projects in our data by the `project size`. The distribution clearly shows that `large` projects are fewer in our data which has caused a drop in the confidence of our model. Based on this insight, we requested additional data with a predictive theory that more data will improve predictions confidence of our model for `large` projects. Once additional data for `large` projects is received and re-trained our baseline model. We found that the confidence score of our baseline model has slightly improved. In this way, we were able to use insights to inform our modelling strategies, which was the future data collection requirements.

3.4.2. Checking Attributes Driving Predictions

The next question often asked of ML engineers is about the importance of attributes. Attribute importance is an excellent topic in ML, which reveals the significance of an attribute on model performance.

Several algorithms are devised to perform attribute importance. The popular ones include permutation importance and model reliance. We used permutation importance in this study because it is fast and model agnostic approach. One benefit of permutation importance is that we compute it without training ML models on every subset of our data. There is no need for training models as it is always performed after the model is trained. This concept of permutation importance is straightforward. The importance of an attribute can be determined from the increase in the model prediction error after we shuffle data in that attribute. We leave rest of attributes intact. The prediction error of the model shall increase if that attribute is vital to the model. The shuffling of data is performed to break the actual relationship between that attribute and the target attribute. The higher the prediction error go, the more important the attribute is to the model. Algorithm 2 illustrates key steps involved in the permutation importance.

Algorithm 2: Permutation Importance

```
  Data: Trained model f, attribute matrix X, target vector y, error measure L(y,f)
  Result: Importance Score for All Attributes
1 Estimate the original model error $e^{orig}$ = L(y, f(X))
2 foreach attribute a in X do
3     Generate attribute matrix $X^{perm}$ by permuting attribute a in the data X ;
4     Estimate error $e^{perm}$ = L(Y,f(Xperm)) based on the predictions of the permuted data ;
5     Calculate permutation importance $I^j = e^{perm}/e^{orig}$ ;
6 end
7 Sort attributes by descending importance
```

In our CST example, we employed `PermutationImportance` class from python `eli5.sklearn` library. Fig. 4 show the importance of all attributes used for predicting the profit margins. Attributes with higher importance scores are at the bottom of the plot. The model finds these attributes important in driving its predictions. The attributes having lower scores are least significant to the baseline model. Since there is a degree of randomness to the exact performance by shuffling data, this library shuffles data several times to ensure that the real relationship between attributes and target is fully broken. These scores can be negative for some attributes, which reveals that the model attributes no significance to these attributes. Occasionally, it is worth removing such those attributes.

Fig. 4 shows that project complexity in terms of risks, opportunities and the distance which construction route travels through the rivers, roads, rails and utilities, are amongst the most important attributes. Besides, the allocation of resources in terms of the project manager (PM), quantity surveyor (QS), commercial manager, design manager, suppliers and subcontractors are also meaningful in predicting profit margins. Materials, though, are not as significantly important to the model. At this stage, the ML engineers work in tandem with domain experts to discuss results and learn the required domain knowledge. When we shared these results with estimators, they agreed with the attribute importance chart. Our model was able to verify the knowledge estimators had, which helped them to gain confidence in our model. While attribute importance is a great tool for ML engineers to explain the importance of attributes, it is also useful to drive several feature engineering steps. It is worth recognising that our proposed AML process delayed feature engineering so far. In the beginning, we performed a minimum of necessary feature engineering but not based on domain relevance, which most people suggest. The chart in Fig. 4 shows that many attributes fall on the long tail are least important and could be deleted. With several experimentations, we found that we can delete almost half of attributes without significantly losing the model performance. We discussed our findings with end-users and deleted some of those attributes as soon as industry folks agreed. Most of these attributes had the importance of less than 0.05. The removal of attributes slightly changed the importance of other attributes. This is partly due to the correlation between various attributes. When we deleted the least significant attributes, the remaining correlating attributes became more visible and their importance scores improve. This type of deletion enables us to develop efficient models with fewer attributes.

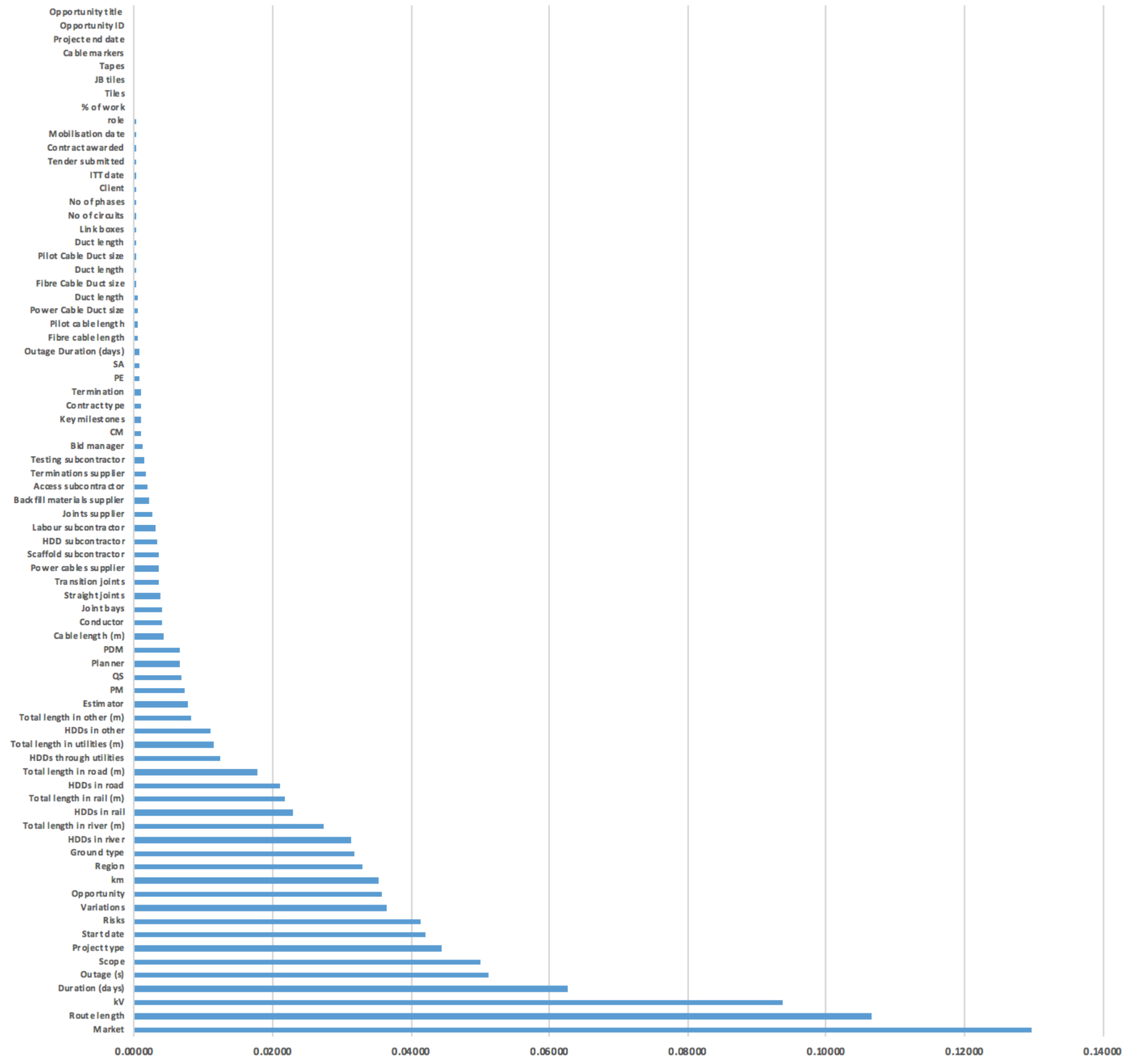


Figure 4: Attribute Importance Chart

3.4.3. Identifying Multicollinearity & Removing Attributes

In the previous section, we slightly talked about the issue of multicollinearity and its effect on model when we delete attributes. Multicollinearity occurs when input attributes correlate with each other. In simple words, a subset of attributes is likely to supply overlapping information to our estimator. Multicollinearity not only adversely affects the performance of ML models but also obscure some important attributes by revealing similar information during the training process. It incapacitates the model's abilities to understand attributes importance. The performance of models can be significantly improved by removing such confounding attributes from our dataset. As a result, we will achieve simpler but more robust models which will be based on fewer input attributes. Such models are always efficient and are hard to overfit. However, the choice of removing attributes shall not be entirely subjected to the intuitions of ML engineers or domain experts. Instead, their judgements shall be informed with some objective evidence. To this end, there are several statistical approaches to find out the similarity between attributes in datasets. Unsupervised ML techniques like clustering can play a significant role to inform ML engineering during the decision of removing attributes. Feature importance, performed in the previous section, is useful to have a feel of least important attributes and remove them from our data. Identifying correlated attributes from the dataset are a key step in our proposed AML process. Supervised ML approaches like clustering are specialised for this type of exploration. Their accuracy exceeds other traditional ML approaches.

Algorithm 3: Multi-Collinearity Assessment Algorithm

```
  Data: Trained model f, Training data X, Target vector y, Clusters C
  Result: Table (T) containing model accuracy as attributes are removed from data
1 foreach Cluster c in Clusters C do
2     foreach Attribute a in c do
3         Drop c from X ;
4         Train model (m) on data without c ;
5         Compute accuracy of m ;
6         Add tuple cluster (c), attribute (a) and accuracy to table T ;
7     end
8 end
```

In our CST example, we used hierarchical agglomerative clustering for finding the similarity between input attributes. We calculated the Spearman ranked correlation and turned it into a distance matrix. Distance matrix provides necessary information to construct the hierarchy between attributes based on their similarity. Fig. 5 shows the results of agglomerative clustering using dendrogram, where attributes are divided into various groups. We can quickly spot similar attributes in the dendrogram as they fall closer to each other under a single parent. While clustering provides an abstract grouping of attributes that worth investigation for multicollinearity. This information shall still be verified from domain experts. We shared these clusters with our industry folks who acknowledged the results of the clustering algorithm. The clustering algorithm suggested eleven (10) groups where one or several attributes are collinear. For example, the clustering algorithm revealed that attributes `Total length in river (m)`, `Total length in rail (m)`, `Total length in road (m)`, `Total length in utilities` and `Total length in other (m)` in `cluster 3` are correlating with `Horizontal Directional Drilling (HDD) in river`, `HDD in rail`, `HDD in road`, `HDD in utilities` and `HDD in other`, respectively. We then used algorithm 3 to find out the feasibility of removing attributes from these group. The overall aim is to remove those attributes where the accuracy of the model does not drop too much. According to experts, horizontal directional drilling (HDD) is a specialised activity and one of the highly risky activity on a project. It is a method of creating cable trenches underneath rivers, rail, bridges and public roads such that the path of the cable continues. The system dictates that 'no of HDDs in...' is not as strong as 'total length of HDDs in...'. The is backed up by the industrial experts, and the argument is that the total length of HDDs in any of the above-mentioned scenario allows the project team to develop a fail-safe mitigation strategy to performing this task. The total length of HDDs translates to risk levels, risk pot percentages, subcontractor hire, drill and drill bit hire, permit orders and environmental distress. The more there is to do the more risks there are. Therefore, we dropped `Horizontal Directional Drilling (HDD) in river`, `HDD in rail`, `HDD in road`, `HDD in utilities` and `HDD in other` and left their counterparts.

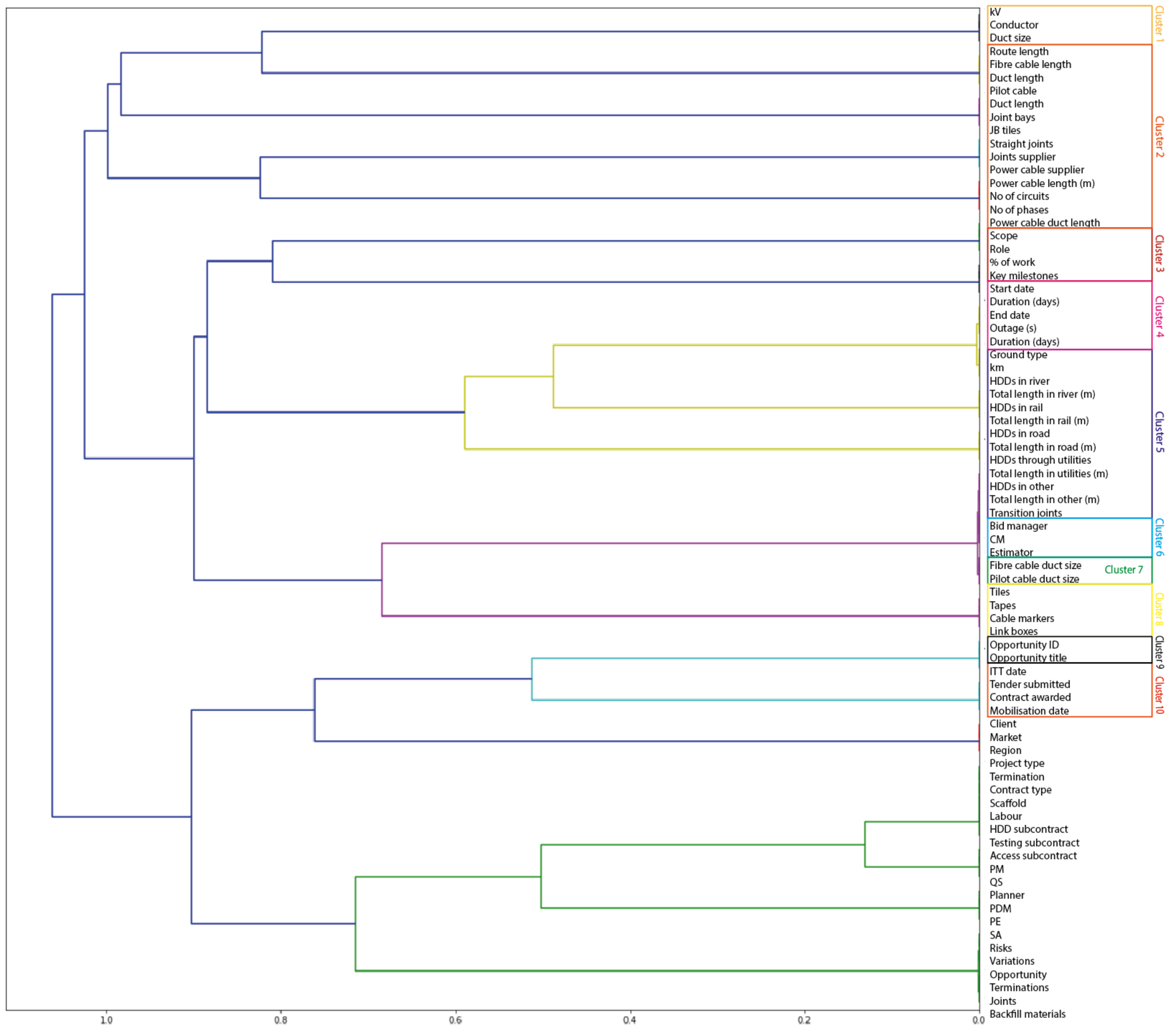


Figure 5: Agglomerative Clustering for Attributes Similarity

Similarly, the route length of a project was found to have several attributes correlated to it such as the power cable length, no of circuits, no of phases, no of joint bays, fibre cable length etc. Experts explained that the route length plus the no of circuits and the no of phases come together to give the total length of power cable needed. As you can see here, there are two correlation levels between the route length and cable length and between no of circuits and phases and cable length. But the ultimate parent attribute to all is the route length. Likewise, there is one correlation between the route length and total length of fibre and route length and pilot cables to be used on the project. Whereas in the first example of the HDDs and total length in HDDs areas where some attributes were completely removed from the system, the system shows that in this instance between route lengths and its correlated and sub correlated attributes, some attributes still demonstrate very strong contributions to the profitability such as the cable length and no of joint bays. Therefore, not all child-attributes were deleted in this example, only those whose contributions were completely mundane and those that industry experts have acknowledged their redundancy were deleted. After we dropped several attributes from each group, the training and validation accuracy of our model slightly dropped from 93.33% and 91.32% to 89.89% and 87.12% respectively.

3.4.4. Extrapolating Estimator

Most ML models are excellent at interpolation, which means that they can predict with higher accuracy what is known to them (i.e. data from training sets). However, another important feature of robust ML models is extrapolation, which is about making predictions for what is outside of these known, i.e. validation or unseen real-life data. Extrapolation reveals the generalisability of a model. Overfitting is a major cause of restricting the extrapolation capabilities of ML models. However, we can disclose the overfitting of a model by devising good validation sets. Oftentimes, ML engineers use attributes during the model training, which restrict models from extrapolation. This issue becomes more prominent for tree-based models as their predictive accuracy drastically declines during production on unseen data if such attributes are involves in rules in tree nodes. Temporal attributes like year is one example of attributes that can restrict a model to perform reliably in real-life scoring. A tree model cannot accurately predict for projects which are from a year that is not available in the training data. The deletion of such features is one of the key step in the AML process to overcome overfitting and achieving robust ML models.

In our CST example, there were several attributes with date elements such as `Start date`, `End date`, `ITT date`, `Tender submission date`, `Contract award date` and `Mobilisation dates`. Besides, attributes like `Opportunity id` and `Opportunity title` are sort of unique identifiers and have no relevance for predicting the profit margin of a construction project. We performed thorough experimentation to see how the removal of these type of attributes affects the performance of our model. In theory, the validation score of the model shall improve if we remove attributes that hinder the generalisability of our model. We derived `Project Duration` from attributes like `Start date` and `End date` and then dropped those attributes from our training data. All unique identifiers are also removed from the data. The predictive accuracy of the model is checked before the deletion of all temporal or unique identifiers.

In most cases, the model was able to achieve higher accuracy on validation data. The final model has fewer attributes but has a higher accuracy of 95.23% and 93.44% on training and validation data, respectively. Our model is intended to be used for real-life scoring on future projects. So, we removed the time-dependent attributes and unique identifiers from our data. In this way, we were able to increase the extrapolation of our ML model for future projects and tackle the issue of overfitting.

3.4.5. Model Interpreter

Another common end-users requirement is to show the breakdown predictions. In our CST example, how the model factored out project-specific nuances to build the profit margin. The end-users this explanation for clients to explain the output is higher or low than their expectations. There are several techniques to breakdown prediction for ML models. In the case of random forest, each tree node contains a value which is the sum of the target feature for all the rows contained in that specific node. So, when we traverse the tree

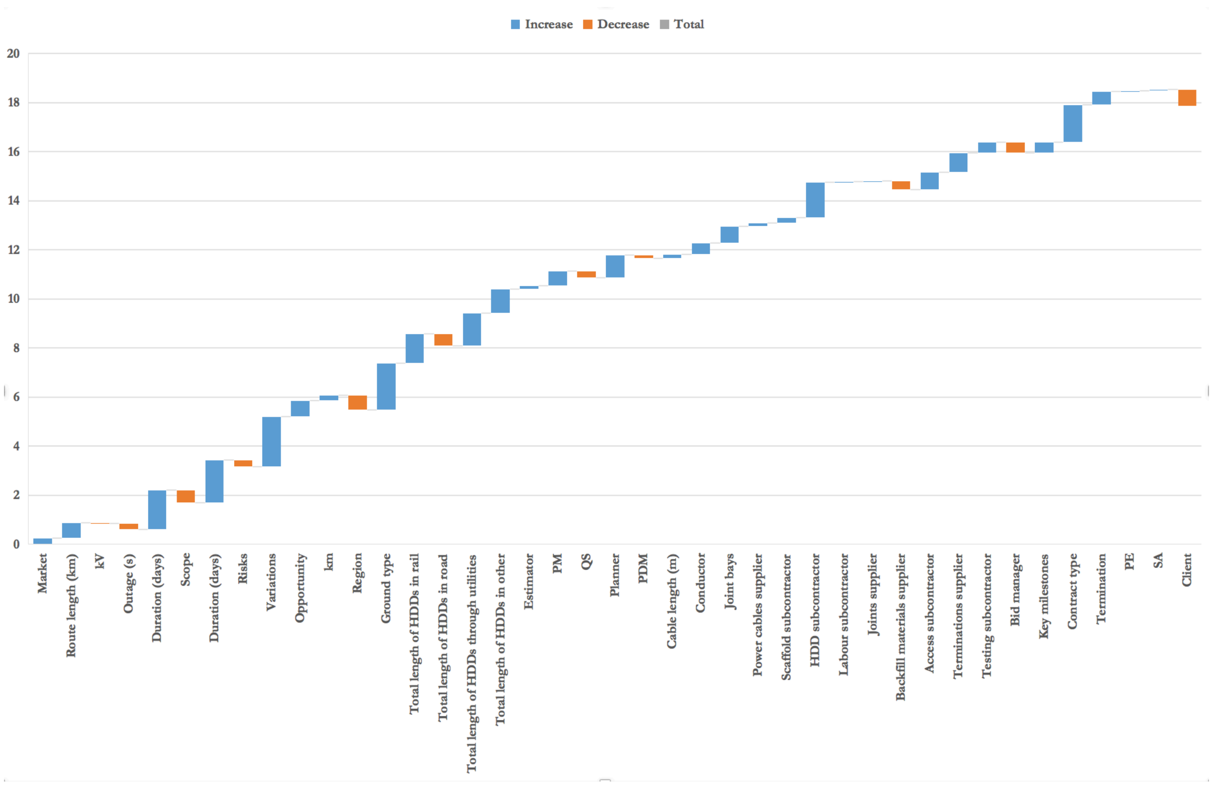


Figure 6: Tree interpretability for profit margin forecasting

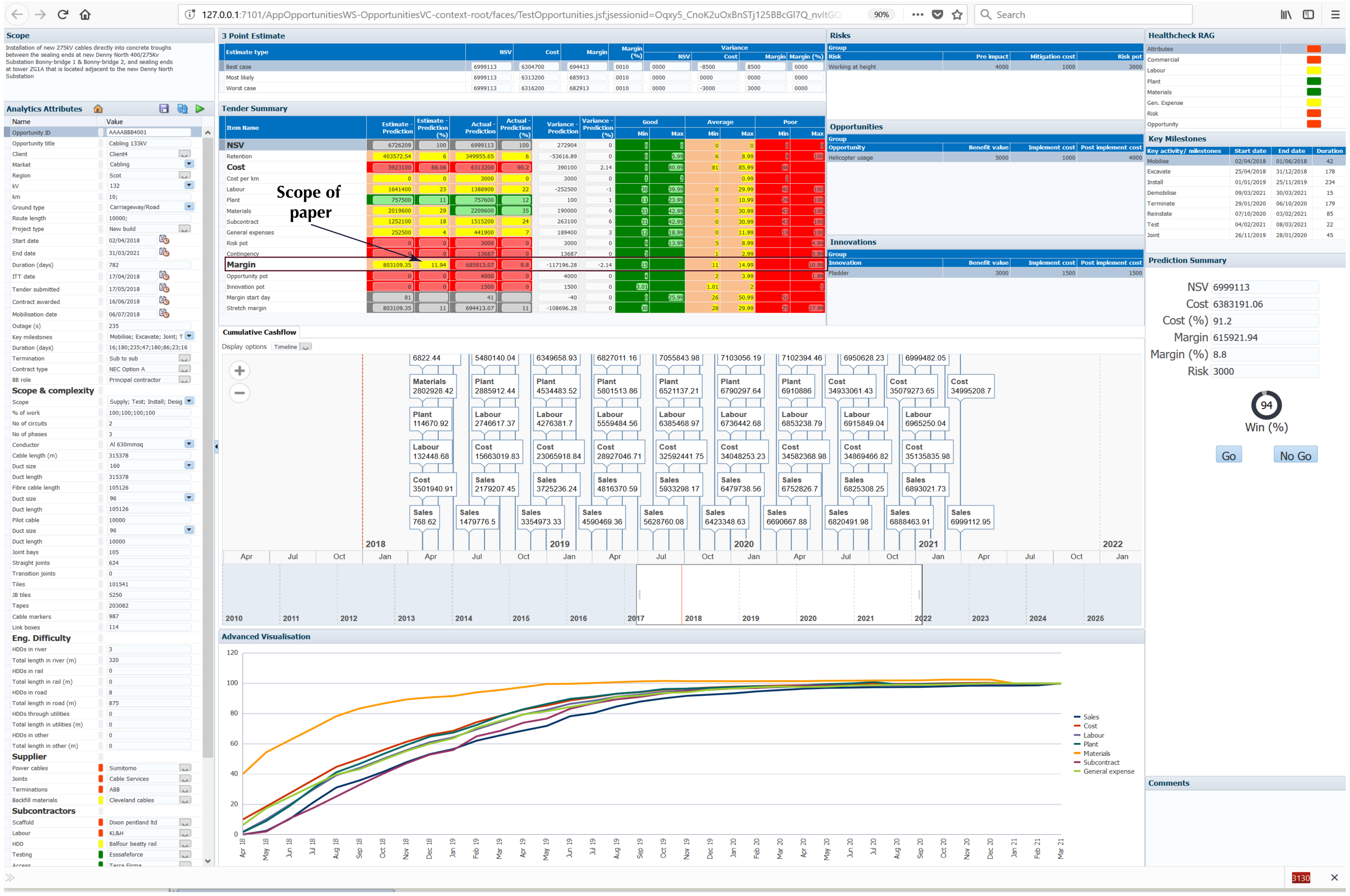
from top to down, this value fluctuates on every split of the node. This fluctuation can be used to calculate the contributions of attributes to final predictions.

In our CST example, we used python package `treeinterpreter` for this purpose. This package has a `predict()` function which accepts the model and the row for which we want to predict the profit margin and returns predictions, bias and contributions of each attribute towards the prediction. We used a power transmission and distribution project as a case study. The scope of the project was to design, excavate, lay, joint and terminate 2500mm2 copper cables through a mildly rural and urban area between two power poles. The model estimated a margin of 17.86% and displayed a rare functionality of ML applications by revealing the process to calculate the project margin, as shown in Fig 6 using waterflow chart. The industry folks realised that the model has learnt that a 3.5km, 110kV cabling project with just one outage is widespread and the firm makes a profit from these. Whereas with attributes such as HDD in rail, and the firm rarely approve projects that require directional drilling through a rail track because of the complexity and social and economic distress associated with it. Therefore, the system is right to push the margin of this attribute higher than the others. Likewise, the system evaluated the possible profitable variations to be made from the project and understand that this is too low for this type of project; hence, it raises it's associated margin higher. These experts explored each percentage against their knowledge and data to validate the process the system has adopted. At the end of the exploration, the experts praised the model and expressed their support for the further development of the tool.

3.5. The Final Model

So far, we have performed a lot of experimentation to achieve two major things. Firstly, we explored the hyperparameters space to see what values would enhance the learning abilities of the underlying algorithm for the given dataset. Secondly, we examined the data through Interpretable Machine Learning to understand the suitability of input attributes for the ML task. During the AML process, we realised that some attributes are insignificant to our modelling task, which were discarded. We also observed that some attributes correlate with others and need to be removed as they confuse the model. Therefore, we removed such attributes. ML engineers shall have a good idea at this AML process stage about what data and hyperparameters would work for the given task. We start training our final estimator at this stage. We slightly talked about the test set at the beginning that it is advised to hold it out for later evaluation. Test sets become relevant at this stage to know the final accuracy of our estimator before the final model is trained. To this end, we first merge the training and validation sets. Then use the test set to evaluate our trained model using merged data. However, we train the final model on all datasets, including training, validation and test sets. There is a substantial reason to utilise entire data for training the final model. The model will not be able to learn patterns of variations from the validation and test sets if we skip those datasets during the training. Consequently, the predictive accuracy of the model will drift significantly for unseen data due to gaps in historical data used for its training. We shall ensure to combine all datasets before start training the final estimator. The training of the final model relatively takes more time. We shall stick to the best values of hyperparameters that were discovered during the process.

In our CST example, we combined two datasets `X_train` and `X_valid` into `X` tensor and then trained the random forest model. The model has attained the accuracy of 96.77% and 94.91% on the training (`X`) and test (`X_test`) sets. This accuracy is reasonably high. In addition, the generalisability of the model is significant for the unseen data from the `X_test` set. The test data is eventually merged and the final model is trained. We stored the model into `.pkl` file to be utilised in our opportunities analytics dashboard for production deployment.


127.0.0.1:7101/AppOpportunitiesWS-OpportunitiesVC-context-root/faces/TestOpportunities.jsf;jsessionid=Oqxy5_CnoK2uOxBnSTj125BBcGl7Q_nvltGQ

Scope

Installation of new 275kV cables directly into concrete troughs between the sealing ends at new Denny North 400/275Kv Substation Bonny-bridge 1 & Bonny-bridge 2, and sealing ends at tower ZG1A that is located adjacent to the new Denny North Substation

Analytics Attributes

Name	Value
Opportunity ID	AAAABBB4001
Opportunity title	Cabling 133kV
Client	Client4
Market	Cabling
Region	Scot
kV	132
km	10;
Ground type	Carriageway/Road
Route length	10000;
Project type	New build
Start date	02/04/2018
End date	31/03/2021
Duration (days)	782
ITT date	17/04/2018
Tender submitted	17/05/2018
Contract awarded	16/06/2018
Mobilisation date	06/07/2018
Outage (s)	235
Key milestones	Mobilise; Excavate; Joint; T
Duration (days)	16;180;235;47;180;86;23;16
Termination	Sub to sub
Contract type	NEC Option A
BB role	Principal contractor
Scope & complexity	
Scope	Supply; Test; Install; Desig
% of work	100;100;100;100
No of circuits	2
No of phases	3
Conductor	Al 630mmsq
Cable length (m)	315378
Duct size	160
Duct length	315378
Fibre cable length	105126
Duct size	96
Duct length	105126
Pilot cable	10000
Duct size	96
Duct length	10000
Joint bays	105
Straight joints	624
Transition joints	0
Tiles	101541
JB tiles	5250
Tapes	203082
Cable markers	987
Link boxes	114
Eng. Difficulty	
HDDs in river	3
Total length in river (m)	320
HDDs in rail	0
Total length in rail (m)	0
HDDs in road	8
Total length in road (m)	875
HDDs through utilities	0
Total length in utilities (m)	0
HDDs in other	0
Total length in other (m)	0
Supplier	
Power cables	🟥 Sumitomo
Joints	🟥 Cable Services
Terminations	🟥 ABB
Backfill materials	🟨 Cleveland cables
Subcontractors	
Scaffold	🟥 Dixon pentland ltd
Labour	🟥 KL&H
HDD	🟨 Balfour beatty rail
Testing	🟩 Esssafeforce
Access	🟩 Terra Firma

3 Point Estimate

Estimate type	NSV	Cost	Margin	Margin (%)	Variance NSV	Variance Cost	Variance Margin	Variance Margin (%)
Best case	6999113	6304700	694413	0010	0000	-8500	8500	0000
Most likely	6999113	6313200	685913	0010	0000	0000	0000	0000
Worst case	6999113	6316200	682913	0010	0000	-3000	3000	0000

Tender Summary

Item Name	Estimate - Prediction	Estimate - Prediction (%)	Actual - Prediction	Actual - Prediction (%)	Variance - Prediction	Variance - Prediction (%)	Good Min	Good Max	Average Min	Average Max	Poor Min	Poor Max
NSV	6726209	100	6999113	100	272904	0	0	0	0	0	0	0
Retention	403572.54	6	349955.65	6	-53616.89	0	0	5.99	6	8.99	9	100
Cost	5923100	88.06	6313200	90.2	390100	2.14	0	80.99	81	85.99	86	
Cost per km	0	0	3000	0	3000	0	1	2		0.99	2	
Labour	1641400	23	1388900	22	-252500	-1	30	39.99	0	29.99	40	100
Plant	757500	11	757600	12	100	1	11	23.99	0	10.99	24	100
Materials	2019600	29	2209600	35	190000	6	31	42.99	0	30.99	43	100
Subcontract	1252100	18	1515200	24	263100	6	31	42.99	0	30.99	43	100
General expenses	252500	4	441900	7	189400	3	12	18.99	0	11.99	19	100
Risk pot	0	0	3000	0	3000	0	9	13.99	5	8.99		4.99
Contingency	0	0	13687	0	13687	0	3		1	2.99		0.99
Margin	803109.35	11.94	685913.07	9.8	-117196.28	-2.14	15		11	14.99		10.99
Opportunity pot	0	0	4000	0	4000	0	4		2	3.99		1.99
Innovation pot	0	0	1500	0	1500	0	3.01		1.01	2		1
Margin start day	81		41		-40	0	0	25.99	26	50.99	51	
Stretch margin	803109.35	11	694413.07	11	-108696.28	0	30		28	29.99	25	27.99

Scope of paper

Risks

Group			
Risk	Pre impact	Mitigation cost	Risk pot
Working at height	4000	1000	3000

Opportunities

Group			
Opportunity	Benefit value	Implement cost	Post implement cost
Helicopter usage	5000	1000	4000

Innovations

Group			
Innovation	Benefit value	Implement cost	Post implement cost
Pladder	3000	1500	1500

Healthcheck RAG

Attributes	🟥
Commercial	🟥
Labour	🟨
Plant	🟩
Materials	🟩
Gen. Expense	🟨
Risk	🟥
Opportunity	🟥

Key Milestones

Key activity/ milestones	Start date	End date	Duration
Mobilise	02/04/2018	01/06/2018	42
Excavate	25/04/2018	31/12/2018	178
Install	01/01/2019	25/11/2019	234
Demobilise	09/03/2021	30/03/2021	15
Terminate	29/01/2020	06/10/2020	179
Reinstate	07/10/2020	03/02/2021	85
Test	04/02/2021	08/03/2021	22
Joint	26/11/2019	28/01/2020	45

Prediction Summary

NSV	6999113
Cost	6383191.06
Cost (%)	91.2
Margin	615921.94
Margin (%)	8.8
Risk	3000

94 Win (%)

Go No Go

Cumulative Cashflow

Display options Timeline

6822.44 | 5480140.04 | 6349658.93 | 6827011.16 | 7055843.98 | 7103056.19 | 7102394.46 | 6950628.23 | 6999482.05

Materials 2802928.42 | Plant 2885912.44 | Plant 4534483.52 | Plant 5801513.86 | Plant 6521137.21 | Plant 6790297.64 | Plant 6910886 | Cost 34933061.43 | Cost 35079273.65 | Cost 34995208.7

Plant 114670.92 | Labour 2746617.37 | Labour 4276381.7 | Labour 5559484.56 | Labour 6385468.97 | Labour 6736442.68 | Labour 6853238.79 | Labour 6915849.04 | Labour 6965250.04

Labour 132448.68 | Cost 15663019.83 | Cost 23065918.84 | Cost 28927046.71 | Cost 32592441.75 | Cost 34048253.23 | Cost 34582368.98 | Cost 34869466.82 | Cost 35135835.98

Cost 3501940.91 | Sales 2179207.45 | Sales 3725236.24 | Sales 4816370.59 | Sales 5933298.17 | Sales 6479738.56 | Sales 6752826.7 | Sales 6825308.25 | Sales 6893021.73

Sales 768.62 | Sales 1479776.5 | Sales 3354973.33 | Sales 4590469.36 | Sales 5628760.08 | Sales 6423348.63 | Sales 6690667.88 | Sales 6820491.98 | Sales 6888463.91 | Sales 6999112.95

2018
2019
2020
2021
2022
Apr Jul Oct Jan Apr Jul Oct Jan Apr Jul Oct Jan Apr Jul Oct Jan Apr Jul Oct Jan
2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023 2024 2025
Advanced Visualisation
120
100
80
60
40
20
0
Apr 18 May 18 Jun 18 Jul 18 Aug 18 Sep 18 Oct 18 Nov 18 Dec 18 Jan 19 Feb 19 Mar 19 Apr 19 May 19 Jun 19 Jul 19 Aug 19 Sep 19 Oct 19 Nov 19 Dec 19 Jan 20 Feb 20 Mar 20 Apr 20 May 20 Jun 20 Jul 20 Aug 20 Sep 20 Oct 20 Nov 20 Dec 20 Jan 21 Feb 21 Mar 21
Sales
Cost
Labour
Plant
Materials
Subcontract
General expense
Comments
3130

Figure 7: Machine Learning (ML)-driven Opportunities Analytics Dashboard

3.6. Production Deployment

The last step in our proposed AML process is the production deployment, where we put our models for real-life scoring. ML technology is elegant, but through enterprise solutions, we take it to the next level of producing actionable insights. These models often form a part of complex software. There are several technological challenges associated with production deployment. For prototyping our models, we tend to use python or R languages. However, the technologies behind enterprise applications may use entirely different programming languages. We need to integrate ML models into these systems to accrue the real benefits of automation. There are two widely used approaches to integrate ML models into enterprise solutions. The first is to rewrite the code for the model into the language of the system. This sounds interesting, but it entails enormous programming efforts. Besides, most programming languages are not suitable to efficiently perform heavy computation required by these models. Secondly, we can use web services for ML models to ensure language-agnostic deployment. This is the most popular option. Most software engineers use web services for invoking ML models from their enterprise applications. The first approach is more suitable for onboard AI systems where devices have limitations for speed, memory and connectivity, whereas the second approach is vastly adopted approach in most enterprise business applications.

In our CST example, we aimed to utilised our ML model in the Construction Simulation Tool (CST). The model discussed in this study is used for profit margin estimation on ML-based opportunities analytics dashboard. Fig. 7 shows the screenshot of the opportunities analytics dashboard. This is one of the key analytics dashboards provided in the CST. This dashboard contains all key information for understanding the suitability of project opportunities. Main users of this dashboard are bid managers and estimators. This dashboard gets access to a large number of predictive models for various tasks. During the model invocation, CST passes the key details of an opportunity to predictive models which predict key commercial information of the project including costs, margin, cash flow, project plan and risks to name a few. Most ML models are designed to guide estimators about specific tasks. CST is underpinned by a comprehensive benchmarking system which highlights different values into red, amber and green colours. The overall aim is to highlight weak aspect of the given opportunity quickly. CST automatically fetches design details of the construction project and formulate input query for the ML model without any human intervention. We deployed the profit margin estimation model as a Flask-based web service. CST uses REST API based business service to invoke this model. The result from the web services are obtained as JavaScript Object Notations (JSON) files, which are then deciphered by Structured Query Language (SQL) and shown it on the dashboard. The final decision is in the hands of the users who can modify or entirely override predictions of the model. In case of revisions to models predictions, CST logs these changes and use them to refine its predictions through lifelong learning.

4. Evaluation of Applied Machine Learning Process

We evaluated the proposed AML process by creating several machine learning models for various tasks involved in construction project planning and delivery. Table 4 shows the list of those tasks along with their descriptions. We primarily employed random forest and deep neural network for training ML models for these tasks. The discussion of which algorithm is superior over the other is beyond the scope of this study. These classes of algorithms have been successfully employed in diverse applications across different industries. Our focus in this section is to showcase the effectiveness of our proposed AML process towards creating robust ML models. Overall, these algorithms were able to train models with reasonable accuracies. For complex tasks, like predicting the contingency pot which is the additional profit that can be made through effective handling of contingencies in the project, we were able to train the deep neural network with predictive accuracies of 73.34% and 70.09%, respectively. Whereas, some easier tasks such as Bid/No Bid where model informs the decisions of whether the firm shall bid the given project or not, we attained 98.45% and 97.56% accuracies on training and test sets. In most cases, we were able to train models with accuracies above 80% on relatively challenging tasks. While we had access to vast amounts of relevant past data along with domain expertise, the most distinguished aspect of the work was the proposed AML process.

Table 4: Evaluation of AML Process for Several ML Tasks

Task No.	Machine Learning Task	Machine Learning Task Description	Deep Neural Network (DNN)		Random Forest	
			Training Accuracy	Validation Accuracy	Training Accuracy	Validation Accuracy
1	Win(%)	To predict the win probability of a project opportunity.	94.73	89.42	90.44	87.23
2	Retention (%)	Predicting the %age of payments that the firm shall let retain the client until the project finishes.	86.61	83.56	94.56	91.97
3	Labour cost	Predcting the %age of labour cost with respect to net sales value (NSV) for the project to achieve planned margins	88.01	86.51	95.53	92.78
4	Plant cost	Predcting the %age of plants & equipment cost with respect to net sales value (NSV) for the project to achieve planned margins	84.06	81.05	89.44	87.17
5	Materials cost	Predcting the %age of materials cost with respect to net sales value (NSV) for the project to achieve planned margins	91.34	90.11	90.67	89.1
6	Subcontract cost	Predcting the %age of subcontract cost with respect to net sales value (NSV) for the project to achieve planned margins	83.54	82.34	74.31	71.45
7	General cost	Predcting the %age of general expenses with respect to net sales value (NSV) for the project to achieve planned margins	81.45	78.34	74.23	60.55
8	Risk pot	Predicting additional profit that can made through effective mitigation of identified risks	85.87	76.56	84.56	82.56
9	Innovation pot	Predicting additional profit that can be made by bringing innovation in terms of materials or design strategy to the project	84.61	82.08	85.44	84.01
10	Contingency pot	Predicting additional profit that can be made through effective handling of contingencies in the project	73.34	70.09	63.45	60.33
11	Profit margin	Predicting the profit margin that can be made keeping in view project complexity, resourcing, supply chain and other attributes	89.34	84.34	96.77	94.91
12	Margin start day	Predictng the first day when firm will start making profit from the project	94.33	91.07	87.56	86.19
13	Bid/No Bid	Predicting whether the company shall bid for or not the given project	98.45	97.56	85.78	80.83

It is clear from the results that the difference between the training and validation scores for both classes of algorithms is low. The average difference between the training and validation scores for the deep neural networks is 3.28%, whereas that for the random forest is 3.35%. This minor difference reveals the robustness of our ML models, which is hugely desirable for models intended to be used in real-life applications.

5. Conclusions

In this paper, we presented guidelines for Applied Machine Learning (AML) for the construction industry. A common experience in most industries is that seemingly impressive ML models fail when deployed to real-life applications. The fallout includes losing the support for ML-based automation as people start suspecting ML capabilities and reluctant to pursue it further. Surprisingly, ML engineers, as well as construction folks, seldom know tools and techniques for developing robust ML models. ML engineers often make modelling choices about their data and algorithms based on intuition, which are often bias and misleading. As a result, ML models are not up to the expectations as ML steps were either omitted or not executed in proper way.

While ML is an engineering task, there are several ways to create robust ML systems. Currently, there is no literature source that provides construction IT folks about guidelines for developing great predictive models. This paper fills this void and presents the AML process in detail, which we learned over several years while developing the Construction Simulation Tool (CST). The process is elaborated in the context of profit margin estimation. Our AML process is evaluated for several ML tasks to ensure it works.

Interpretable Machine Learning is desirable for using these models in enterprise solutions. This explanation is key for end-users to trust ML systems. Besides, users require to understand the limitations of models so that they can manually override in case ML is making wrong predictions. Our proposed process included interpretable machine learning for ML engineers as key step to expose the internals of models. The ML engineers are not expected of prior domain knowledge; they learn it through ML models as they progress through the process. The process stimulates critical modelling tasks like feature engineering as ML engineers learn more about the domain from industry folks. This study is a part of developing a machine learning-based construction simulation tool which aims to harness historical data to automate or facilitate activities across construction activities involving opportunity selection, design optimisation, construction estimating and project execution. CST has employed a large number of ML models to support users' tasks at various stages of the lifecycle. This paper explains our proposed AML process, which was key to developing robust ML models in CST.

References

Akintoye, A., & Skitmore, M. (1991). Profitability of uk construction contractors. *Construction Management and Economics*, *9*, 311–325.

Bagies, A., & Fortune, C. (2006). Bid/no-bid decision modelling for construction projects. In *Procs 22nd Annual ARCOM Conference* (pp. 511–521). Birmingham.

Banaitiene, N., & Banaitis, A. (2012). Risk management in construction projects. In *Risk Management-Current Issues and Challenges*. IntechOpen.

Bau, D., Zhu, J.-Y., Strobelt, H., Zhou, B., Tenenbaum, J. B., Freeman, W. T., & Torralba, A. (2019). Visualizing and understanding generative adversarial networks. *arXiv preprint arXiv:1901.09887*, .

Bilal, M., Oyedele, L. O., Qadir, J., Munir, K., Ajayi, S. O., Akinade, O. O., Owolabi, H. A., Alaka, H. A., & Pasha, M. (2016). Big data in the construction industry: A review of present status, opportunities, and future trends. *Advanced engineering informatics*, *30*, 500–521.

Bose, I., & Mahapatra, R. K. (2001). Business data mining—a machine learning perspective. *Information & management*, *39*, 211–225.

Chang, P.-T., Hung, L.-T., Pai, P.-F., & Lin, K.-P. (2013). Improving project-profit prediction using a two-stage forecasting system. *Computers & Industrial Engineering*, *66*, 800–807.

Christodoulou, S. (2010). Bid mark-up selection using artificial neural networks and an entropy metric. *Engineering, Construction and Architectural Management*, *17*, 424–439.

Cox, V. (2017). Exploratory data analysis. In *Translating Statistics to Make Decisions* (pp. 47–74). Springer.

Dikmen, I., & Birgonul, M. T. (2004). Neural network model to support international market entry decisions. *Journal of Construction Engineering and Management*, *130*, 59–66.

Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, .

Dulaimi, M. F., & Shan, H. G. (2002). The factors influencing bid mark-up decisions of large-and medium-size contractors in singapore. *Construction Management & Economics*, *20*, 601–610.

Enshassi, A., Mohamed, S., & El Karriri, A. (2010). Factors affecting the bid/no bid decision in the palestinian construction industry. *Journal of Financial Management of Property and Construction*, *15*, 118–142.

Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning* volume 1. Springer series in statistics New York.

Gomber, P., Kauffman, R. J., Parker, C., & Weber, B. W. (2018). On the fintech revolution: interpreting the forces of innovation, disruption, and transformation in financial services. *Journal of Management Information Systems*, *35*, 220–265.

Hulten, G. (2018). Machine learning intelligence. In *Building Intelligent Systems* (pp. 245–261). Springer.

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*, 255–260.

Kim, S.-Y., Huynh, T.-A. et al. (2008). Improving project management performance of large contractors using benchmarking approach. *International Journal of Project Management*, *26*, 758–769.

Leon, H., Osman, H., Georgy, M., & Elsaid, M. (2017). System dynamics approach for forecasting performance of construction projects. *Journal of Management in Engineering*, *34*, 04017049.

Li, H., & Love, P. E. (1999). Combining rule-based expert systems and artificial neural networks for mark-up estimation. *Construction Management & Economics*, *17*, 169–176.

Li, H., Shen, L., & Love, P. (1999). Ann-based mark-up estimation system with self-explanatory capacities. *Journal of construction engineering and management*, *125*, 185–189.

Lin, C.-T., & Chen, Y.-T. (2004). Bid/no-bid decision-making–a fuzzy linguistic approach. *International Journal of Project Management*, *22*, 585–593.

Luong, M.-T., Le, Q. V., Sutskever, I., Vinyals, O., & Kaiser, L. (2015). Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*, .

Manyika, J. (2017). A future that works: Ai, automation, employment, and productivity. *McKinsey Global Institute Research, Tech. Rep*, .

Mendelson, E. B. (2019). Artificial intelligence in breast imaging: Potentials and limitations. *American Journal of Roentgenology*, *212*, 293–299.

Mitchell, T. M. (1997). Does machine learning really work? *AI magazine*, *18*, 11.

Moselhi, O., Hegazy, T., & Fazio, P. (1993a). Dbid: analogy-based dss for bidding in construction. *Journal of Construction Engineering and Management*, *119*, 466–479.

Moselhi, O., Hegazy, T., & Fazio, P. (1993b). Expert: an expert system for strategic bidding. *Proceedings of Annual Conference of Canadian Society for Civil Engineering, Fredericton, NB, Canada*, .

Mullainathan, S., & Spiess, J. (2017). Machine learning: an applied econometric approach. *Journal of Economic Perspectives*, *31*, 87–106.

Ng, A. (2018). *AI Transformation Playbook: How to lead your company into the AI era.* Mcgraw-hill New York.

Petruseva, S., Sherrod, P., Pancovska, V. Z., & Petrovski, A. (2016). Predicting bidding price in construction using support vector machine. *TEM J*, .

Rudin, C., & Vahn, G.-Y. (2014). The big data newsvendor: Practical insights from machine learning, .

Rui, Z., Peng, F., Ling, K., Chang, H., Chen, G., & Zhou, X. (2017). Investigation into the performance of oil and gas projects. *Journal of Natural Gas Science and Engineering*, *38*, 12–20.

Shapiro, A. (2017). Reform predictive policing. *Nature News*, *541*, 458.

Shash, A. A. (1993). Factors considered in tendering decisions by top uk contractors. *Construction management and economics*, *11*, 111–118.

Toshev, A., & Szegedy, C. (2014). Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1653–1660).

Wanous, M., Boussabaine, A., & Lewis, J. (2000). To bid or not to bid: a parametric solution. *Construction Management & Economics*, *18*, 457–466.

Wanous, M., Boussabaine, H. A., & Lewis, J. (2003). A neural network bid/no bid model: the case for contractors in syria. *Construction Management and Economics*, *21*, 737–744.

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann.