Multi-Dimensional Prediction Model for Estimating Software Project Implementation Outcomes from a Client Perspective in the Energy Sector

by Humie Woo

Bachelor of Applied Science, June 2002, University of Toronto
Master of Engineering, November 2012, University of Toronto

A Praxis submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial fulfillment of the requirements
for the degree of Doctor of Engineering

January 7, 2022

Praxis directed by

Rebecca Yassan
Professorial Lecturer of Engineering Management and Systems Engineering

The School of Engineering and Applied Science of The George Washington University certifies that Humie Woo has passed the Final Examination for the degree of Doctor of Engineering as of December 3, 2021. This is the final and approved form of the Praxis.

Multi-Dimensional Prediction Model for Estimating Software Project Implementation Outcomes from a Client Perspective in the Energy Sector

Humie Woo

Praxis Research Committee:

Rebecca Yassan, Professorial Lecturer of Engineering Management and Systems Engineering, Praxis Director

J.P. Blackford, Professorial Lecturer of Engineering Management and Systems Engineering, Committee Member

Amir Etemadi, Associate Professor of Science and Engineering, Committee Member

© Copyright 2021 by Humie Woo
All rights reserved

Dedication

The author would like to dedicate this praxis to her family. To her late father, Patrick, who always encouraged her to pursue higher education. To her mother, Midge, who has always been there for her. To Connie and Milton for their help to give her more time to work on this praxis. Thank you to her loving husband and best friend, James, and their two daughters, Ella and Marissa, for their support and understanding throughout these last two years.

Acknowledgements

The author wishes to acknowledge her advisor, Dr. Rebecca Yassan, for her support and guidance throughout this praxis process.

Abstract of Praxis

Multi-Dimensional Prediction Model for Estimating Software Project Implementation Outcomes from a Client Perspective in the Energy Sector

Increasing complexity in software systems and software project implementations results in 50% of software projects being over budget, late or lacking the required functionality[1], and causes organizations to incur financial losses of millions of dollars. This praxis presents a Multi-Dimensional Prediction Model (MDPM) by developing and comparing four machine learning models (Multiple Linear Regression, Decision Tree, Random Forest, Neural Network) to predict multi-dimensional software project implementation outcomes. The software project data set was obtained from a large-size Canadian organization in the Energy Sector that implements software projects in partnerships with external vendors. 102 project instances were identified for this praxis with ten Critical Success Factors (CSFs) as predictors.

This praxis demonstrates how project sponsors can use the MDPM to support decision-making and cost benefit analysis to reduce the likelihood of failed projects. The final MDPM predicts, within a 20% margin of error, the schedule and cost contingencies required to manage project uncertainties and risks, and the number of system defects required to deliver a quality end product. The top five CSFs with the most significant influence on the output variables were Integration of the System, Project Base Cost, Project Base Schedule, Project Team Capability, and Top Management Support. Random Forest model was selected to be the most effective method in estimating multi-dimensional project outcomes.

[1] The Standish Group. (2020). CHAOS 2020: Beyond Infinity. The Standish Group.

Table of Contents

Dedication ..... **iv**

Acknowledgements ..... **v**

Abstract of Praxis ..... **vi**

Table of Contents ..... **vii**

List of Figures ..... **xii**

List of Tables ..... **xvii**

List of Symbols ..... **xviii**

List of Equations ..... **xix**

List of Acronyms ..... **xx**

List of Glossary of Terms ..... **xxii**

Chapter 1—Introduction 1

1.1 Background 1

1.2 Research Motivation 2

1.3 Problem Statement 3

1.4 Thesis Statement 4

1.5 Research Objectives 4

1.6 Research Questions and Hypotheses 5

1.7 Scope of Research 5

1.8 Research Limitations 6

1.9 Organization of Praxis 7

Chapter 2—Literature Review 8

2.1 Introduction 8

2.2 Critical Success Factors for Software Projects 8
2.2.1 Overview of Critical Success Factors 8
2.2.2 Technical Critical Success Factors 9
2.2.3 Project Management Critical Success Factors 10
2.2.4 Team-Related Critical Success Factors 12
2.2.5 Organizational Critical Success Factors 12
2.2.6 Environmental Critical Success Factors 13
2.3 Dimensions of Software Project Performance Outcomes 14
2.3.1 Overview of Project Performance Outcomes 14
2.3.2 Project Schedule Outcome 15
2.3.3 Project Cost Outcome 15
2.3.4 Project Quality Outcome 17
2.3.5 Project Contingencies 17
2.3.6 Multi-Dimensional Performance Outcomes 19
2.4 Analysis of Feature Importance and Ranking 20
2.5 Machine Learning Model Analysis 22
2.5.1 Background of Machine Learning 22
2.5.2 Multiple Linear Regression Model 23
2.5.3 Decision Tree Model 26
2.5.4 Random Forest Model 26
2.5.5 Neural Network Model 27
2.5.6 Comparison of Machine Learning Models 28
2.6 Analysis of Machine Learning Evaluation Metrics 29

2.7 Summary and Conclusion 33
Chapter 3—Methodology 35
3.1 Introduction 35
3.2 Data Source Selection 36
3.2.1 Data Collection Business Process 36
3.2.2 Critical Success Factors Selection 39
3.2.3 Multi-Dimensional Outputs Selection 41
3.3 Preprocessing of Data sets 41
3.4 Exploratory Data Analysis Methods 43
3.5 Feature Importance and Ranking Method 44
3.6 Applied Machine Learning Methods 46
3.6.1 Multiple Linear Regression 48
3.6.2 Decision Tree 50
3.6.3 Random Forest 51
3.6.4 Neural Network 52
3.7 Data Validation 53
3.8 Model Performance Evaluation Metrics 55
Chapter 4—Results 58
4.1 Introduction 58
4.2 Exploratory Data Analysis Results 58
4.3 Hypothesis 1 and Results 62
4.3.1 Multiple Linear Regression Feature Results 63
4.3.2 Decision Tree Feature Results 66

4.3.3 Random Forest Feature Results 69
4.3.4 Neural Network Feature Results 72
4.3.5 Summary of Feature Results 75
4.4 Hypothesis 2 and Results 76
4.5 Hypothesis 3 and Results 82
4.6 Hypothesis 4 and Results 86
4.7 Hypothesis 5 and Results 90
4.8 Model Validation 94
4.8.1 Multiple Linear Regression Validation Results 95
4.8.2 Decision Tree Validation Results 97
4.8.3 Random Forest Validation Results 99
4.8.4 Neural Network Validation Results 101
Chapter 5—Discussion and Conclusions 103
5.1 Discussion 103
5.1.1 Discussion of Hypothesis 1 103
5.1.2 Discussion of Hypothesis 2 104
5.1.3 Discussion of Hypothesis 3 105
5.1.4 Discussion of Hypothesis 4 106
5.1.5 Discussion of Hypothesis 5 107
5.1.6 Discussion of Model Comparison 107
5.2 Conclusions 109
5.2.1 Lessons Learned 110
5.3 Contributions to Body of Knowledge 111

5.4 Recommendations for Future Research 111

References 113

List of Figures

Figure 2-1. Major Project Development Baselines and Overruns (GAO-21-306, 2021) . 16

Figure 2-2: Integrated Total Project Cost & Schedule 18

Figure 3-1. End-to-end Business Planning Process 37

Figure 3-2. Sample Data Set 39

Figure 3-3. SHAP Value Plots Sample 45

Figure 3-4. Chained Multi-Dimensional Prediction Model 47

Figure 3-5. Correlation Matrix Heatmap Sample 49

Figure 4-1. Descriptive Statistics 59

Figure 4-2. Scatter Plot 60

Figure 4-3. Correlation Matrix Heatmap 61

Figure 4-4. MLR Feature Importance and Ranking Results for the Schedule Contingency Dimension 63

Figure 4-5. MLR Feature Importance and Ranking Results for the Cost Contingency Dimension 64

Figure 4-6. MLR Feature Importance and Ranking Results for the System Defect Dimension 65

Figure 4-7. DT Feature Importance and Ranking Results for the Schedule Contingency Dimension 66

Figure 4-8. DT Feature Importance and Ranking Results for the Cost Contingency Dimension 67

Figure 4-9. DT Feature Importance and Ranking Results for the System Defect Dimension 68

Figure 4-10. RF Feature Importance and Ranking Results for the Schedule Contingency Dimension 69
Figure 4-11. RF Feature Importance and Ranking Results for the Cost Contingency Dimension 70
Figure 4-12. RF Feature Importance and Ranking Results for the System Defect Dimension 71
Figure 4-13. NN Feature Importance and Ranking Results for the Schedule Contingency Dimension 72
Figure 4-14. NN Feature Importance and Ranking Results for the Cost Contingency Dimension 73
Figure 4-15. NN Feature Importance and Ranking Results for the System Defect Dimension 74
Figure 4-16. Summary Plot of Top 5 CSF Count 76
Figure 4-17. MLR Residual plot and histogram plot of residuals for the Schedule Contingency Dimension 77
Figure 4-18. MLR Residual plot and histogram plot of residuals for the Cost Contingency Dimension 78
Figure 4-19. MLR Residual plot and histogram plot of residual for the System Defect Dimension 78
Figure 4-20. MLR Predicted vs. True Value for the Schedule Contingency Dimension . 79
Figure 4-21. MLR Predicted vs. True Value for the Cost Contingency Dimension 79
Figure 4-22. MLR Predicted vs. True Value for the System Defects Dimension 80

Figure 4-23. MLR Residual plot and histogram plot of residual for the Schedule Contingency Dimension 80
Figure 4-24. MLR Residual plot and histogram plot of residual for the Cost Contingency Dimension 81
Figure 4-25. MLR Residual plot and histogram plot of residual for the System Defect Dimension 81
Figure 4-26. DT Predicted vs. True Value for the Schedule Contingency Dimension 83
Figure 4-27. DT Predicted vs. True Value for the Cost Contingency Dimension 83
Figure 4-28. DT Predicted vs. True Value for the System Defects Dimension 83
Figure 4-29. DT Residual plot and histogram plot of residual for the Schedule Contingency Dimension 84
Figure 4-30. DT Residual plot and histogram plot of residual for the Cost Contingency Dimension 85
Figure 4-31. DT Residual plot and histogram plot of residual for the System Defects Dimension 85
Figure 4-32. RF Predicted vs. True Value for the Schedule Contingency Dimension 87
Figure 4-33. RF Predicted vs. True Value for the Cost Contingency Dimension 87
Figure 4-34. RF Predicted vs. True Value for the System Defects Dimension 87
Figure 4-35. RF Residual plot and histogram plot of residual for the Schedule Contingency Dimension 88
Figure 4-36. RF Residual plot and histogram plot of residual for the Cost Contingency Dimension 89

Figure 4-37. RF Residual plot and histogram plot of residual for the Cost Contingency Dimension 89
Figure 4-38. NN Predicted vs. True Value for the Schedule Contingency Dimension 91
Figure 4-39. NN Predicted vs. True Value for the Cost Contingency Dimension 91
Figure 4-40. NN Predicted vs. True Value for the System Defects Dimension 91
Figure 4-41. NN Residual plot and histogram plot of residual for the Schedule Contingency Dimension..... 92
Figure 4-42. NN Residual plot and histogram plot of residual for the Cost Contingency Dimension 93
Figure 4-43. NN Residual plot and histogram plot of residual for the System Defects Dimension 93
Figure 4-44. Data Input Histogram 95
Figure 4-45. MLR Cross Validation Box Plot for the Schedule Contingency Dimension 96
Figure 4-46. MLR Cross Validation Box Plot for the Cost Contingency Dimension 96
Figure 4-47. MLR Cross Validation Box Plot for the System Defects Dimension 97
Figure 4-48. DT Cross Validation Box Plot for the Schedule Contingency Dimension .. 98
Figure 4-49. DT Cross Validation Box Plot for the Cost Contingency Dimension 98
Figure 4-50. DT Cross Validation Box Plot for the System Defects Dimension 99
Figure 4-51. RF Cross Validation Box Plot for the Schedule Contingency Dimension 100
Figure 4-52. RF Cross Validation Box Plot for the Cost Contingency Dimension 100
Figure 4-53. RF Cross Validation Box Plot for the System Defects Dimension..... 101
Figure 4-54. NN Validation Model Loss for the Schedule Contingency Dimension 102

Figure 4-55. NN Validation Model Loss for the Cost Contingency Dimension 102

Figure 4-56. NN Validation Model Loss for the System Defects Dimension 102

List of Tables

Table 3-1. Critical Success Factor Variables 40

Table 3-2. Performance Output Definitions 41

Table 3-3. CSF Encoding Values 43

Table 4-1. Top 5 CSFs for each Dimension and Model 75

Table 4-2. MLR Normality Check P-Value 77

Table 4-3. MLR Model Performance Results 82

Table 4-4. DT Model Performance Results 86

Table 4-5. RF Model Performance Results 90

Table 4-6. NN Model Performance Results 94

List of Symbols

b_0	estimated intercept
b_i	estimated coefficient for the ith predictor
β	parameter estimate
e	random error term
F	set of input features
$f(x)$	model prediction function
λ	regularization penalty term
M	number of input features
N	total number of data points
n	total number of sample instances
P	total number of parameters
ϕ_i	Shapley Additive Explanations value for the ith feature
S	subset of input features
x_i	value of the ith predictor
y_i	actual observed value
$\widehat{y_i}$	predicted value
$\bar{y}$	mean of all historical observed values
y_{max}	maximum value of all data points
y_{min}	minimum value of all data points

List of Equations

Equation 2-1. Shapley Additive Explanations (SHAP) Value (Messalas et al., 2019) 21

Equation 2-2. Sum of the Squared Error 23

Equation 2-3. OLS Linear Regression 24

Equation 2-4. Sum of the Squared Error for L2 Regularization 24

Equation 2-5. Sum of the Squared Error for L1 Regularization 25

Equation 2-6. Mean Absolute Percentage Error (MAPE) 29

Equation 2-7. Root Mean Squared Percentage Error (RMSPE) 30

Equation 2-8. Root Relative Squared Error (RRSE) 30

Equation 2-9. Mean Absolute Error (MAE) 31

Equation 2-10. Root Mean Squared Error (RMSE) 31

Equation 2-11. Relative Absolute Error (RAE) 32

Equation 3-1. Mean Absolute Error (MAE) 56

Equation 3-2. Normalized MAE (NMAE) 56

Equation 3-3. Root Mean Squared Error (RMSE) 56

Equation 3-4. Normalized RMSE (NRMSE) 57

List of Acronyms

AI	Artificial Intelligence
COCOMO	Constructive Cost Model
CPM	Critical Path Method
CSF	Critical Success Factor
EDA	Exploratory Data Analysis
ERP	Enterprise Resource Planning
FSTM	First Step Trained Model
IFS	Independent Feature Set
IT	Information Technology
KDE	Kernel Density Estimation
KNN	K-Nearest Neighbor
LASSO	Least Absolute Shrinkage and Selection Operator
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MDPM	Multi-Dimensional Prediction Model
MSE	Mean Squared Error
NMAE	Normalized Mean Absolute Error
NRMSE	Normalized Root Mean Squared Error
OLS	Ordinary Least Squares
PERT	Program Evaluation and Review Technique
RAE	Relative Absolute Error
RCPSP	Resource-Constrained Project Scheduling Problem

ReLU	Rectified Linear Activation Function
RGT	Run-Grow-Transform Model
RMSE	Root Mean Squared Error
RMSPE	Root Mean Square Percentage Error
RRSE	Root Relative Squared Error
SDLC	Software Development Life Cycle
SHAP	Shapley Additive Explanations
SSE	Sum of the Squared Errors
SVBP	Shapley Additive Explanations Value Bar Plot
SVM	Support Vector Machine
SVSP	Shapley Additive Explanations Value Summary Plot
SSTM	Second Step Trained Model
TSTM	Third Step Trained Model
XAI	Explainable Artificial Intelligence

List of Glossary of Terms

Bias	Bias in machine learning refers to the difference between the predicted value by a machine learning model and the true value. High bias models refer to models that are underfitting.
Black Box Model	A black box model in machine learning refers to algorithms that are created directly from data. These are difficult for humans to understand and interpret.
Bottoms-up Estimate	Bottom-up estimate is a technique that involves estimations at the lowest level of detail. These estimates are then aggregated to create a total base estimate for the project.
Business Case	A business case is a formal document that provides the objectives and goals of the project, the expected cost, benefits and detailed financial analysis.
Client Perspective	A client perspective refers to an organization which seeks professional services and pays a fee for services as agreed in a contract from external vendors.
Construct Validity	Construct validity evaluates whether the measurement tools accurately represent the variables which the model intended to measure.
Contingency Reserve	Contingency reserve is allocated to respond to the known risks and is part of the bottoms-up estimates and work package estimates.
Cost Contingency	Cost contingency is the difference in dollars between the baseline project budget and the actual cost at project completion.
External Validity	External validity refers to how well the outcome of a scientific study can be applied to settings outside the context of the study.
Feasibility Study	A feasibility study is conducted before a business case is created and it determines if a project is advantageous and beneficial for the company to undertake.
Heavyweight Methodology	Heavyweight methodology refers to the traditional plan-based project management approach such as the waterfall model. This methodology requires comprehensive planning and detailed documentation.
Internal Validity	Internal validity is the degree of confidence that the causal relationship being tested is not influenced by confounding variables.

Lightweight Methodology	Lightweight methodology refers to an adaptive project management approach that has short iterative development cycles such as agile and scrum models.
Management Reserve	Management reserve is allocated at a high level for unknown risks and unexpected events.
One-Hot Encoding	One-hot encoding is an encoding method that assigns a new dichotomous dummy variable for each unique value of the categorical variable.
Project Closure Document	The project closure document is the formal handoff from project execution to project sustainment which includes information such as the final cost, final duration, as well as lessons learned.
Project Execution	Project execution refers to the end-to-end implementation of a project from initiation to closure.
Project Failure	Project failure refers to projects that are late, over budget, or delivered with less than the required scope.
Project Launch	Project launch refers to the go-live or the time at which the system becomes available for use. Project launch is the point at which software code moves from the test environment to the production environment.
Project Sponsor	A project sponsor is the person who provides the financial resources and is the decision-maker for the project.
Project Sustainment	Project sustainment is the phase after a project is formally closed and it involves supporting and maintaining the software system.
Schedule Contingency	Schedule contingency is the difference in months between the baseline project estimated schedule and the actual schedule at project completion.
Work Package	A work package is a group of related tasks within a project that is required to produce a deliverable.

Chapter 1—Introduction

1.1 Background

Global enterprise Information Technology (IT) spending is forecast to increase 4.1% in 2021 with a 6.0% five-year compound annual increase over this period to reach $3.9 trillion by 2025 (Agamirzian et al., 2021). The complexity of engineering projects has significantly increased due to the exponential growth of computer systems and software development in the IT industry (Blanchard, 2003). Project managers of these complex software engineering projects are challenged by limited resources, unrealistic stakeholder pressures, and external constraints, especially in the public sector. As a result, these projects experience persistent schedule delays, cost overruns, and software product quality issues.

Research in project failure started in the software industry in 1968 when the term "software crisis" was first introduced (Naur & Randell, 1969). Software development is complex, accounting for the high rate of project failure (Dalcher, 2014). Inaccurate estimation of software project schedule and cost is the main contribution to the failure (Kumari & Pushkar, 2018). Software estimation is a critical initial phase of the software lifecycle process. The objective of this process is to gain insight into the project progress and its deliverables (Asheeri & Hammad, 2019).

Over the past two decades, there has been extensive research on machine learning and artificial intelligence to address the problems of project complexity and success (Mitrovic et al., 2020). Machine learning is a subset of artificial intelligence which is based on the proposition that "systems can learn from analyzed data, recognize patterns and make calculated decisions with minimal or no human interaction needed" (Predescu

et al., 2019, p.76). Machine learning can be an effective tool to address project complexity from project conceptualization to project execution, leading to a positive impact on the overall project performance. By enhancing the estimation models using machine learning, one would have better control over the schedule, budget, and quality for any software project (Kumari & Pushkar, 2018).

1.2 Research Motivation

Over the past ten years, the demand for high quality software products has risen at a very high rate. The expectations for software organizations to improve project management practices, increase productivity, and reduce the time to market has significantly increased (Khan & Mahmood, 2015). The research motivation for this praxis is to identify the Critical Success Factors (CSFs) and develop an improved software project estimation model using machine learning. A machine learning model is needed to accurately predict the required cost contingency, schedule contingency, and number of system defects during the planning stage of a software implementation project.

Cost overruns do not always lead to project failure, but they take monetary resources away from other priority projects (Bouayed, 2016). Bouayed (2016) stated that "in the public sector, cost overruns also translate into loss of public confidence in the way the government manages taxpayers' money" (p.293). One of the common challenges stems from the fact that project promoters routinely omit project costs to gain initial approvals from project sponsors (Guillaume-Joseph & Wasek, 2015). An unbiased and accurate prediction model is required for project managers to estimate cost, time, and quality necessary for implementation of a successful project. Inaccurate and subjective estimation processes mostly result in failure of projects (Kumari & Pushkar, 2018).

A machine learning model can help to predict whether the software project implementation will be successful before financial investments are committed (Guillaume-Joseph & Wasek, 2015). Decision makers require an unbiased prediction model to assist them in their strategic planning and to make informed decisions early in the project lifecycle. This praxis aims to identify the advantages of using machine learning in IT project management and select the optimum CSFs and machine learning model to improve project management performance.

1.3 Problem Statement

Increasing complexity in software systems and software project implementations results in 50% of software projects being over budget, late or lacking the required functionality, and causes organizations to incur financial losses of millions of dollars.

Project failure often stems from "increasing complexity due to system of systems integration, technological advancements, and distributed development" (Ryan et al., 2014, p.10). This increasing complexity results in 50% of software projects being over budget, late or lacking the required functionality (The Standish Group, 2020). Accurate estimation results are required to help project managers perform better prediction of the project cost, project schedule, and the overall product quality (Asheeri & Hammad, 2019).

1.4 Thesis Statement

A multi-dimensional prediction model is required to estimate schedule contingency, cost contingency, and the number of system defects in software implementation projects from a client perspective.

By combining an optimum set of CSFs and machine learning models, this praxis develops a predictive model using machine learning to accurately predict the multi-dimensional outcomes in software implementation projects in the Canadian Energy Sector. This model aims to support project sponsors in their cost-benefit analysis and decision-making process to reduce the likelihood of failed projects.

1.5 Research Objectives

There are three main research objectives in this praxis:

1. Develop a Multi-Dimensional Prediction Model (MDPM) to support project managers and sponsors to make unbiased and informed decisions in the organization's end-to-end business planning process.
2. Identify the optimal set of CSFs that can be effective inputs to the MDPM to accurately predict the cost and schedule contingencies, and the number of system defects for each software implementation project.
3. Develop and compare four machine learning models (Multiple Linear Regression, Decision Tree, Random Forest, Neural Network) using clearly-defined model performance metrics.

1.6 Research Questions and Hypotheses

The following two questions guide this research to address the problem of software projects being over budget, late or lacking the required functionality. In response to these two questions, five hypotheses are proposed.

RQ1: What features are effective predictors of multi-dimensional project outcomes from a client perspective?

RQ2: Which proposed prediction model is the most effective in estimating multi-dimensional project outcomes based on prior project performance data?

H1: The Critical Success Factors (Independent Variables) identified in this praxis have significant influence on the dependent variables.

H2: A Multiple Linear Regression model can be developed to predict multi-dimensional project outcomes with Normalized Mean Absolute Error (NMAE) and Normalized Root Mean Squared Error (NRMSE) to be less than or equal to 20%.

H3: A Decision Tree model can be developed to predict multi-dimensional project outcomes with NMAE and NRMSE to be less than or equal to 20%.

H4: A Random Forest model can be developed to predict multi-dimensional project outcomes with NMAE and NRMSE to be less than or equal to 20%.

H5: A Neural Network model can be developed to predict multi-dimensional project outcomes with NMAE and NRMSE to be less than or equal to 20%.

1.7 Scope of Research

This praxis includes an extensive literature review to understand the CSFs in project management to achieve optimal performance. The data set chosen as potential predictors are CSFs in five categories synthesized from the literature: technical factors,

project management factors, team factors, organization factors and environmental factors. This research draws upon historical data obtained from a Canadian large-size client organization that implements software projects in partnerships with external vendors in the Energy sector. This model can be tuned and applied to a broader industry such as banking, manufacturing, and healthcare. This research contributes to the body of knowledge by exploring prediction techniques that extend to a multi-dimensional output. The model will predict project cost contingency, schedule contingency and the number of system defects in the project planning stage with the objective of reducing the percentage of failed projects. This research explores the impact of the selected CSFs and inputs them into four different machine learning models to predict the multi-dimensional project implementation outcomes. This tool acts as a new step in the end-to-end business planning process to assist project sponsors in their development of a cost-benefit analysis and their decision-making strategies.

1.8 Research Limitations

The data used in this research is limited to the data set obtained from a large-size Canadian organization in the Energy Sector that implements software projects in partnerships with external vendors. 208 projects were identified from 2016 to 2020. Formally defined data validation rules were applied to the data set with 102 projects meeting the criteria and included in this research. This is not considered a large sample size; a larger sample size would allow for more training data in the machine learning models and could potentially provide additional insights in the analysis (Myrtveit et al., 2005; Chu et al., 2012; Cui & Gong, 2018; Kuhn & Johnson, 2018).

The models developed are trained, validated and tested using historical data. However, they have not yet been deployed and evaluated on new projects.

There are many different machine learning techniques in predictive modeling; four machine learning models were selected for this study. These four models represent the most common and most effective techniques in the context of this praxis.

These limitations and boundaries must be considered when applying the model and interpreting the results of the research.

1.9 Organization of Praxis

The remainder of this praxis research is as follows. Chapter two is a review of peer-reviewed literature and discusses key themes such as CSFs, software performance, and machine learning models. Chapter three presents the research methodology that addresses the data set, CSF predictors, multi-dimensional outputs, machine learning methods, and model performance evaluation metrics. Chapter four details key results and observations. Chapter five provides the conclusion that summarizes the contributions to the body of knowledge and recommendations for future research.

Chapter 2—Literature Review

2.1 Introduction

Managing software implementation is challenging due to the complexity, size, and the intangibility of the software itself (Nasir & Sahibuddin, 2011). An innovative and practical tool is required to mitigate this problem. This literature review discusses different prediction models used by researchers to estimate software project implementation outcomes. Recent advances and applications of machine learning models are reviewed. The intent is to analyze relevant academic publications on Critical Success Factors (CSFs), dimensions of software project outcomes, feature importance and ranking methods, and machine learning models.

Section 2.2 explains the CSFs in software project implementation that contribute to software project success. **Section 2.3** describes the different dimensions of software project performance outcomes. **Section 2.4** analyzes feature importance and ranking methods to support the discovery of an optimal feature set. **Section 2.5** reviews the different machine learning models that aid the prediction of software project implementation outcomes. **Section 2.6** considers the machine learning evaluation metrics, and **Section 2.7** is a summary and draws conclusions from the literature review.

2.2 Critical Success Factors for Software Projects

2.2.1 Overview of Critical Success Factors

CSFs are characterized as independent variables that directly influence the success of a project (Kerzner, 2018). The relationship between the CSFs as the independent variables and project outcome as the dependent variable is complex. Many researchers such as Mitrovic et al. (2020), Ahimbisibwe et al. (2015), and Sudhakar

(2012) explored this relationship, both linearly and nonlinearly, when designing their project-outcome prediction models. Creation of prediction models using CSFs as input variables can provide insights to project managers during the planning phase of a software project. “The key to modeling usable project outcome prediction models is to move beyond the limits of easily available data and to conceive of information as it relates to key areas of activity in which favorable results are absolutely necessary for project success” (Mitrovic et al., 2020, p.213622). Project success rates can be improved if organizations concentrate on the important CSFs for software projects (Sudhakar, 2012).

The CSFs for software project implementation synthesized from research journals, books and conference proceedings are discussed in this section. Five categories were identified based on literature review of CSFs for software projects (Ahimbisibwe et al., 2015; Sudhakar, 2012; Nasir and Sahibuddin, 2011):

(1) Technical Factors;

(2) Project Management Factors;

(3) Team Factors;

(4) Organization Factors; and

(5) Environmental Factors.

The following sections explain each factor category and its integral CSFs.

2.2.2 Technical Critical Success Factors

The technical category of CSFs includes technical complexity in a software project and the project’s technical model (Prabhakar, G.P., 2008; Sudhakar, 2012). Project failure stems from complex projects as the need to integrate and develop multiple

software subsystems in a distributed environment continues to increase (Ryan et al., 2014). Software complexity is considered to be the main reason behind project failure (Mitrovic et al., 2020; Kumari & Pushkar, 2018; Nasir and Sahibuddin, 2011). Literature review indicates that this problem can be minimized by addressing complexity, technical uncertainty, and the integration of the system (Mitrovic et al., 2020; Svejvig & Andersen, 2015; Sudhakar, 2012).

There are three types of software projects in Information Technology (IT): (1) "Run" projects maintain essential business processes such as software upgrades, (2) "Grow" projects expand and improve current business processes, and (3) "Transform" projects are new business ideas or processes (Adnams et al., 2018; Agamirzian et. al., 2021). The Run-Grow-Transform (RGT) is the technical model used to address complexity and uncertainty, and support the implementation of software projects (Adnams et al., 2018; Agamirzian et. al., 2021). The RGT model acts as a simplification tool to aid project managers and sponsors to make decisions to improve project performance outcomes (Adnams et al., 2018).

2.2.3 Project Management Critical Success Factors

There are two main delivery methodologies in the research of project management: (1) the traditional plan-based waterfall method, and (2) the agile method (Shawky, 2014; Chow and Cao, 2008; Highsmith, 2013). The traditional waterfall method was invented by Royce in 1970 (Sommerville, 1996), and it has become the standard methodology for many software development projects. The waterfall method is considered a heavyweight approach because it requires comprehensive planning and detailed documentation. The standard practice for waterfall projects follows the Software

Development Life Cycle (SDLC) which is divided into seven stages including "conception, initiation, analysis, design, construction, testing, and maintenance" (Shawky, 2014, p.109). The waterfall model is the standard framework for large and complex engineering projects. It is an effective method for projects where customer requirements can be identified up front; however, as it is a detailed plan-based approach, it is less flexible in handling changes and complexities (Highsmith, 2013).

By contrast, the agile methodology embraces complexity and the higher rates of change (Shawky, 2014). The agile methodology is considered a lightweight approach because it employs short iterative cycles and small incremental deliverables designed to be more flexible and to emphasize the importance of continuous improvement in the SDLC (Shawky, 2014). The agile method is considered the standard framework for small to medium-sized software projects where the main deliverable can be broken down into incremental releases (Highsmith, 2013). The choice of delivery methodology is critical in the success of a software project.

Nasir & Sahibuddin (2011), Ahmed et al. (2008), Chow & Cao (2008), and Suliman & Kadoda (2017) demonstrated that project base schedule and project base cost are two CSFs that should be considered as key project management CSFs. In the planning phase of a project, bottoms-up estimations of schedule and cost are required (Chen et al., 2016). Project managers start at the activity-level estimates, which are the lowest level of detail, and these estimates are aggregated to create the work-package-level estimate and finally, the total project-base-level estimate (Chen et al., 2016). Arbitrary and illogical schedule and cost estimations due to upper management pressure are the top contributors to project failure (Nasir & Sahibuddin, 2011). Accurate schedule

and cost estimations are crucial to project success as resource allocations are directly dependent on the estimates provided by project managers (Ahmed et al., 2008).

2.2.4 Team-Related Critical Success Factors

Team Factors that relate to project team expertise, experience, and composition have a positive impact on software project success (Tam et al., 2020; Chow & Cao, 2008; Fayaz et al., 2017). As indicated by Tam et al. (2020), a highly capable team delivers software that focus on the quality of product and on customers' requirements. Technical competence and expertise, in addition to proper provision of technical training, is critical to the success of agile projects (Chow & Cao, 2008). Training is one of the most often quoted CSFs in the implementation of IT projects success in academic publications (Fayaz et al., 2017). Technical expertise that is supported by training and learning, allows project teams to deal with risks better, and improve the project performance outcomes (Ahimbisibwe et al., 2015). Training and learning refer to skills development, continuous improvement, and sharing of knowledge that directly influence the success of a project (Misra et al., 2009). Training is an important CSF especially for projects that employ agile methodology; teams must be properly trained to follow agile best practices (Dikert et al., 2016). Project team capability, and training and education have a significant effect on IT project success.

2.2.5 Organizational Critical Success Factors

Organizational CSFs include factors that are affected by business strategic direction, top-level management support, and organizational culture (Jung et al., 2008; Ahimbisibwe, et al., 2015; Imreh & Raisinghani, 2011; Mansor et al., 2011). Based on literature review, top-level management support is considered to be the primary success

factor (Jung et al., 2008; Ahimbisibwe, et al., 2015). Ahimbisibwe, et al. (2015) identified 37 CSFs for both agile and traditional software projects from an empirical study of 148 publications on software development. Ahimbisibwe, et al. (2015) identified top management support as the highest ranked CSF. Project will not finish successfully without commitment from top-level management (Imreh & Raisinghani, 2011; Mansor et al., 2011). In the latest publication by the Standish Group (2020), stakeholders and executive project sponsors were newly added as CSFs. Project success requires sustained top management commitment to provide the necessary resources, leadership, and influence.

2.2.6 Environmental Critical Success Factors

Vendor partnerships in a software implementation project are critical to the success of the project (Nasir and Sahibuddin, 2011; Elragal & Haddara, 2013). Research has indicated that an effective and compatible partnership with software vendors is required to implement software projects successfully (Elragal & Haddara, 2013). In complex project implementations such as in an Enterprise Resource Planning (ERP), procuring an excellent implementation vendor is the most important criterion for a successful project (Elragal & Al-Serafi, 2011). In the context of software projects implemented in partnerships with external vendors, vendor partnership becomes an extremely important CSF.

The energy sector in Canada is subject to strict regulation governing their operations and financial spending due to the fact that electricity and natural gas transmission and distribution in Canada are considered to be a monopoly (Canada Energy Regulator, 2019). Laws and regulations are designed to protect the interests of the

consumers (Canada Energy Regulator, 2019). Therefore, external constraints and regulations are particularly influential to the success of any software projects in the Canadian Energy Sector.

2.3 Dimensions of Software Project Performance Outcomes

2.3.1 Overview of Project Performance Outcomes

A review of the software project management literature indicates that there are many ways to define and measure project performance and project success (Ika, 2009). Criteria for project success often differ from one software project to another. Project success criteria are commonly defined by project timeliness, cost, scope, and quality (Ahimbisibwe et al., 2015). Project performance describes how well the project planning and project management processes have been performed and are evaluated based on whether a project is delivered on time, on budget and within scope and quality (Jun et al., 2011). The Project Management Institute (Project Management Institute, 2017) identifies the triple constraints in project management as time, cost, and scope. On-time and on-cost deliverables refer to a software project meeting its performance goals for schedule and budget, respectively. Project scope refers to the specific features and functions required to deliver a product or service (Jun et al., 2011). While fulfilling the scope, quality of the delivered scope is instrumental as it describes whether the end-product is functioning as designed without major defects (Jun et al., 2011).

In the area of project management, studies have focused on predicting one dimension of project performance. In the next sections, each dimension of the project performance outcome is discussed. In addition, this review investigates schedule and cost

contingencies required for project success, and broadens the project performance prediction research into multiple dimensions.

2.3.2 Project Schedule Outcome

In software project implementation literature, meeting project schedule is an important criterion of project success (Chen & Zhang, 2013). Khan & Mahmood (2015) focused their research on schedule estimation, and indicated that schedules require adequate contingencies in order for developers to deliver a quality product. There are many project management tools and techniques available for scheduling and staffing management. Traditional techniques such as the Program Evaluation and Review Technique (PERT) (Malcolm et al., 1959), the Critical Path Method (CPM) (Shtub et al., 2005), and the Resource-Constrained Project Scheduling Problem (RCPSP) (Brucker et al., 1999) model have commonly been used in software project schedule planning in the past. Although these traditional techniques are "important and helpful, they are increasingly considered to be inadequate for modeling the unique characteristics of today's software projects" (Chen & Zhang, 2013, p.1). Advances in machine learning methods to predict the project schedule significantly increase the likelihood of success (Khan & Mahmood, 2015; Chen & Zhang, 2013). Project schedule as a performance outcome is important; however, decision-making based on the schedule dimension alone as an outcome is not sufficient.

2.3.3 Project Cost Outcome

Predictive models are trained with historical data and aim to improve various product performance indices. Mittas & Angelis (2013) and Asheeri & Hammad (2019) designed cost estimation models using machine learning methods in their research. In a

detailed analysis of various historical projects, Flyvbjerg (2014) indicated that project cost overrun is an ongoing challenge in both the public and private sectors around the world. Bouayed (2016) also demonstrated that cost overruns are common especially on government projects. In an assessment report on United States government major projects, Government Accountability Office (2021) demonstrated that cost overruns continued to occur in recent years. **Figure 2-1** illustrates the overruns in United States government major projects from 2012 to 2021.

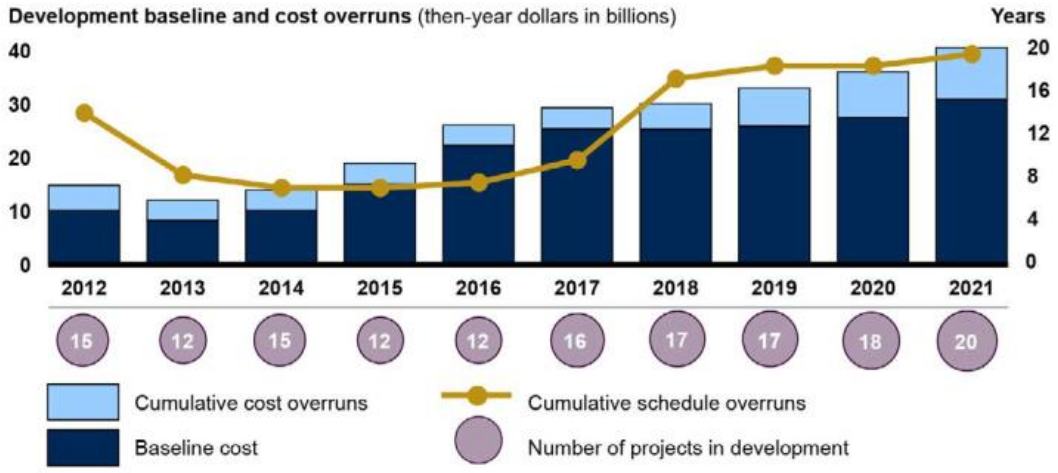

Figure 2-1. Major Project Development Baselines and Overruns (GAO-21-306, 2021)

Project cost as a performance outcome is crucial and is often the highest priority in an organization (Asheeri & Hammad, 2019). For cost estimation, the traditional method commonly used in project management is the Constructive Cost Model (COCOMO) method (Boehm, 2000). With the recent advances in Artificial Intelligence (AI), researchers have demonstrated that machine learning methods can outperform the traditional methods, and are considered to be the preferred application to improve project cost estimation (Mittas & Angelis, 2013; Asheeri & Hammad, 2019). However, assessing the cost dimension alone is not sufficient.

2.3.4 Project Quality Outcome

Projects that are on-time and on-budget but contain many system defects are not considered successful projects. One of the important performance outcomes must aim to identify and fix system defects in the early stages of the project software lifecycle to achieve defect-free software (Jun et al., 2011). Research in software quality estimation considers topics such as defect identification, defect remediation, and testing estimation (Pushphavathi, 2017). Defects occur when actual results deviate from the expected results in a software system, and defects can have varying degrees of complexity and severity (Yusop, 2015). A centralized software defect repository is required for effective defect management which is a critical component of good software engineering practice (Yusop, 2015).

Software defects refer to both product and process defects identified throughout the software project lifecycle (Pushphavathi, 2017). Defects can be identified at the requirement analysis phase; they can also be design flaws or implementation errors. The ability to predict the number of software defects prior to software implementation directly affects the quality of the end product (Pushphavathi, 2017). The quality as a performance outcome helps to ensure that projects achieve conformance to the quality standard at the delivery of the system or product (Leon et al., 2018). Project success cannot be defined only by project timeliness and cost; scope and quality are equally important.

2.3.5 Project Contingencies

Another approach of assessing project cost and schedule as performance outcomes is to consider the cost and schedule contingencies allocated in a project. A project will not be successful if not enough contingencies are estimated at the beginning of the project

(Chen et al., 2016). The ability to predict performance variances and contingencies at completion is necessary in order to provide early indicators of expected project performance results (Chen et al., 2016).

There are always risks and uncertainties when project managers are estimating and planning a project. A contingency reserve is necessary to manage both the cost and schedule uncertainties during the SDLC (Hammad et al., 2015). An estimate with insufficient contingencies will jeopardize the success of the project leading to cost overrun, schedule overrun, and reduced quality (Hammad et al., 2015). An estimate with excessive contingencies represents potential missed opportunities, and funds which cannot be used on other projects (Bouayed Z., 2016). The Association for the Advancement of Cost Engineering (AACE) (2000) defines contingency as "an amount added to the estimate to achieve a specific confidence level", and to allow for changes that will be required based on expert knowledge. **Figure 2-2** is an example of an integrated estimate that includes the bottoms-up estimate and the contingency reserve.

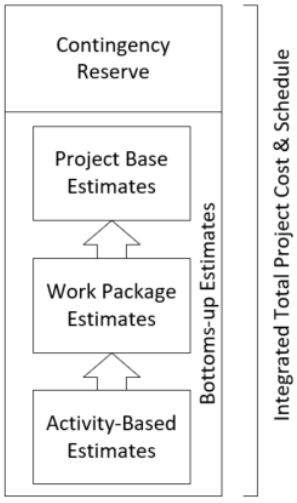

Figure 2-2: Integrated Total Project Cost & Schedule

Contingency is a vital input in an estimate and should be clearly presented as a separate item. There are different methods used to estimate contingencies in project management. The percentage approach is the simplest traditional method, where the contingency is calculated as a percentage addition to the total base estimate (Baccarini, 2005). A fixed contingency percentage is the most common method but it is overly simplistic and does not take into account explicitly the underlying project risks (Baccarini, 2005). An alternative approach is to calculate contingency percentages based on each activity-based estimate, which recognizes that different activities of a project have different risks and uncertainties (Hammad et al., 2015). However, these two traditional methods using percentages imply a degree of certainty that is not justified and are not sufficient as contingency estimators (Bouayed Z., 2016). Barraza & Bueno (2007) and Hammad et al. (2015) attempted to use Monte Carlo simulations to estimate the required cost contingencies and proved that their methods are more effective than the traditional percentage approach. Contingencies have to be properly estimated and must be carefully controlled during project execution. However, no estimate can be correct in every detail, and as a project manager, it is a difficult task to determine the right level of detail for each project estimation (Barraza & Bueno, 2007). In order to address this challenge, AI and machine learning methods using observed and empirical data are needed to estimate contingencies to reduce the likelihood of failed projects.

2.3.6 Multi-Dimensional Performance Outcomes

Compared to traditional machine learning predictions for a single dimensional output, multi-output learning provides a more comprehensive prediction and can solve more complex decision-making problems (Xu et al., 2019). The goal of multi-

dimensional learning is to predict multiple outputs simultaneously given a set of input features or CSFs (Xu et al., 2019; Zhang and Zhou, 2014). This is an important learning problem as "making decisions in the real world often involves multiple complex factors and criteria" (Xu et al., 2019, p.1). The increasing demand of complex decision-making tasks has led to the requirement of multiple outputs and complex structures (Borchani et al., 2015). Therefore, predicting multiple dimensions of project performance outcomes allow project managers and sponsors to make more informed decisions, and directly increases the likelihood of project success.

2.4 Analysis of Feature Importance and Ranking

In machine learning, feature importance ranking is the process of ordering features by the value of feature-importance and the individual feature's predictive power (Lundberg & Lee, 2017). Based on the results from the ranking process, feature selection is carried out to find the optimal feature subset as input variables for the machine learning model in the testing phase. Feature selection is a critical step in the development of any machine learning model as it identifies and removes the irrelevant features in order to maximize the performance of a machine learning model (Wojtas & Chen, 2020). In addition, feature importance ranking is a common way to interpret a machine learning model as it calculates and displays the contribution of each feature to the model prediction. It is also a powerful tool for Explainable Artificial Intelligence (XAI).

XAI is an emerging research area that aims to help users and developers of machine learning models understand the behavior of the models (Saarela & Jauhiainen, 2021). Feature importance ranking performs the discovery of an optimal feature subset and ranks the importance of those features simultaneously (Wojtas & Chen, 2020). XAI

helps to create trust and transparency in the decision-making process when comparing different machine learning models (Messalas et al., 2019). According to Saarela & Jauhiainen (2021) and Fryer et al. (2021), feature importance ranking using Shapley Additive Explanations (SHAP) has become one of the most popular explanation techniques for interpretable machine learning models in recent years.

Derived from cooperative game theory, SHAP is a mathematically well-grounded measure for feature importance (Roth, 1988; Bowen & Ungar, 2020). "SHAP assigns each feature an importance value for a particular prediction to compute the explanation" as the unified measure of additive feature attributions (Messalas et al., 2019, p.2). **Equation 2-1** is the formula for SHAP value denoted as ϕ.

$$\phi_i = \sum_{S \epsilon F \backslash \{i\}} \frac{|S|!\,(M - |S| - 1)!}{M!} [f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S)]$$

Equation 2-1. Shapley Additive Explanations (SHAP) Value (**Messalas et al., 2019)**

where

ϕ_i is the SHAP value for the ith feature,

S is a subset of input features,

F is the set of input features,

M is the number of input features,

f(x) is the model prediction output,

$f_{S \cup \{i\}}(x_{S \cup \{i\}})$ is the model prediction output when the ith feature is present, and

$f_S(x_S)$ is the model prediction output when the ith feature is withheld.

SHAP provides simple, accurate, and easy-to-interpret explanations assigning an importance value for each feature (Messalas et al., 2019). It enables the identification

and prioritization of features during the machine learning training phase (Rodriguez-Perez & Bajorath, 2020).

2.5 Machine Learning Model Analysis

2.5.1 Background of Machine Learning

Literature review suggests that it is increasingly important to embed machine learning models into predictive project management research (Lundberg & Lee, 2017; Predescu et al., 2019). Machine learning models are currently used to assist in medical diagnosis (Khanna & Das, 2020), recruit employees (Asiedu et al., 2017), detect cybersecurity threats (Garces et al., 2019), and to assess outcomes in criminal trials (Mitchell et al., 2020). It is crucial to be able to interpret the output of a prediction model as it gains "appropriate user trust, provides insight into how a model may be improved, and supports understanding of the process being modeled" (Lundberg & Lee, 2017, p.1). Simple models such as Linear Regression are often preferred due to the simplicity and the ease of interpretation, even though the more complex machine learning models could potentially have higher prediction accuracies (Lundberg & Lee, 2017). The growing availability of big data, however, has made it possible for organizations to develop more complex models such as deep learning and Neural Networks. Many organizations are incorporating big data analytics and advanced technologies in their decision-making processes (Maroufkhani et al., 2019).

The primary challenge facing project managers is to deliver projects on-time, on-budget, and with quality within the given constraints. Machine learning algorithms provides a solution to the challenge as they simplify the project management estimation process (Predescu et al., 2019). Machine-learning algorithms take advantage of the

structure in the data set to make estimation predictions without having to understand the underlying statistical model (Broniatowski & Tucker, 2017). Researchers assess these algorithms based on their demonstrated predictive power against new data, despite the fact that some complex algorithms such as Neural Networks are difficult to understand and interpret (Broniatowski & Tucker, 2017).

The four machine learning methods researched in this praxis include three classic machine learning algorithms, Multiple Linear Regression, Decision Tree, and Random Forest, and one deep learning algorithm, Neural Network.

2.5.2 Multiple Linear Regression Model

Regression techniques are simple and they estimate project performance in numerical value based on the relative weightings of input features (Gemino et al., 2010). Tiwana and Keil (2004) proved that regression modelling can be effective in identifying the performance of software development projects in his research with a sample size of 720 software project assessments in 60 large technology companies. Linear regression uses the Ordinary Least Squares (OLS) method to minimize the Sum of the Squared Errors (SSE) between the observed and predicted values (Kuhn & Johnson, 2018). **Equation 2-2** is the formula for SSE and **Equation 2-3** is a form of OLS Linear Regression.

$$\mathrm{SSE} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Equation 2-2. Sum of the Squared Error

where

SSE is the Sum of the Squared Error,

n is total number of samples,

y_i is the observed value, and

$\hat{y}_i$ is the predicted values.

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_i x_i + e$$

Equation 2-3. OLS Linear Regression

where

$\hat{y}$ represents the Linear Regression prediction value,

b_0 represents the estimated intercept,

b_i represents the estimated coefficient for the ith predictor,

x_i represents the value of the ith predictor, and

e represents the random error term.

OLS Linear Regression estimates parameter values that have minimum bias; an alternative to OLS Linear Regression is the Ridge and Least Absolute Shrinkage and Selection Operator (LASSO) regression where they find estimates that have lower variance (Kuhn & Johnson, 2018). Kuhn & Johnson (2018) stated that in the event when the OLS model overfits the data, a penalty term can be added to the SSE in order to control and regularize the estimated parameter. Ridge regression (Hoerl & Kennard, 2000) adds a penalty term to the sum of the squared regression parameters and this regression is denoted as the L2 regularization, which signifies that a second-order penalty is being used on the parameter estimates. **Equation 2-4** is the formula for L2 SSE.

$$\mathrm{SSE(L2)} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{P} \beta_j^{\,2}$$

Equation 2-4. Sum of the Squared Error for L2 Regularization

where

SSE(L2) is the L2 regularization (Ridge) Sum of the Squared Error,

n is total number of samples,

y_i is the observed value,

$\widehat{y_i}$ is the predicted values,

P is the total number of parameters,

λ is the regularization penalty term, and

β is the parameter estimate.

Another regularization method is the LASSO model (Tibshirani, 1996). LASSO uses a similar penalty term to ridge regression, but LASSO takes the absolute value of the penalty term. LASSO is also called the L1 regularization method. **Equation 2-5** is the formula for L1 SSE.

$$\mathrm{SSE(L1)} = \sum_{\mathrm{i=1}}^{\mathrm{n}} (\mathrm{y_i} - \widehat{\mathrm{y_i}})^2 + \lambda \sum_{\mathrm{j=1}}^{\mathrm{P}} |\beta_j|$$

Equation 2-5. Sum of the Squared Error for L1 Regularization

where

SSE(L1) is the L1 regularization (LASSO) Sum of the Squared Error,

n is total number of samples,

y_i is the observed value,

$\widehat{y_i}$ is the predicted values,

P is the total number of parameters,

λ is the regularization penalty term, and

β is the parameter estimate.

While the LASSO regression may seem to be only a small modification to the ridge regression, the practical implications are significant (Kuhn & Johnson, 2018). Taking

the absolute value of the penalty term in a L1 regularization will cause some parameters to be set to 0 (Kuhn & Johnson, 2018). LASSO is effective in feature selection where less important features will be removed.

2.5.3 Decision Tree Model

Decision Tree is another common machine learning technique. It is an intuitive tool that can be used for both regression and classification data sets. The structure consists of the root node at the top of the Decision Tree, and it expands into one or more levels of leaf nodes which contain all possible outcomes called the decision attributes (Tishya et al., 2019). The Decision Tree is constructed recursively by evaluating splitting rules based on maximizing the information gained from the data set. During the training phase of the machine learning process, the knowledge learned from the data set can be formulated into a visual hierarchical structure which is easy to interpret by experts and non-experts (Tishya et al., 2019). It is important for developers to control the maximum depth of the tree to avoid overfitting in the Decision Tree model and avoid noise in the training data. Gemino et al. (2010) analyzed Decision Tree as a modeling technique to predict IT project performance using a sample size of 440 IT projects, and demonstrated that Decision Tree can provide higher predictive accuracy when compared with regression using their sample data set.

2.5.4 Random Forest Model

Random Forest is an ensemble modeling approach constructed on multiple independent Decision Trees (Asheeri & Hammad, 2019). Ensemble methods are machine learning techniques that combine several base learning algorithms with the aim to create a more optimal predictive model. Ensemble models often perform better than a

single learning algorithm, but they are more complex and harder to interpret for users and developers. According to Pospieszny et al. (2018), ensemble models are robust for handling outliers and noises in the data set, and can prevent overfitting. In the experiment performed by Asheeri & Hammad (2019), two public data sets were used to predict software project costs, and they concluded that Random Forest is the most effective technique with a very low error rate.

2.5.5 Neural Network Model

Neural network is a deep learning algorithm that models the information-processing structure of the human brain and nervous system (Mitrovic et al., 2020). One of the benefits of a Neural Network algorithm is "their ability to capture the underlying patterns of available data sets and model complex relationships between input and output variables, while not having a complete understanding of the complexity of functional relationships variables" (Mitrovic et al., 2020, p.213621). A Neural Network structure has an input layer, an output layer, and one or more hidden layers. Neural Network requires a lot of training data and computational power, as it applies a trial-and-error process and requires many iterations to find the optimal result (Mitrovic et al., 2020). Advances in big data analytics have made Neural Network extremely popular in software project management in recent years (Costantino et al., 2015). However, one of the drawbacks of using deep learning is its interpretability and traceability as it is considered a black box model due to the multi-layer and non-linear structure (Wojtas & Chen, 2020). A black box model in machine learning refers to algorithms that are complex and difficult for humans to understand and interpret how predictions are made (Messalas et al., 2019).

2.5.6 Comparison of Machine Learning Models

Each machine learning model has its advantages and disadvantages. There are many competing criteria when comparing machine learning models. Evaluating the models based only on model performance and accuracy is not sufficient. Other concerns that must be considered are interpretability, cost, and maintainability. It is also important to compare different machine learning models consistently. A fair comparison ensures each model is evaluated in the same way on the same set of data.

Many researchers select Linear Regression algorithm due to its ease of implementation, model's training efficiency, and the various analytical software tools for this model (Gemino et al., 2010). Regression techniques require full information including all dependent or independent data, and assumptions such as homoscedasticity and lack of multicollinearity must be met (Gemino et al., 2010). Similar to Linear Regression, a Decision Tree model is also considered a simple model and the tree structure can be easily visualized and interpreted. In addition, the Decision Tree method does not require extensive data preparation such as data normalization. However, similar to Linear Regression, it does not support missing data. Another disadvantage of the Decision Tree model is that it tends to overfit and create a tree with a large depth that does not generalize well (Tishya et al., 2019). The overfitting problem can be reduced by training multiple Decision Trees in an ensemble learner such as the Random Forest, where the features are randomly sampled with replacement (Pospieszny et al., 2018).

The ability to interpret how a model made a certain prediction is important in machine learning because it helps to gain trust, transparency and accountability (Messalas et al., 2019). Complex machine learning algorithms, such as Random Forests and Neural

Networks are considered black box models, which often have high accuracy scores, but they lack interpretability (Messalas et al., 2019). Accuracy versus interpretability is a fundamental trade-off in machine learning research (Messalas et al., 2019). The black box nature of these complex models allows for powerful predictions, but it is very challenging to understand the internal mechanism of these algorithms (Adadi & Berrada, 2018). This challenge has prompted a new debate and research field on XAI which promises to improve trust and transparency, and aims to explain to human subject matter experts the underlying decisions made by the machine learning and AI algorithms (Adadi & Berrada, 2018; Saarela & Jauhiainen, 2021).

2.6 Analysis of Machine Learning Evaluation Metrics

Evaluation metrics explain the performance of a machine learning model. Studies suggest that conclusions from machine learning model comparisons are often dependent on the chosen accuracy evaluation metrics (Myrtveit et al., 2005). Each evaluation metric weighs the importance of characteristics differently and the choice of metric ultimately influences the final selection of the model. There are a wide range of metrics used for classification and regression data sets. Chen et al. (2003) evaluated Mean Absolute Percentage Error (MAPE) presented in **Equation 2-6** and Root Mean Square Percentage Error (RMSPE) presented in **Equation 2-7** when performing sales forecasts in the recreation industry.

$$\text{MAPE} = \frac{1}{n} \times \sum_{i=1}^{n} \left(\frac{|y_i - \hat{y}_i|}{y_i}\right) \times 100\%$$

Equation 2-6. Mean Absolute Percentage Error (MAPE)

where

MAPE is the Mean Absolute Percentage Error,

n is total number of sample instances,

y_i is the actual observed value, and

$\widehat{y_i}$ is the predicted values.

$$\text{RMSPE} = \sqrt{\frac{\sum_{i=1}^{n}(\frac{|y_i - \widehat{y_i}|}{y_i})^2}{n}} \times 100\%$$

Equation 2-7. Root Mean Squared Percentage Error (RMSPE)

where

RMSPE is the Root Mean Squared Percentage Error,

n is total number of sample instances,

y_i is the actual observed value, and

$\widehat{y_i}$ is the predicted values.

Myrtveit et al. (2005) employed Root Relative Squared Error (RRSE) presented in **Equation 2-8** to assess performance of prediction models for software engineering projects.

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

Equation 2-8. Root Relative Squared Error (RRSE)

where

RRSE is the Root Relative Squared Error,

n is total number of sample instances,

y_i is the actual observed value,

$\widehat{y_i}$ is the predicted values, and

$\bar{y}$ is the mean of all historical observed values.

Evaluating prediction accuracy is a difficult task as it does not exist as a one-size-fits-all metric. In the more recent literature, Predescu et al. (2019) evaluated the performance of the software estimation models by calculating Mean Absolute Error (MAE) presented in **Equation 2-9** and Root Mean Squared Error (RMSE) in **Equation 2-10**.

$$\mathrm{MAE} = \frac{1}{n} \times \sum_{i=1}^{n} |y_i - \widehat{y_i}|$$

Equation 2-9. Mean Absolute Error (MAE)

where

MAE is the Mean Absolute Error,

n is total number of sample instances,

y_i is the actual observed value, and

$\widehat{y_i}$ is the predicted values.

$$\mathrm{RMSE} = \sqrt{\frac{1}{\mathrm{n}} \sum_{\mathrm{i}=1}^{\mathrm{n}} (\mathrm{y_i} - \widehat{\mathrm{y_i}})^2}$$

Equation 2-10. Root Mean Squared Error (RMSE)

where

RMSE is the Root Mean Squared Error,

n is total number of sample instances,

y_i is the actual observed value, and

$\widehat{y_i}$ is the predicted values.

Asheeri & Hammad (2019) used similar metrics to assess their software cost estimation algorithms performance including MAE, RMSE, and also included Relative Absolute Error (RAE), and RRSE. RAE is presented in **Equation 2-11**. Xu et al. (2019) studied

the feasibility of machine learning producing multiple outputs, and highlighted that the MAE and RMSE performance metrics are effective metrics for multiple-output models.

$$\mathrm{RAE} = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{\sum_{i=1}^{n} |y_i - \bar{y}|}$$

Equation 2-11. Relative Absolute Error (RAE)

where

RAE is the Relative Absolute Error,

n is total number of sample instances,

y_i is the actual observed value,

$\hat{y}_i$ is the predicted values, and

$\bar{y}$ is the mean of all historical observed values.

MAPE and RMSPE are often used by researchers as evaluation metrics to measure forecast accuracy. These two metrics calculate the average of the percentage errors and they measure how far off the model's predictions are from their corresponding outputs (Chen et al., 2003). One drawback of these metrics is that when there are high errors during periods when actual outputs are low, these metrics will be skewed and will significantly impact the evaluation results (Chen et al., 2003).

The RAE is expressed as a ratio, comparing the mean predicted error to errors generated by a naïve model which sets the forecast to be the average of all historical values (Asheeri & Hammad, 2019). RRSE is similar to RAE but it takes the square root of the total squared error and divides it by the total squared error from the average of the actual values (Myrtveit et al., 2005). By expressing RAE and RRSE as a ratio, the error becomes normalized and can be compared among other models whose errors are measured in the different units.

MAE and RMSE are the most commonly-used evaluation metrics (Predescu et al., 2019; Asheeri & Hammad, 2019; Xu et al., 2019). MAE measures the absolute average magnitude of error produced by the machine learning model. RMSE is very similar to MAE, but takes the square root of the average squared error. RMSE is more sensitive to the outliers as it penalizes the higher errors when compared to MAE (Asheeri & Hammad, 2019). These two metrics are considered standard in machine learning evaluation; however, these two metrics are not scaled to the average error and the metric unit is specific to the output variables (Predescu et al., 2019; Asheeri & Hammad, 2019). These metrics become less effective when comparing different machine learning models. Normalizing the MAE and RMSE metrics make them unitless and can help researchers to compare prediction accuracies between data sets or models with different scales.

2.7 Summary and Conclusion

The complexities of software systems and software implementations have been a primary cause of project failure (Mitrovic et al., 2020). Software planning needs to leverage advanced technologies such as AI and machine learning to adapt to the rapidly changing market, technologies, and customer needs (Agamirzian et. al., 2021). IT project performance is critical to meeting business strategic objectives and to the financial success of an organization. A robust and reliable prediction model to estimate cost, time, and quality of a software project is needed to improve project success rates.

Accurate estimation management and project performance predictions provide project sponsors with important insights to assist them in their decision making early in the project planning cycle (Bouayed Z., 2016). Predictive analysis tools help to identify risks and mitigation strategies early, and recognize the need to cancel a project that is

predicted to fail (Guillaume-Joseph & Wasek, 2015). This praxis emphasizes the importance of early detection and creates a framework that will support project managers and sponsors to make informed decisions before a business case is approved to start project execution. Predictions for project cost contingencies, schedule contingencies, and the number of system defects need to be accurately estimated.

Identification of software CSFs and development of predictive machine learning models are required to produce accurate project performance predictions. The application of machine learning models with well-defined CSFs is a solution for predicting software project outcomes. The rapid development of machine learning algorithms in project management has fundamentally changed the way project managers run and execute projects (Predescu et al., 2019). There are different techniques in predictive modeling; each technique has its benefits and limitations, and no predictive technique excels in all dimensions. Therefore, when comparing these models, it is important to choose the applicable metrics and criteria to evaluate the models.

Most research in the area of software project prediction focuses on predicting a single dimension of software project performance such as project cost outcome or project schedule outcome. This praxis intends to fill this gap by developing a reliable and trustworthy model from a multi-dimensional perspective. Significant value can be realized with an unbiased and rigorous project outcome prediction tool to account for the required cost and schedule contingencies, and the quality of the software, in order to create a positive impact to project success.

Chapter 3—Methodology

3.1 Introduction

This chapter describes in detail the methodology used to study the two research questions and test the hypotheses of this praxis. The development of a prediction model to effectively estimate the cost contingency, schedule contingency, and the number of system defects in projects that are implemented in partnerships with external vendors will be described. A summary of the source data set used, description of the data pre-processing and validation steps required to conduct the data analysis will be discussed. The critical success factors (CSFs) and the multi-dimensional output variables selected for the praxis are defined, and the four machine learning methods employed to test the hypotheses are examined. The four machine learning models are programmed in Python version 3.7.12 code using the Google Co-laboratory development environment. Lastly, the equations used to evaluate the model performance are explained.

Section 3.2 discusses the data source selection including the selection of critical success factors and the multi-dimensional outputs. **Section 3.3** reviews the pre-processing steps of the data sets. **Sections 3.4** presents the exploratory data analysis methods. **Section 3.5** presents the feature importance and ranking method. **Section 3.6** details the machine learning methods including Multiple Linear Regression, Decision Tree, Random Forest, and Neural Network. **Section 3.7** reviews the data validation, and **Section 3.8** discusses the different model performance evaluation metrics.

3.2 Data Source Selection

3.2.1 Data Collection Business Process

Figure 3-1 represents the end-to-end business planning process within an organization that integrates the data collection steps and the prediction model into the overall decision-making. The process starts with a pipeline of ideas and proposed initiatives. After conducting a feasibility study, a set of business cases is created with defined project objectives, project base estimates, project methodology, stakeholders, and external constraints. The output of a business case serves as the input and as the CSFs required for the prediction model. The output for the prediction model is the multi-dimensional project performance including the schedule contingency, cost contingency, and system defects that will be integrated into the baseline business case. This multi-dimensional output feeds into the cost-benefit analysis allowing decision makers to make more informed decisions. Project execution and sustainment are the final two steps of the process. The outputs from these two steps serve as new historical data in the continual learning of the prediction model via a feedback loop.

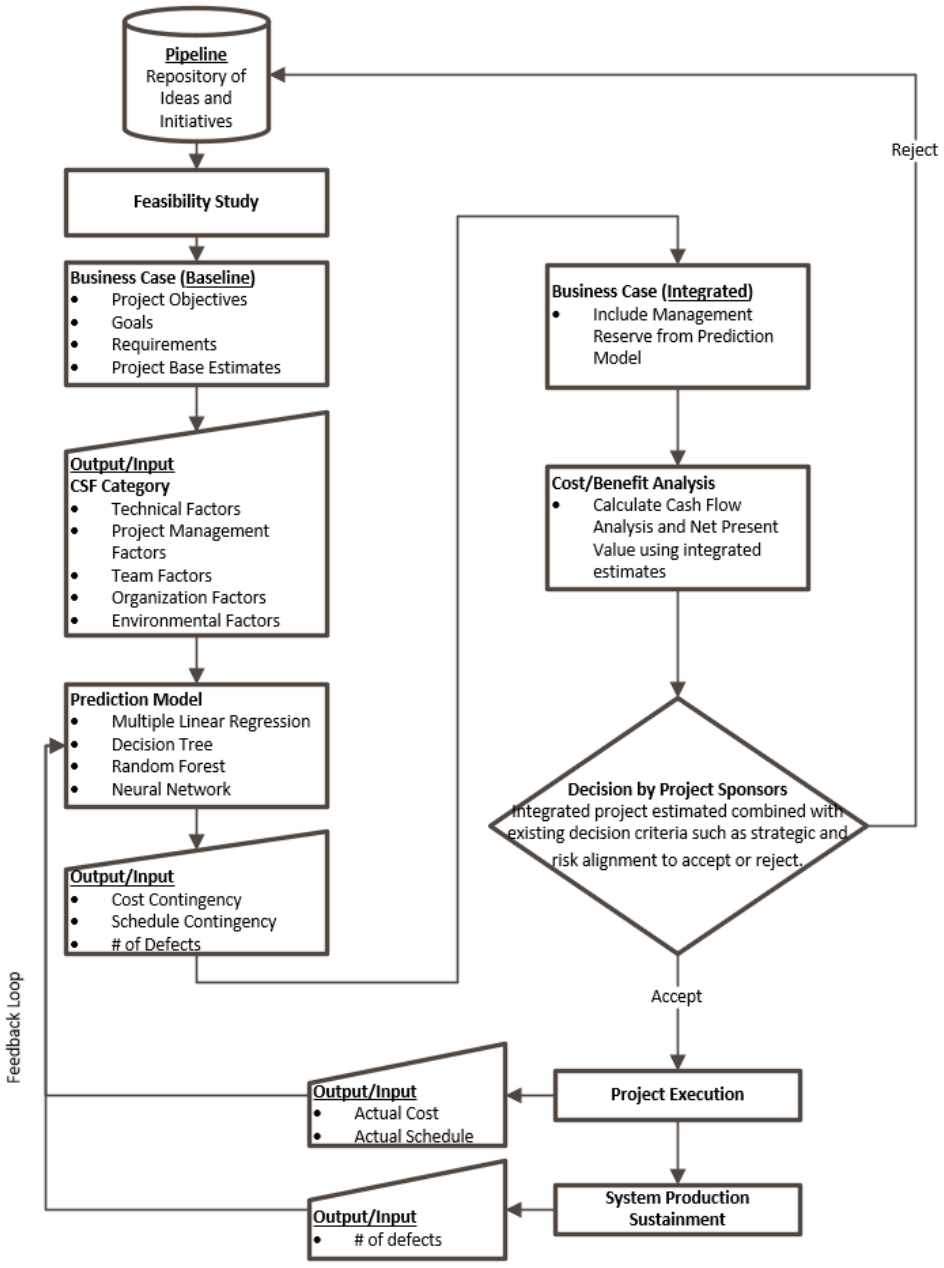

Figure 3-1. End-to-end Business Planning Process

The data source for this praxis was obtained from a Canadian large-size client organization that implements software projects in partnerships with external vendors in the Energy Sector. As part of the agreement to use the data for this research, the organization name, program names, resource names and any references to the company have been masked. The research questions outlined in this praxis guided the data selection and usage in this research.

RQ1: What features are effective predictors of multi-dimensional project outcomes from a client perspective?

This question was analyzed using the selected data set of CSFs and multi-dimensional software project performance outputs. Related literature was explored to identify the CSFs to be used in this praxis that contribute to the multi-dimensional software project success.

RQ2: Which proposed prediction model is the most effective in estimating multi-dimensional project outcomes based on prior project performance data?

This question was addressed by applying four machine learning methods using the CSFs and the multidimensional outputs from the selected data set.

The selected data set retrieved from the organizational repositories includes business case documents, project closure documents, and system defect records. The timeframe of the various software projects included in the data set ranges from 2016 to 2020. The data set contained a total of 208 software projects and 12,500 system defect records. Business case documents, project closure documents, and system defect records contain the identified CSFs and the multi-dimensional outputs to be used to train and test the supervised machine learning models. Business cases must be approved by an

executive sponsor and are stored in the project management centralized repository in the OpenText system. Business cases include CSFs such as project base cost, project base schedule, project methodology, and project team structure. The comprehensive list of CSFs is described in **Section 3.2.1**. Project closure documents are stored in a centralized repository on SharePoint and summarize the details of the project implementation including the actual project costs and duration. System defect records track all system defects in the organization's centralized ticketing system. Data retrieved from these documents serve as the labeled input and output data for the four supervised machine learning algorithms in this praxis. **Figure 3-2** is output from Python code using the Google Co-laboratory development environment and displays the first five entries of the variables in the data set.

	IntegrationPoints	Run	Grow	Transform	ProjectBaseCost	ProjectBaseSchedule	ProjectMethodology	ProjectTeamCapability
0	12	1	0	0	5614000	18	-1	25
1	2	0	0	1	836000	10	-1	6
2	2	0	1	0	283000	12	-1	5
3	1	0	1	0	1500000	15	-1	5
4	7	1	0	0	3843000	24	-1	7

TrainingEducation	TopManagement	Vendors	ExternalConstraints	Cost	Schedule	SystemDefects
1	3	3	-1	8462000	30	45
1	4	2	-1	892000	12	6
1	5	2	-1	620000	24	10
1	4	2	-1	2437000	19	12
1	2	2	-1	4309000	30	35

Figure 3-2. Sample Data Set

3.2.2 Critical Success Factors Selection

In this praxis, 10 CSFs were selected and divided into 5 factor categories based on the findings from literature review presented in **Section 2.2**. The factor categories and the CSFs were also selected based on their applicability to the Energy sector and software projects implemented in partnerships with external vendors. The factor category, the

name of the CSF, the variable type, and the definitions are detailed in **Table 3-1**. CSFs were clearly and formally defined to ensure data collection was consistently applied for all software projects with minimal bias.

Table 3-1. Critical Success Factor Variables

Factor Category	CSF	Factor Type	Definition
Technical	Integration of the System	Continuous	The number of integration points between software application systems.
	Technical Model	Categorical	Technical Model is a nominal categorical variable with values Run, Grow, or Transform based on the Gartner's Run-Grow-Transform model (Adnams, et al., 2018). Run refers to software project implementations that are required to operate and maintain the current environment; Grow refers to software project implementations to enhance and expand the current environment and footprint; Transform refers to software project implementations to innovate and drive new opportunities and changes in the organization.
Project Management	Project Base Cost	Continuous	This is the project cost estimate in dollars based on activity-based and work packaged estimates. This variable is formally documented in the Baseline Business Case.
	Project Base Schedule	Continuous	This is the project schedule estimate in months based on activity-based and work packaged estimates. This variable is formally documented in the Baseline Business Case.
	Project Methodology	Categorical	The methodology refers to the Project Management methodology. It is a categorical variable with values Heavyweight/Traditional Plan-Based or Lightweight/Agile.
Team	Project Team Capability	Continuous	Team Capability is represented by the project team size.
	Training and Education	Categorical	Training and Education refers to whether training is a pre-requisite for the project launch.
Organization	Top Management Support	Integer	This is an integer variable represented by the number of executive sponsors on the project.

Environmental	Vendor Partnership	Integer	This is an integer variable represented by the number of external vendors that have been contracted by the project team to work on the project.
	External Constraints	Categorical	External Constraints refers to whether external constraint exists such as regulatory compliance in the Energy Sector, safety compliance, accessibility compliance, etc.

3.2.3 Multi-Dimensional Outputs Selection

The goal of multi-dimensional learning is to predict multiple outputs simultaneously with a set of input features or CSFs. The multi-dimensional dependent variables in this praxis are schedule contingency, cost contingency, and the number of defects. **Table 3-2** defines the three output variables.

Table 3-2. Performance Output Definitions

Output Name	Definition
Cost Contingency	Cost contingency is of type continuous and it is the numeric difference in dollars between the baseline project budget before project starts, and the actual cost when project completes.
Schedule Contingency	Schedule contingency is of type continuous and it is the numeric difference in months between the baseline project duration before project starts, and the actual duration when project completes.
Number of System Defects	Number of system defects is the total number of defects tracked in the Remedy System in the first 3 months after the project is launched into a production environment.

3.3 Preprocessing of Data sets

Data preprocessing is a mandatory stage of any machine learning method and it impacts the accuracy of predictions (Huang et al., 2015). Careful attention must be paid to the data that is being used to design the machine learning algorithms. The first step of data preprocessing in this praxis is to screen all projects and remove non-qualifying entries. Projects must meet a set of defined criteria to be included in the data set for the

machine learning model application. Projects that do not meet all criteria will be excluded. Inclusion criteria are as follows:

1. Project must have a Business Case document approved by the project sponsor and stored in the company's OpenText System.
2. Project base cost must be $100,000 or above.
3. Project must be formally closed and a project closure document exists in the company's SharePoint system.
4. Project must be implemented in a production system and currently in service. System defects must be tracked in the company's Remedy system.
5. Project must have data for all 10 CSFs and all 3 project performance outputs.

Using the criteria above, 102 software projects met all criteria and were included in this praxis.

Encoding is another required preprocessing step when working with categorical variables in machine learning algorithms. While numerical data includes features composed of numbers that can be discrete or continuous, categorical variables contain label values that are limited to a fixed set. Categorical variables can be ordinal which is comprised of label values with a ranked ordering or they can be nominal where the label values have no relationship. In this praxis, binary encoding is used for categorical variables that have 2 values. One-hot encoding is a popular and effective method, and it is applied to the nominal categorical values in this praxis. This approach assigns a new dichotomous dummy variable with a Boolean value of ones or zeros for each unique nominal categorical label (Raschka & Mirjalili, 2019). Love and Edwards (2004), Jafarzadeh et al. (2014), and Forcada et al. (2017) have all employed the one-hot

encoding method for the categorical variables in their research. **Table 3-3** details the encoding values for the categorical CSFs in this praxis.

Table 3-3. CSF Encoding Values

Categorical CSF Variable	Encoding Method Used	Encoding Values
Technical Model	Technical model is a nominal categorical variable with values Run, Grow or Transform.	Three dichotomous dummy variables are created: Run: 0 or 1 Grow: 0 or 1 Transform: 0 or 1
Project Methodology	The methodology has 2 values: Heavyweight/Traditional Plan-Based or Lightweight/Agile. Binary Encoding is used.	0: Heavyweight/Traditional Plan-Based 1: Lightweight/Agile
Training and Education	Training and Education has two values representing training is required or training is not required. Binary encoding is used.	0: Training not required 1: Training required
External Constraints	External Constraints uses binary encoding to express whether external constraints exist or not.	0: No external constraints 1: External Constraints exist

3.4 Exploratory Data Analysis Methods

Exploratory Data Analysis (EDA) is a recommended approach that should be performed before the training of a machine learning model (Raschka & Mirjalili, 2019). In this praxis, descriptive statistics and graphical EDA are performed as initial investigations on the data to discover patterns and anomalies. EDA helps to maximize insights into a data set and minimize potential errors that may occur later in the process (Raschka & Mirjalili, 2019). Descriptive statistics using the *Panda* library (McKinney et al., 2010) and the *dataframe* class (McKinney et al., 2010) in the Python programming language summarize the central tendency, dispersion and shape of a data set's

distribution including the count, mean, standard deviation, percentiles, minimum, and maximum. Scatter plot matrices and correlation matrix heatmaps are effective EDA methods to obtain a high-level understanding of the relationship among the variables (Kuhn & Johnson, 2018). A scatter plot matrix helps visualize pair-wise correlations between different variables in a data set, and a correlation matrix heatmap is used to investigate the dependence between variables. The *seaborn* library (Waskom, 2021) and the *pairplot* class (Waskom, 2021) in Python programming language were used to create the scatter plots. The Pearson correlation coefficient was calculated measuring the degree of linear relationship between variables while the correlation matrix was created using the *heatmap* class in *seaborn* library (Waskom, 2021). If correlations between two variables are high, one of the variables will be eliminated. According to Sabilla et al. (2019), relationship strength is considered very strong if a correlation coefficient is between 0.70 and 0.89, and relationship strength is near perfect if a coefficient is ≥ 0.90. The threshold used to consider as a high correlation was set at 0.7 for this praxis.

3.5 Feature Importance and Ranking Method

The first research question was analyzed in this praxis using the Shapley Additive Explanations (SHAP). As discussed in **Section 2.4**, SHAP is a mathematically well-grounded measure for feature importance (Roth, 1988; Bowen & Ungar, 2020). SHAP values calculate the importance of a feature by estimating the performance of the prediction model with and without the feature (Lundberg & Lee, 2017). Features are then ordered and ranked based on the calculated SHAP values. The main advantage of SHAP values is its interpretability, especially in complex machine learning models.

The *shap* library (Lundberg & Lee, 2017) in the Python programming language has predefined classes to calculate the SHAP values for different machine learning models, and predefined feature importance plots for easy interpretation. Three classes in the *shap* library (Lundberg & Lee, 2017) are used in this praxis. The *LinearExplainer* (Lundberg & Lee, 2017) is used for the Multiple Linear Regression model which computes the SHAP values for a linear model. *TreeExplainer* (Lundberg et al., 2020) is used for the Decision Tree and Random Forest models which is a predefined method to estimate SHAP values. *KernelExplainer* (Lundberg & Lee, 2017) is used for the Neural Network model which is a generic class to explain the output of any prediction model. Two plots are created for each model to display the distribution of the impacts each feature has on the model outputs. The first plot is a SHAP value summary plot that shows the positive and negative relationship for the feature importance with the output variables. The second plot is a SHAP value summary bar plot that produces bars with the average absolute SHAP value for each feature and the features are ranked from the highest value to the lowest. **Figure 3-3** is an example of the SHAP value plots.

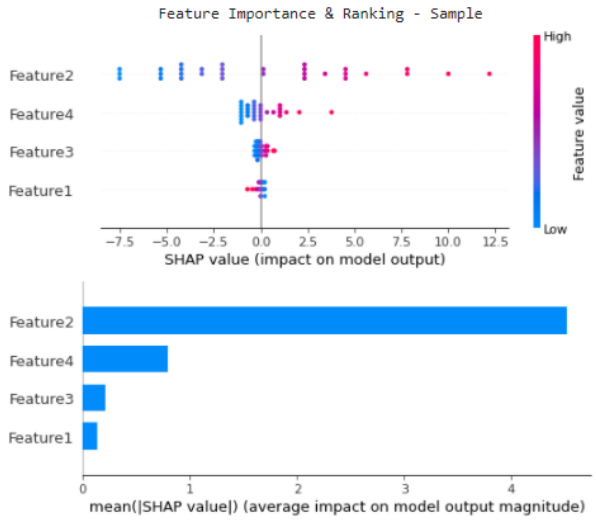

Figure 3-3. SHAP Value Plots Sample

3.6 Applied Machine Learning Methods

Four machine learning methods are applied in this praxis: Multiple Linear Regression, Decision Tree, Random Forest, and Neural Network. These machine learning algorithms aim to empower sound decision-making about software projects from a multi-dimensional perspective. The machine learning algorithms are programmed in Python version 3.7.12 code using the Google Co-laboratory development environment. Each machine learning method has 10 CSFs as inputs and 3 output variables as described in **Section 3.2**. This Multi-Dimensional Prediction Model (MDPM) is implemented in a chained 3-step process. The first step is denoted as the First Step Trained Model (FSTM) which takes the 10 CSFs as input representing the Independent Feature Set (IFS) #1. The second step is denoted as the Second Step Trained Model (SSTM) which takes the predicted value from FSTM in addition to the IFS #1 as input representing the IFS #2. The third and final step is denoted as the Third Step Trained Model (TSTM) which takes the predicted value from the SSTM in addition to the IFS #2 as input representing the IFS #3. **Figure 3-4** is the overall chained MDPM of all inputs, outputs, and the 3-step process. The overall MDPM output is composed of outputs from FSTM, SSTM, and TSTM.

In this praxis, the data set is randomly split into training and testing sets. The split is set at 70% for training and 30% for testing. The 70% and 30% split consists of output variables and input features at the same time, keeping correspondence between all output variables and its features. The training set is used to build and tune the model while the test set is used to create an unbiased evaluation of prediction performance. The same distribution of training and testing sets is utilized for all four methods consistently so the

performance evaluation of the prediction for the test set can be compared among the four machine learning methods.

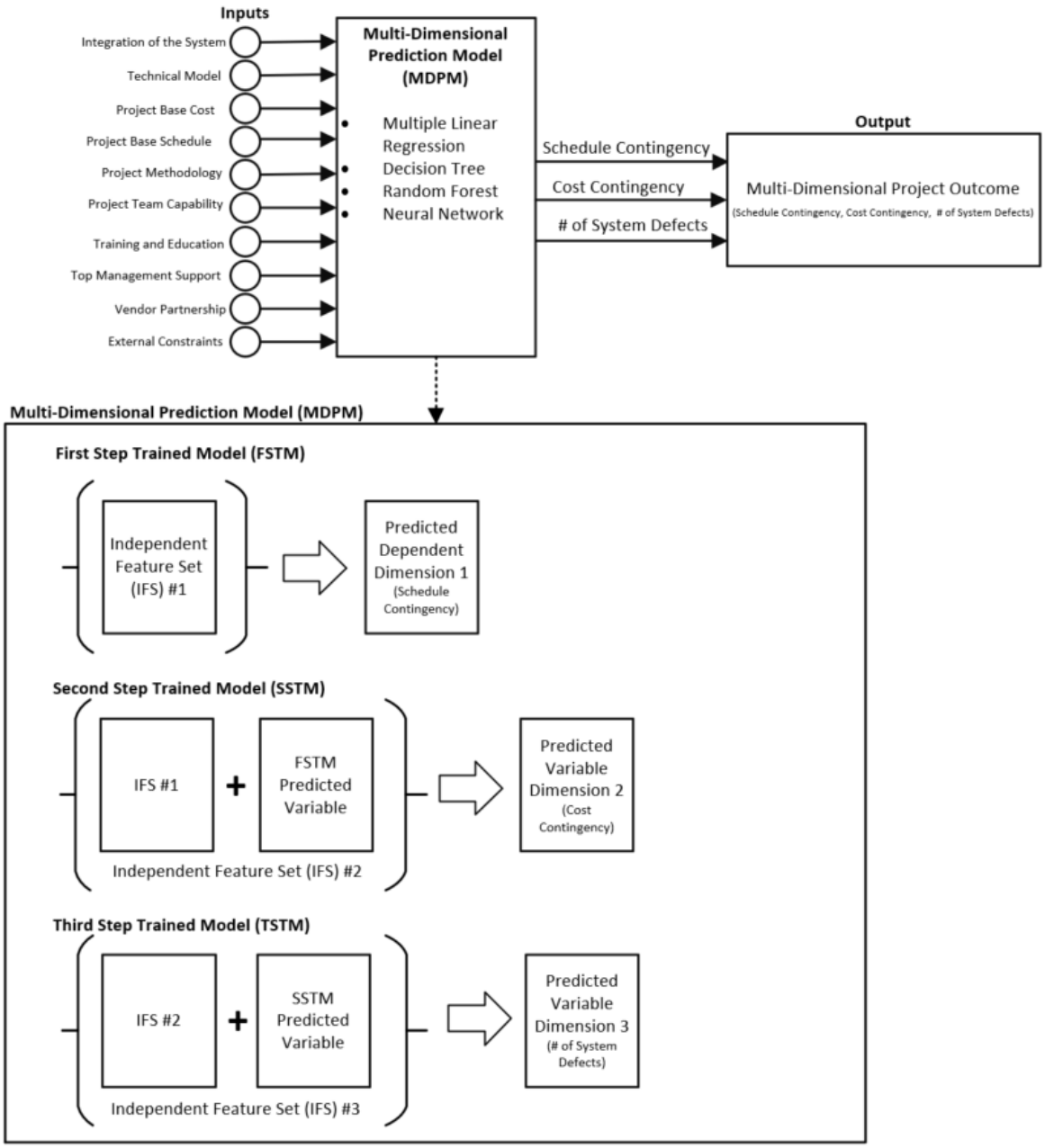

Figure 3-4. Chained Multi-Dimensional Prediction Model

3.6.1 Multiple Linear Regression

As discussed in **Section 2.5.2**, Linear Regression is the simplest machine learning method but it is very effective. For Linear Regression, the multi-dimensional outputs are expected to be a linear combination of the input features. Machine learning methods put emphasis on accurate predictions. For this machine learning method, in addition to calculating the model prediction accuracy, three main assumptions associated with the statistical Linear Regression must also be verified (Jafarzadeh et al., 2014). The first assumption of normality of the residual distribution is verified by performing the Anderson-Darling Normality Test. The Anderson-Darling Test is designed to detect if a distribution has any departure from the normal distribution (Nelson, 1998). The second assumption of homoscedasticity of residuals is inspected via scatter plots of the residuals to ensure residuals have constant variance for all output variables. The third assumption of lack of multicollinearity is validated by inspecting that no systematic pattern is detected in the residual plot. Jafarzadeh et al. (2014) stated that multicollinearity can also be validated by looking at the correlation matrix among all input variables. Correlation coefficients are calculated and the correlation matrix heatmap is generated as part of the exploratory data analysis in this praxis. **Figure 3-5** is a sample of a correlation matrix heatmap generated in Python programming language.

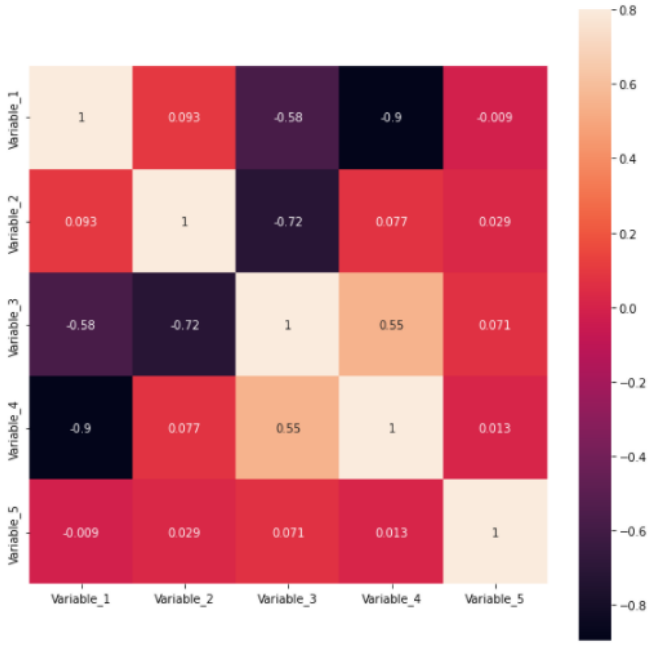

Figure 3-5. Correlation Matrix Heatmap Sample

Using the *sklearn.linear_model* library (Pedregosa et al., 2011) in the Python programming language, the *LinearRegressor* class (Pedregosa et al., 2011) is used to build the Linear Regression model with the Ordinary Least Squares (OLS) method. *LinearRegression* is a linear approximation method and generates coefficients to minimize the residual sum of squares between the predicted outputs and the observed output variables in the data set (Pedregosa et al., 2011). The coefficient estimates for OLS are based on the independence of the features which is verified by observing the correlation matrix. The default hyperparameters for the *LinearRegressor* class are used for this praxis. 10-fold cross-validation is also used for validating the prediction performance of the Linear Regression model to avoid overfitting.

In this praxis, the Least Absolute Shrinkage and Selection Operator (LASSO) Linear Regression was also implemented to see if the performance of the model improved. As described in **Section 2.5.2**, LASSO estimates sparse coefficients with L1

regularization where the regression coefficients of the less observed variable shrink to 0 by the penalty function (Wang et al., 2021). Using the same *sklearn.linear_model* library (Pedregosa et al., 2011) in the Python programming language, the *LassoCV* class (Pedregosa et al., 2011) is used to build the LASSO regression model.

3.6.2 Decision Tree

The Decision Tree method is an effective supervised learning algorithm for regression problems as discussed in **Section 2.5.3** in the literature review. It is a powerful machine learning model to achieve high accuracy while being highly interpretable (Raschka & Mirjalili, 2019). The Decision Tree has a tree-like structure with root and leaf nodes, and is built using recursive partitioning where data repeatedly is split based on the selected criteria with the objective to minimize prediction error (Tishya et al., 2019). The goal is to create a tree model that is a piecewise constant approximation and can learn decision rules inferred from the input features (Pedregosa et al., 2011).

Using the *sklearn.tree* library (Pedregosa et al., 2011) in the Python programming language, the *DecisionTreeRegressor* class (Pedregosa et al., 2011) is used to build the Decision Tree model. There are different hyperparameters for the *DecisionTreeRegression* in Python including the *criterion* which is the splitting criterion used to measure the quality of the split, *max_depth* which is the maximum depth of the Decision Tree, and *min_samples_split* which defines the minimum number of samples required to split an internal node. The splitting criteria selected for this model is the Mean Squared Error (MSE) calculation. The *max_depth* and *min_samples_split* hyperparameters are tuned during the training phase to optimize the prediction

performance without overfitting. In addition, 10-fold cross-validation is used for building and validating the prediction performance of the Decision Tree model to avoid overfitting.

3.6.3 Random Forest

As discussed in **Section 2.5.4**, a Random Forest is a meta estimator that combines multiple Decision Trees with the aim to increase the predictive accuracy and minimize over-fitting (Pedregosa et al., 2011). Each tree in the Random Forest ensemble is built from a bootstrap sample from the input data set. The bootstrap technique is a statistical technique for estimating quantities about a population which uses random sampling with replacement (Mittas & Angelis, 2013). Another source of randomness in a Random Forest algorithm is the splitting of each node during the construction of a tree. The split is found from a random subset of a size that is set by the maximum number of features as a hyperparameter for this learning method (Pedregosa et al., 2011). The bootstrap and splitting techniques help to decrease the variance and the Random Forest can be a better prediction model than an individual Decision Tree method.

Using the *sklearn.ensemble* library (Pedregosa et al., 2011) in the Python programming language, the *RandromForestRegressor* class (Pedregosa et al., 2011) is an averaging algorithm used to build the Random Forest model. There are different hyperparameters for the *RandromForestRegressor* in Python including *criterion* which is the function used to measure the quality of the split, *max_depth* which is the maximum depth of the trees, *max_features* which is the number of features to consider when looking for the best split, and *bootstrap* which defines whether bootstrap samples are to be used when building the trees. The splitting criteria selected for this model is the MSE

calculation, and the bootstrap is set to be true. The max_depth and *max_features* hyperparameters are experimented and tuned as part of the training phase to optimize the prediction performance. In addition, 10-fold cross-validation is used for building and validating the prediction performance of the Random Forest model.

3.6.4 Neural Network

As discussed in **Section 2.5.5**, Neural Network is a deep learning technique and has proven to be successful in different applications in various industries. Neural Network is comprised of an input layer, an output layer, and one or more hidden layers. *Keras* (Chollet, 2015) is an open-source library written in Python programming language for developing and evaluating deep learning models. The *sequential* model (Chollet, 2015) is a *Keras* model for Neural Network method and is used for this praxis. The model developed for this praxis has one input layer, 3 hidden layers and one output layer. A process of trial and error was performed to find the optimal number of layers for the model's connected network. These fully connected layers are defined using the *Dense* class (Chollet, 2015). Initializations define the way to set the initial random weights of Neural Network layers (Chollet, 2015). The normal distribution is selected as the kernel initializer for all layers of the Neural Network. The activation function in a layer defines the output of that node given a set of inputs, and this function is a critical part of the design of a Neural Network (Chollet, 2015). The choice of activation function in the input and hidden layers controls how well the network model learns from the training data set. The choice of activation function in the output layer defines the type of predictions the model can make (Chollet, 2015). The rectified linear activation

function (ReLU) is used for the input and hidden layers. The linear activation function is used for the output layer.

The Neural Network model is compiled using the Mean Absolute Error (MAE) as the error evaluation metric and the loss function. The Adam optimizer algorithm (Chollet, 2015) is an extension to stochastic gradient descent and is employed to set the network weights in the training data. Two hyperparameters required to compile the Neural Network model are epoch and batch size where the former is the number of times the algorithm must learn through the training data set while the latter is the number of training samples to work through for each epoch. These hyperparameters are tuned during the training phase to find the optimal number of batches and epochs for this praxis. In addition, this method often performs significantly better when the input variables are normalized to be between 0 and 1. Therefore, in this praxis, transformation is performed to normalize all input and output variables as a preprocessing step for this method. The *MinMaxScaler* class in the *sklearn.preprocessing* (Pedregosa et al., 2011) library in Python programming language is used.

3.7 Data Validation

According to Myrtveit et al. (2005), the methodology by which the data sets are collected and measured are crucial to empirical software engineering while data validation is an imperative step in machine learning. Broniatowski & Tucker (2017) also emphasized that the goal of assessing the accuracy of a model's causal relationship is to compare the model's predictions against observations, and reliable and consistent observations are a necessary condition. In this praxis, construct validity, internal validity, and external validity were carefully examined and followed. From a construct validity

perspective, the measurement tools and the set of documents identified in this praxis were carefully studied to ensure they accurately represent the variables the model intended to measure. From an internal validity perspective, the CSFs and multi-dimensional outputs are clearly and formally defined to ensure the observed association between the independent variables (i.e. CSFs) and the dependent variables (i.e. multi-dimensional outputs) are attributed to a causal link between them. It is hard to eliminate all confounding variables and ensure projects collected for this praxis are in a pure controlled experiment, but variables are consistently applied for all projects to avoid misinterpretation. In addition, in order to minimize selection bias, histograms are generated in Python programming language to verify that the list of selected projects is representative of all types of software projects.

In terms of external validity, the praxis uses validation techniques to ensure the prediction model is not overfitting. The two validation techniques employed are 10-fold cross-validation and validation split set. Tuning the machine learning algorithms is an iterative process. Each machine learning model is trained and validated independently using one of the two techniques. For Multiple Linear Regression, Decision Tree, and Random Forest models, 10-fold cross-validation is used. The training data is split into 10 randomly selected subsets, and the predictor is trained on a set of subsamples and tested on a held out unseen subsample to estimate performance. Using the *sklearn.model_selection* library (Pedregosa et al., 2011) in the Python programming language, the *KFold* class (Pedregosa et al., 2011) is used to perform the 10-fold cross validation, and the *cross_val_score* helper function (Pedregosa et al., 2011) is used to compute the validation scores. For the Neural Network model, the *Keras* (Chollet, 2015)

library was employed which has a pre-defined *validation_split* argument to separate a subset of the training data into a validation data set. The performance evaluation of the model on that validation data set is represented by the validation loss function per epoch indicating how well the model fits new data.

Hyperparameters are tuned during the training phase using the training set. Validation techniques are employed to ensure the model is not overfitting. Once the training phase is complete and satisfactory results are achieved, the testing set, an unseen set of data, is used to test the model to predict the multi-dimensional outcomes. The evaluation will also validate the testing set's true values compared to the machine learning model's predicted values. This methodology aims to create a model that has high validity and minimal bias.

3.8 Model Performance Evaluation Metrics

Several evaluation criteria and metrics used by researchers in measuring prediction performance of machine learning models were discussed in **Section 2.6**. The choice of evaluating machine learning model varies for classification and regression, and varies from different applications and industries (Witten et al., 2016). For this research, two evaluation metrics, Normalized Mean Absolute Error (NMAE) and Normalized Root Mean Squared Error (NRMSE), are used and are presented in **Equations 3-2 and 3-4** respectively. **Equation 3-2** is a normalized metric based on the Mean Absolute Error (MAE) in **Equation 3-1**, and **Equation 3-4** is a normalized metric based on the Root Mean Squared Error (RMSE) in **Equation 3-3**. The NMAE and NRMSE will be calculated for each dimension (schedule contingency, cost contingency, system defects) for each machine learning model. The overall multi-dimensional evaluation metrics will

be the average of the three dimensions. The research objective of this praxis is to compare the four different models. Therefore, it is important to ensure these metrics are consistently applied and compared against the four machine learning models.

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |Actual_i - Predicted_i|$$

Equation 3-1. Mean Absolute Error (MAE)

where

MAE is the Mean Absolute Error,

N is the total number of data points,

$Actual_i$ is the true value, and

$Predicted_i$ is the predicted value from machine learning model.

$$NMAE = \frac{MAE}{y_{max} - y_{min}}$$

Equation 3-2. Normalized MAE (NMAE)

where

NMAE is the Normalized Mean Absolute Error,

MAE is the Mean Absolute Error,

y_{max} is the maximum value of all data points, and

y_{min} is the minimum value of all data points.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(Actual_i - Predicted_i)^2}{N}}$$

Equation 3-3. Root Mean Squared Error (RMSE)

where

RMSE is the Root Mean Squared Error,

N is the total number of data points,

$Actual_i$ is the true value, and

$Predicted_i$is the predicted value from machine learning model.

$$NRMSE = \frac{RMSE}{y_{max} - y_{min}}$$

Equation 3-4. Normalized RMSE (NRMSE)

where

NRMSE is the Normalized Root Mean Squared Error,

RMSE is the Root Mean Squared Error,

y_{max} is the maximum value of all data points, and

y_{min} is the minimum value of all data points.

Interpretation of the percentage error metrics helps to determine the accuracy of the forecast and to compare the different prediction models (Lewis, 1982). The normalized metrics are dimensionless and represented in percentages which makes it easy to interpret and compare to other models and data sets outside of this praxis. Therefore, normalized metrics are preferred and selected for this praxis. According to Lewis (1982), an error rate of less than 10% is considered a highly accurate forecast, 11% to 20% is considered a good forecast, 21% to 50% is considered a reasonable forecast, and 51% or more is considered an inaccurate forecast. These criteria and percentages will be used as the accuracy thresholds in this praxis.

Chapter 4—Results

4.1 Introduction

This chapter presents the results of data analysis using the Multi-Dimensional Prediction Model (MDPM). The data was obtained from a Canadian large-size client organization that implements software projects in the Energy Sector. As part of the agreement to use the data for this research, any references to the company information have been masked. After the preprocessing of the data set using pre-defined criteria as described in **Section 3.3**, 102 software projects were included in this analysis from 2016 to 2020. Four machine learning methods (Multiple Linear Regression, Decision Tree, Random Forest, and Neural Network) were developed and compared using 10 Critical Success Factors (CSFs) as inputs and the multi-dimensional output variables were schedule contingency, cost contingency, and number of system defects.

Section 4.2 reviews the findings of the Exploratory Data Analysis (EDA) as the first step prior to the training of the MDPM. **Section 4.3** presents the results of Hypothesis 1 and provides an overview of the outcomes of feature importance and ranking of the CSFs. **Sections 4.4 - 4.7** present the results from each machine learning method and for Hypotheses 2, 3, 4 and 5, respectively. Finally, **Section 4.8** presents the validation results of the four models.

4.2 Exploratory Data Analysis Results

Effective EDA methods help researchers to obtain a high-level understanding of the data set and the relationship among the variables (Kuhn & Johnson, 2018). Both descriptive statistics and graphical EDA were performed as initial investigations on the input data. **Figure 4-1** is an output from Python code using the Google Co-laboratory

development environment, and it summarizes the descriptive statistics of the variables in the data set including count, mean, standard deviation, percentiles, minimum, and maximum. A review of these descriptive statistics does not reveal any abnormalities.

	Schedule	Cost	SystemDefects
count	102.000000	1.020000e+02	102.000000
mean	11.539216	1.152985e+06	16.696078
std	6.900526	1.466033e+06	11.540535
min	3.000000	5.500000e+04	2.000000
25%	6.000000	3.000000e+05	9.000000
50%	10.000000	7.695000e+05	12.500000
75%	15.750000	1.231250e+06	20.750000
max	30.000000	8.462000e+06	50.000000

	IntegrationPoints	Run	Grow	Transform	ProjectBaseSchedule	ProjectTeamCapability
count	102.000000	102.000000	102.000000	102.000000	102.000000	102.000000
mean	2.862745	0.343137	0.343137	0.313725	8.882353	5.764706
std	2.971945	0.477101	0.477101	0.466298	5.358983	4.229029
min	0.000000	0.000000	0.000000	0.000000	1.000000	2.000000
25%	1.000000	0.000000	0.000000	0.000000	5.000000	3.000000
50%	2.000000	0.000000	0.000000	0.000000	7.500000	5.000000
75%	3.000000	1.000000	1.000000	1.000000	12.000000	7.000000
max	14.000000	1.000000	1.000000	1.000000	25.000000	25.000000

	TrainingEducation	TopManagement	Vendors	ExternalConstraints	ProjectMethodology	ProjectBaseCost
count	102.000000	102.000000	102.000000	102.000000	102.000000	1.020000e+02
mean	-0.058824	2.049020	1.294118	-0.705882	-0.568627	8.868206e+05
std	1.003198	1.137806	0.739122	0.711827	0.826657	1.247495e+06
min	-1.000000	1.000000	0.000000	-1.000000	-1.000000	5.750000e+04
25%	-1.000000	1.000000	1.000000	-1.000000	-1.000000	2.285250e+05
50%	-1.000000	2.000000	1.000000	-1.000000	-1.000000	5.600000e+05
75%	1.000000	3.000000	2.000000	-1.000000	-1.000000	8.870000e+05
max	1.000000	5.000000	4.000000	1.000000	1.000000	6.800000e+06

Figure 4-1. Descriptive Statistics

Figure 4-2 is a scatter plot matrix of the continuous variables in the data set using the Kernel Density Estimate (KDE) function in the Python programming language. A KDE plot helps to visualize the distribution of the pairwise correlations between the continuous input CSF variables and the multi-dimensional output variables in the data set. Analysis of the scatterplots in **Figure 4-2** suggests that relationships between the variables exist and that these relationships should be further explored using machine learning methods.

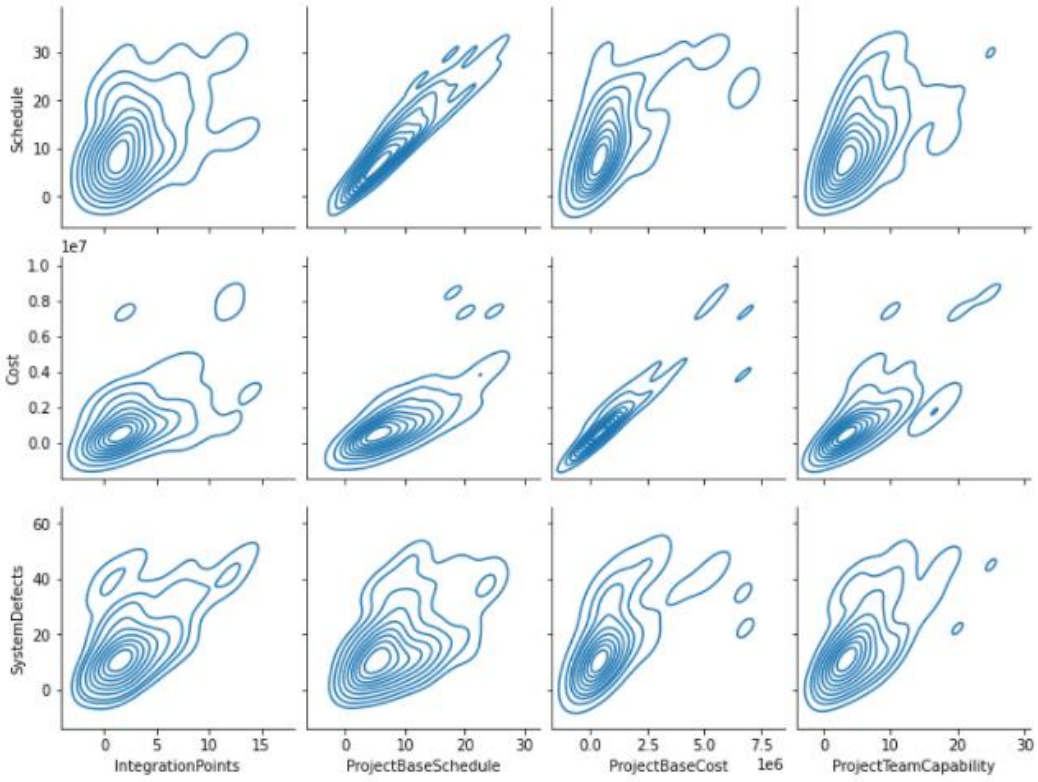

Figure 4-2. Scatter Plot

Figure 4-3 is the correlation matrix heatmap displaying the correlation coefficients between sets of variables. The correlation coefficient value helps to identify any dependence relationship between variables. As discussed in **Section 3.4**, a correlation coefficient of ≥ 0.70 signifies a high correlation between two variables. The results in **Figure 4-3** demonstrate absence of any highly correlated variables.

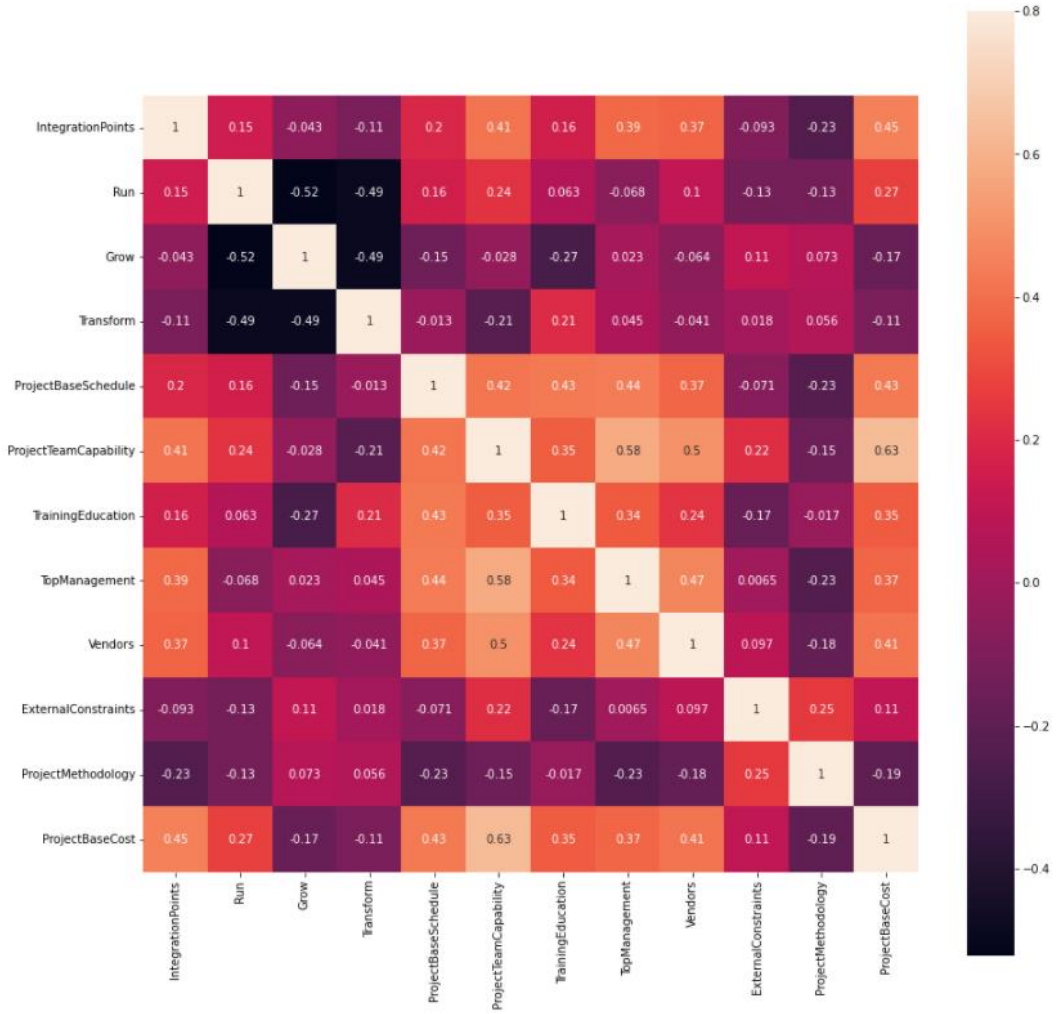

Figure 4-3. Correlation Matrix Heatmap

4.3 Hypothesis 1 and Results

H1: *The Critical Success Factors (Independent Variables) identified in this praxis have significant influence on the dependent variables.*

The hypothesis was tested and the results were analyzed by employing the Shapley Additive Explanations (SHAP) values for the four machine learning models. Using predefined libraries and classes in the Python programming language as discussed in **Section 3.5**, feature importance plots and feature ranking graphs were created. Two plots were generated for each model:

1. SHAP Value Summary Plot (SVSP): it displays the positive and negative relationship for the feature importance with the output variables. Each dot in the visualization represents one prediction. The color pink indicates a high feature value in the data set and color blue represents a low feature value.
2. SHAP Value Bar Plot (SVBP): it displays the average absolute SHAP value for each feature ranked from the highest value to the lowest.

4.3.1 Multiple Linear Regression Feature Results

Figures 4-4, 4-5 and 4-6 present the feature importance and feature ranking results in the schedule contingency dimension, cost contingency dimension, and the system defects dimension, respectively, for the Multiple Linear Regression model. The top plot is the SVSP and the bottom plot is the SVBP. Discussion of these results is presented in **Section 5.1.1**.

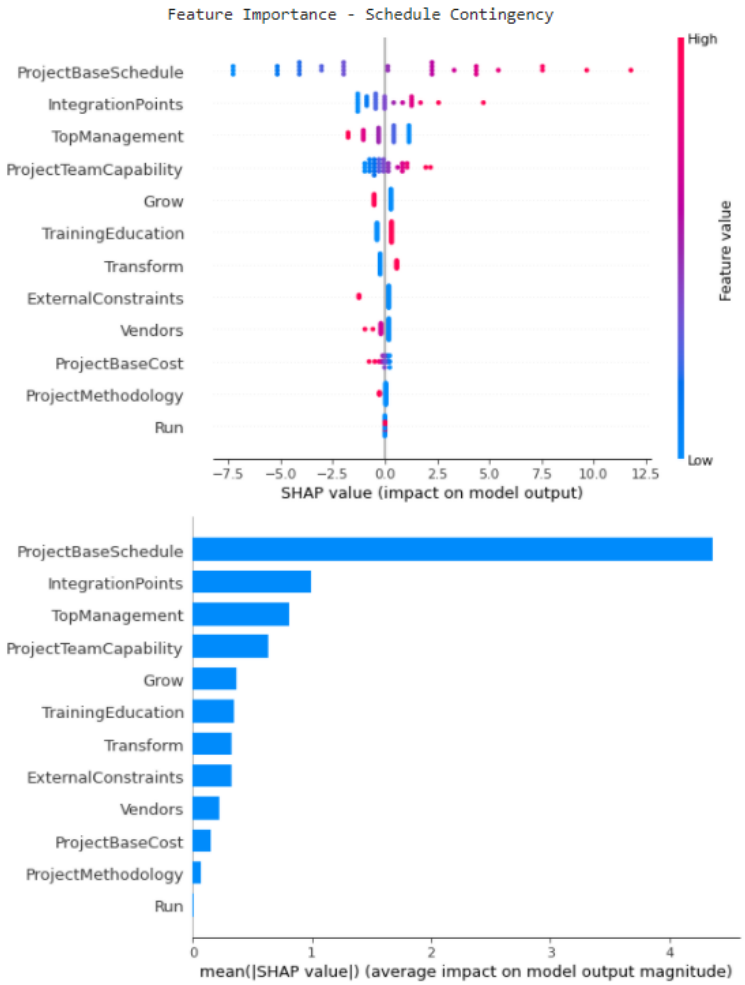

Figure 4-4. MLR Feature Importance and Ranking Results for the Schedule Contingency Dimension

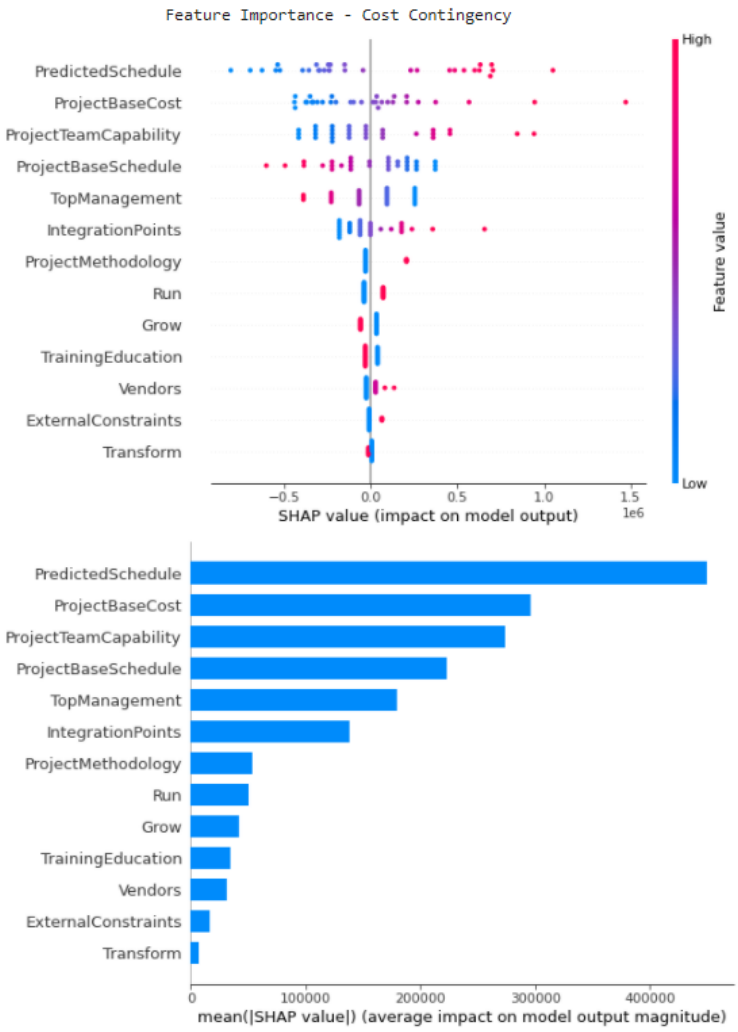

Figure 4-5. MLR Feature Importance and Ranking Results for the Cost Contingency Dimension

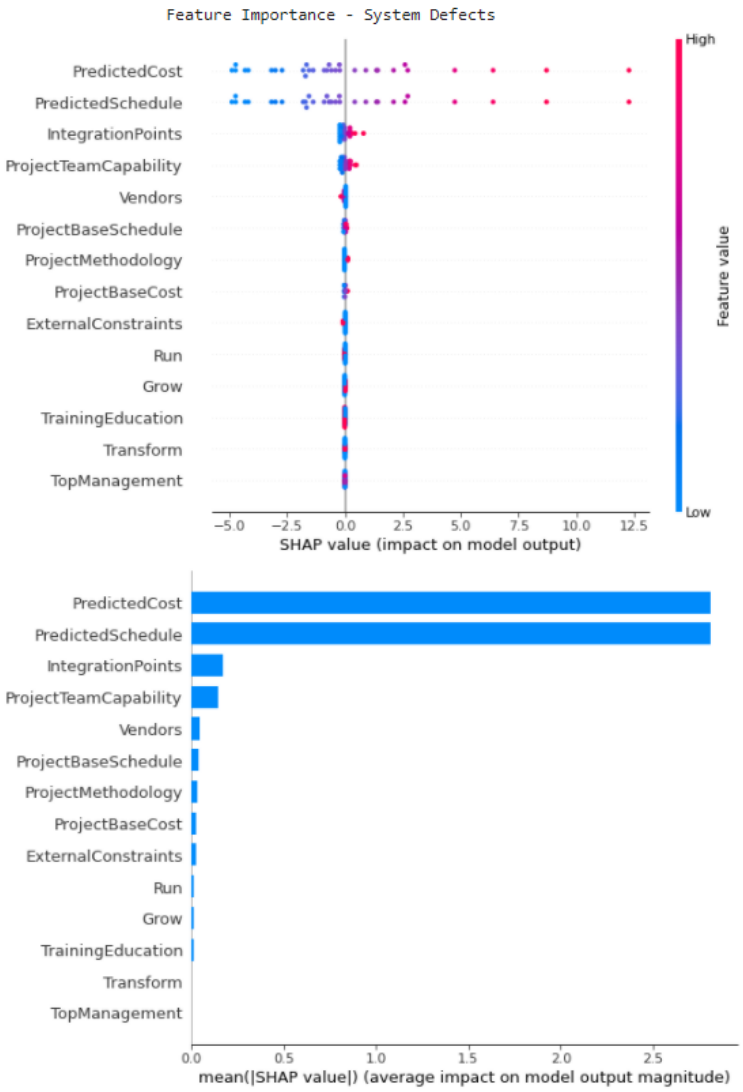

Figure 4-6. MLR Feature Importance and Ranking Results for the System Defect Dimension

4.3.2 Decision Tree Feature Results

This section covers feature importance and feature ranking results of the Decision Tree machine learning model. **Figures 4-7, 4-8, and 4-9** present results in the three dimensions of schedule contingency, cost contingency and system defects, respectively. The top plot is the SVSP and the bottom plot is the SVBP. Discussion of these results is presented in **Section 5.1.1**.

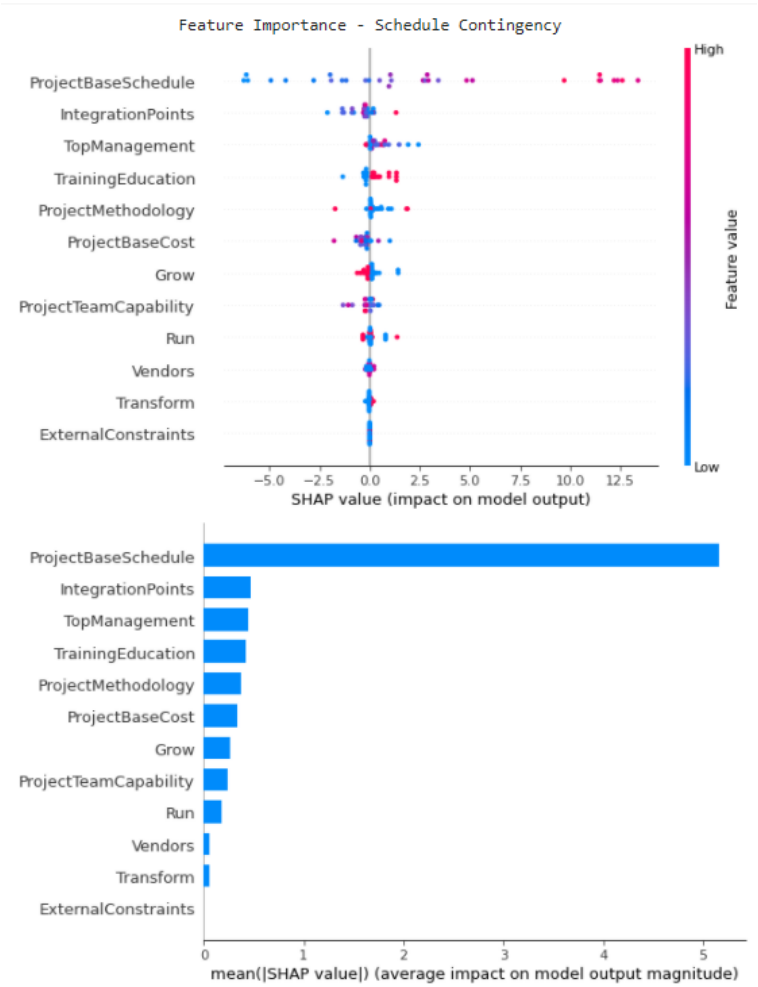

Figure 4-7. DT Feature Importance and Ranking Results for the Schedule Contingency Dimension

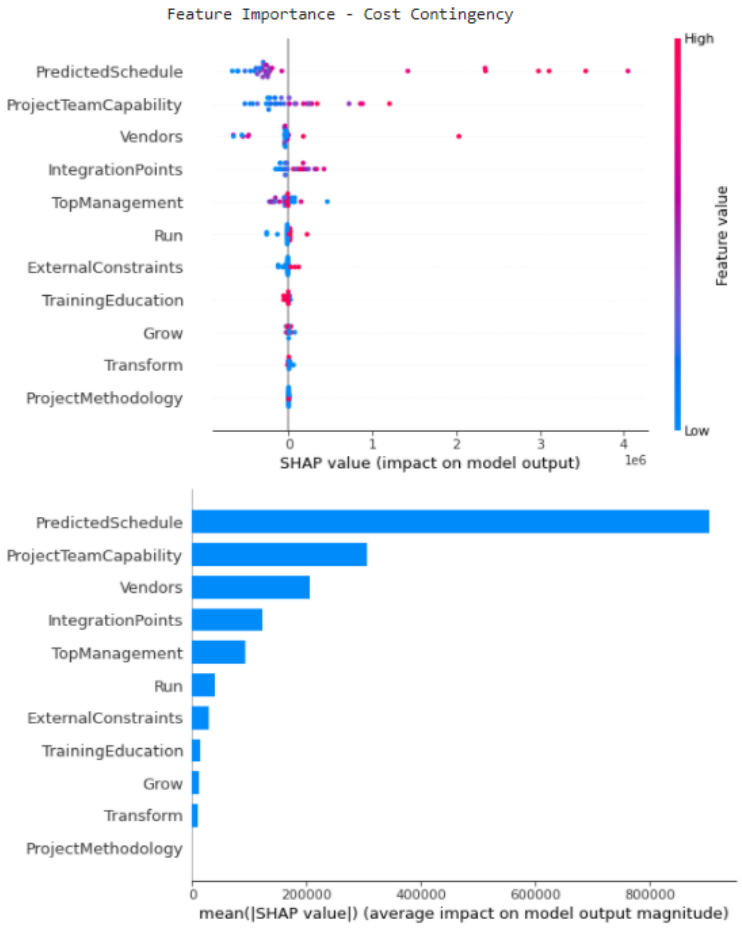

Figure 4-8. DT Feature Importance and Ranking Results for the Cost Contingency Dimension

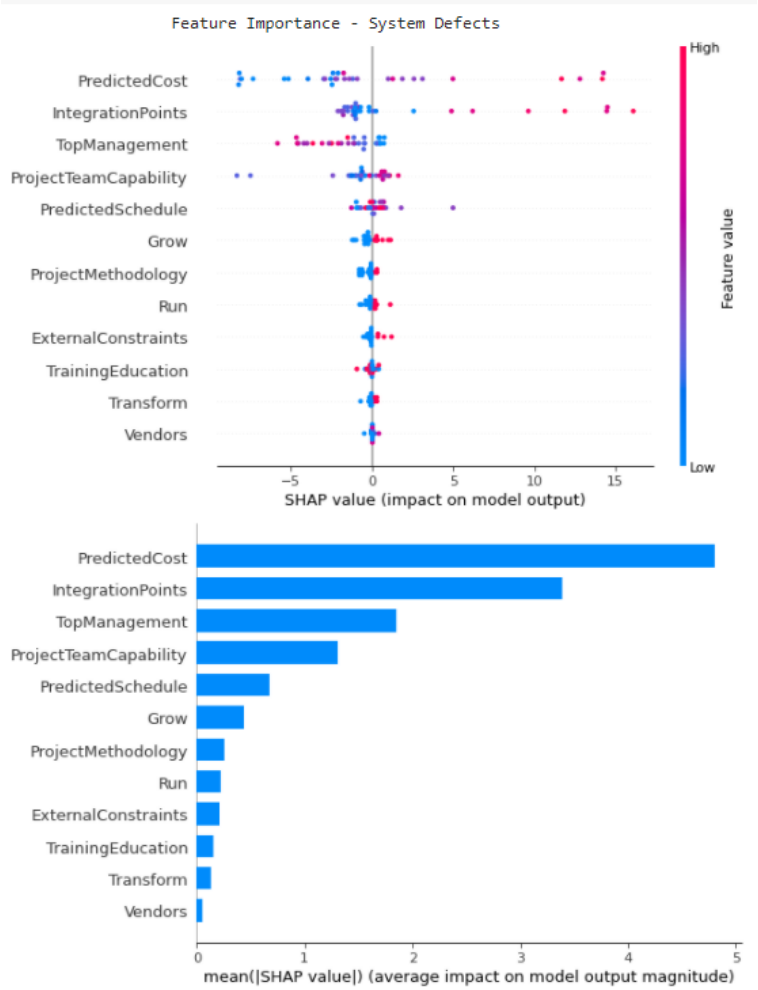

Figure 4-9. DT Feature Importance and Ranking Results for the System Defect Dimension

4.3.3 Random Forest Feature Results

This section covers the results of the Random Forest machine learning model. **Figures 4-10, 4-11, and 4-12** present the feature importance and feature ranking results in the schedule contingency dimension, cost contingency dimension, and system defects dimension, respectively, in both the SVSP and SVBP. Discussion of these plots is presented in **Section 5.1.1**.

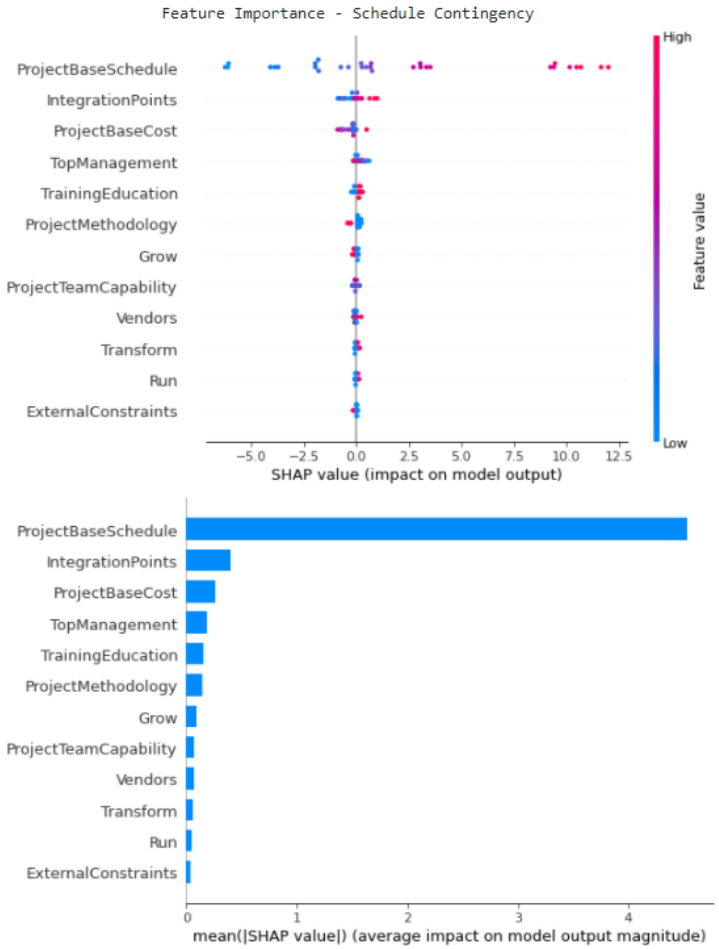

Figure 4-10. RF Feature Importance and Ranking Results for the Schedule Contingency Dimension

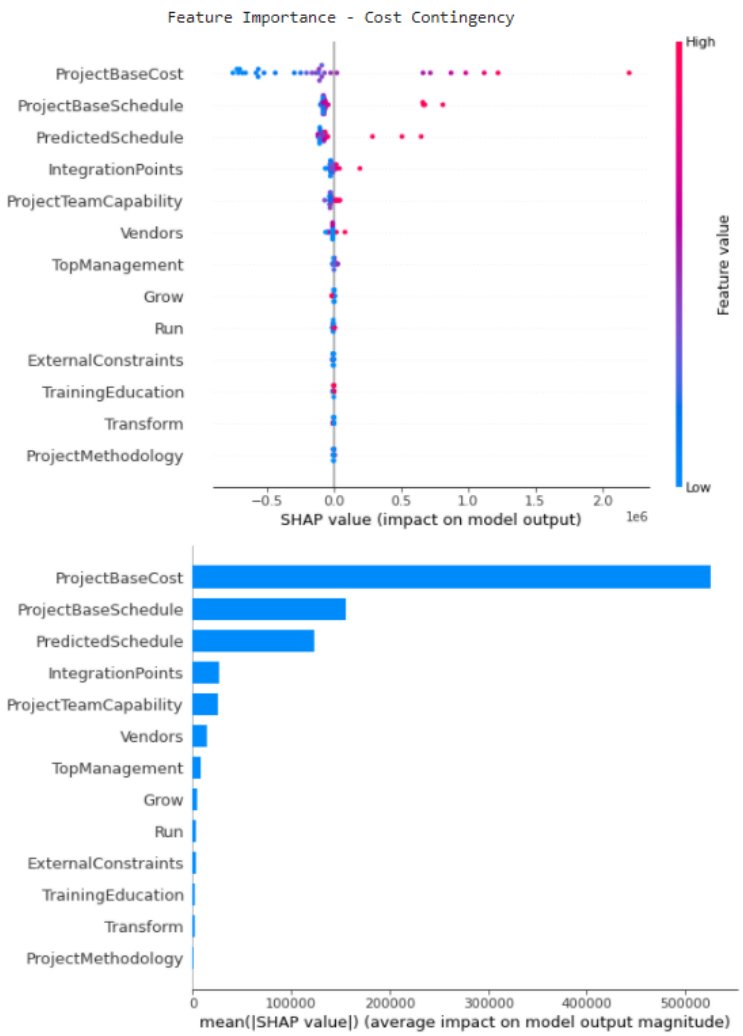

Figure 4-11. RF Feature Importance and Ranking Results for the Cost Contingency Dimension

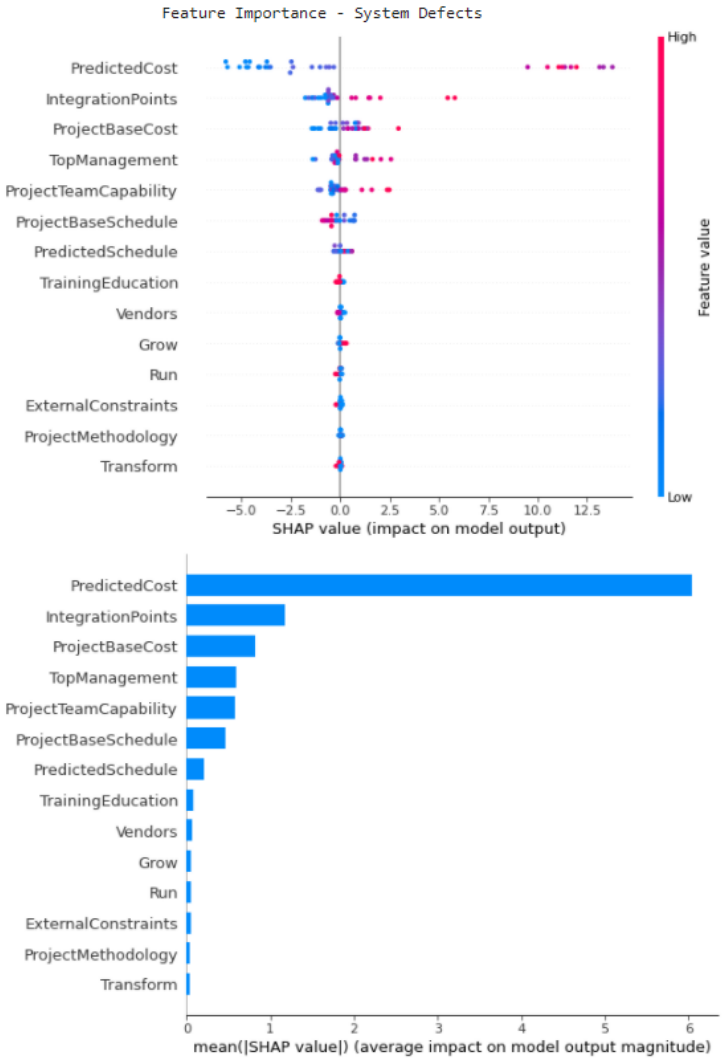

Figure 4-12. RF Feature Importance and Ranking Results for the System Defect Dimension

4.3.4 Neural Network Feature Results

This section covers the results of the Neural Network machine learning model. **Figures 4-13, 4-14, and 4-15** present the feature importance and feature ranking results in the three dimensions of schedule contingency, cost contingency and system defects, respectively. The top plot is the SVSP and the bottom plot is the SVBP. Discussion of these results is presented in **Section 5.1.1**.

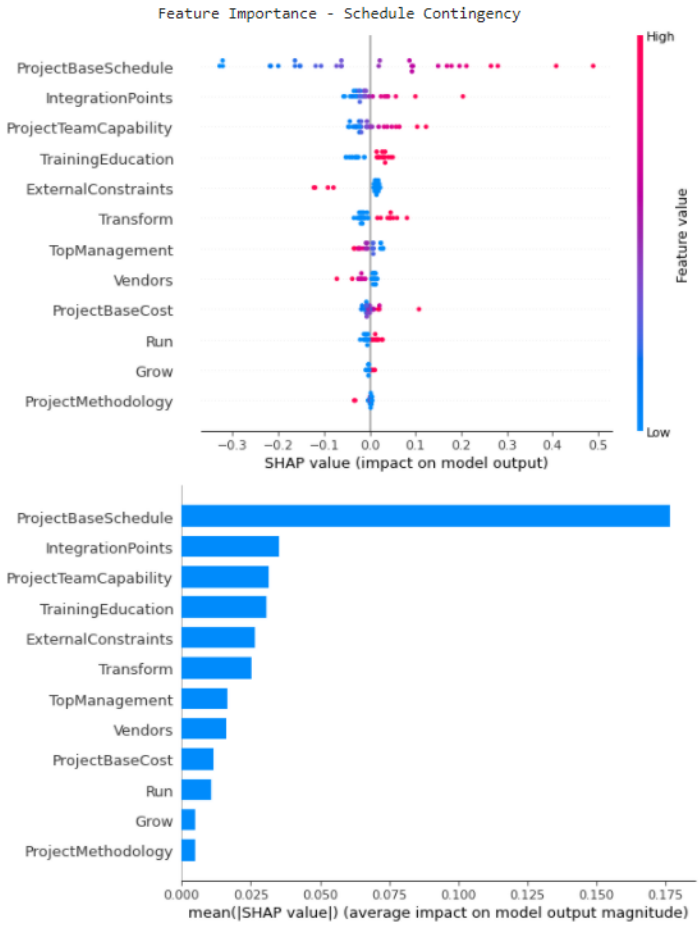

Figure 4-13. NN Feature Importance and Ranking Results for the Schedule Contingency Dimension

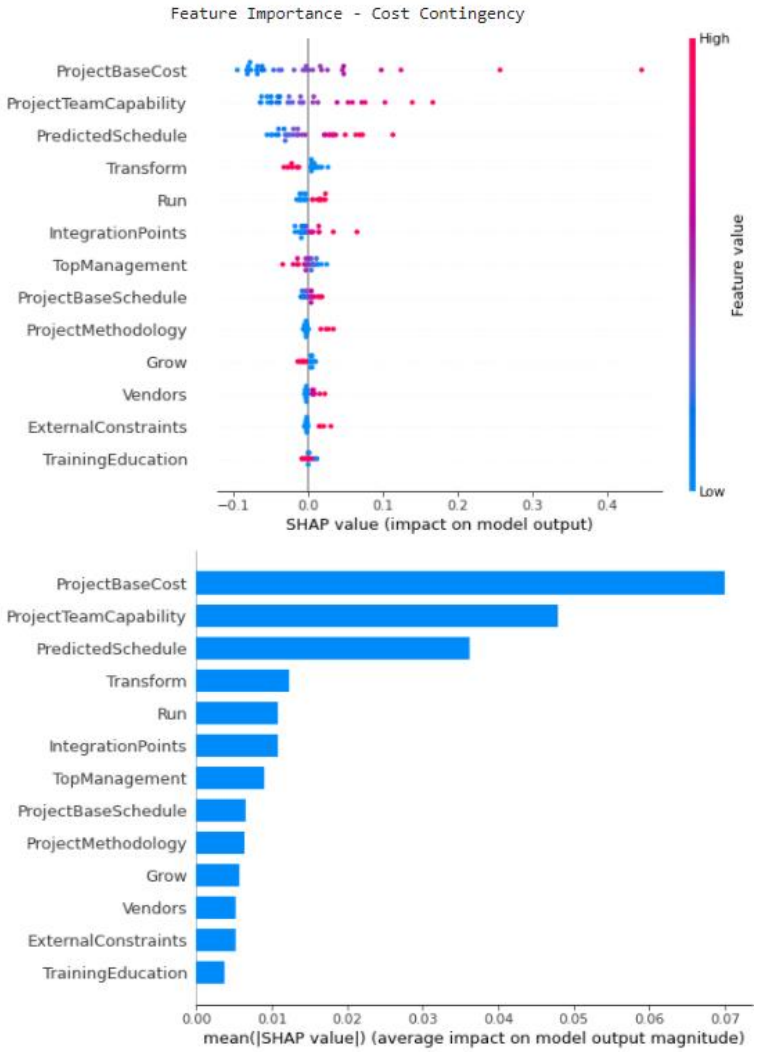

Figure 4-14. NN Feature Importance and Ranking Results for the Cost Contingency Dimension

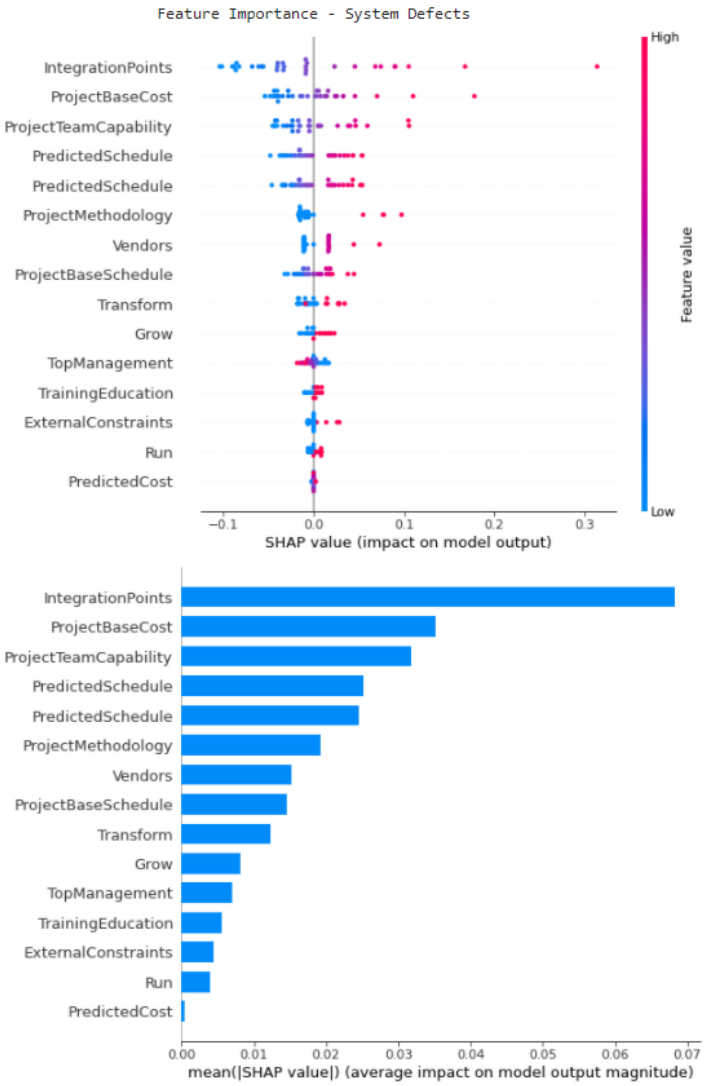

Figure 4-15. NN Feature Importance and Ranking Results for the System Defect Dimension

4.3.5 Summary of Feature Results

This section summarizes the results from the four machine learning models presented in **Sections 4.3.1 to 4.3.4**. **Table 4-1** presents the top five CSFs with the most significant influence on the output variables in the three dimensions of schedule contingency, cost contingency and system defects for each machine learning model. **Figure 4-16** is an aggregated count plot of the top 5 CSFs in each dimension. Discussion of these results is presented in **Section 5.1.1**.

Table 4-1. Top 5 CSFs for each Dimension and Model

	Machine Learning Model			
Dimension	**Multiple Linear Regression**	**Decision Tree**	**Random Forest**	**Neural Network**
Schedule Contingency Dimension	1. Project Base Schedule 2. Integration of the System 3. Top Management Support 4. Project Team Capability 5. Technical Model	1. Project Base Schedule 2. Integration of the System 3. Top Management Support 4. Training and Education 5. Project Methodology	1. Project Base Schedule 2. Integration of the System 3. Project Base Cost 4. Top Management Support 5. Training and Education	1. Project Base Schedule 2. Integration of the System 3. Project Team Capability 4. Training and Education 5. External Constraints
Cost Contingency Dimension	1. Project Base Cost 2. Project Team Capability 3. Project Base Schedule 4. Top Management Support 5. Integration of the System	1. Project Team Capability 2. Vendor Partnership 3. Integration of the System 4. Top Management Support 5. Technical Model	1. Project Base Cost 2. Project Base Schedule 3. Integration of the System 4. Project Team Capability 5. Vendor Partnership	1. Project Base Cost 2. Project Team Capability 3. Technical Model 4. Integration of the System 5. Top Management Support
System Defects Dimension	1. Integration of the System 2. Project Team Capability 3. Vendor Partnership 4. Project Base Schedule 5. Project Methodology	1. Integration of the System 2. Top Management Support 3. Project Team Capability 4. Technical Model 5. Project Methodology	1. Integration of the System 2. Project Base Cost 3. Top Management Support 4. Project Team Capability 5. Project Base Schedule	1. Integration of the System 2. Project Base Cost 3. Project Team Capability 4. Project Methodology 5. Vendor Partnership

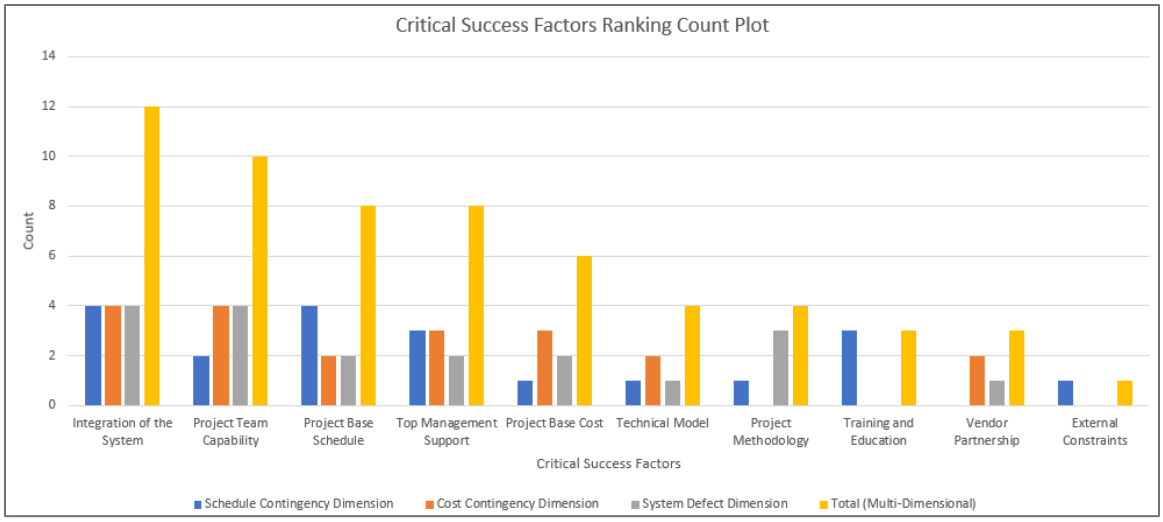

Figure 4-16. Summary Count Plot of Top 5 CSFs

4.4 Hypothesis 2 and Results

H2: *A Multiple Linear Regression model can be developed to predict multi-dimensional project outcomes with Normalized Mean Absolute Error (NMAE) and Normalized Root Mean Squared Error (NRMSE) to be less than or equal to 20%.*

Three main assumptions discussed in **Section 3.6.1** were validated as part of the analysis:

1. Normality of the residual distribution is verified by performing the Anderson-Darling Normality Test. The Anderson-Darling test rejects the hypothesis of normality when the p-value is ≤ 0.05 (Nelson, 1998). Failing the normality test signifies that the data does not fit the normal distribution with a 95% confidence level. P-value of ≥ 0.05 states that data does not have significant departure from normality. **Table 4-2** presents the results of Anderson-Darling Normality tests for the residual values from the training phase in each dimension confirming the data does not have significant departure from normality.

Table 4-2. MLR Normality Check P-Value

Dimension	P-Value
Schedule Contingency	0.496
Cost Contingency	0.229
System Defects	0.065

2. Homoscedasticity of residuals is inspected via scatter plots of the residuals to ensure residuals have constant variance for all output variables. Residual scatter plots from the training phase are presented in **Figures 4-17, 4-18**, and **4-19** for the three dimensions of schedule contingency, cost contingency and system defects, respectively, confirming residuals have constant variance.
3. Lack of multicollinearity is validated by inspecting that no systematic pattern is detected in the residual plot. Assumptions state that residuals must have a constant variance, normally distributed with a mean of zero, and independent of one another. The residual scatter plots and residual distribution plots from the training phase are presented in **Figures 4-16, 4-17, and 4-18** for the three dimensions, respectively, confirming lack of multicollinearity.

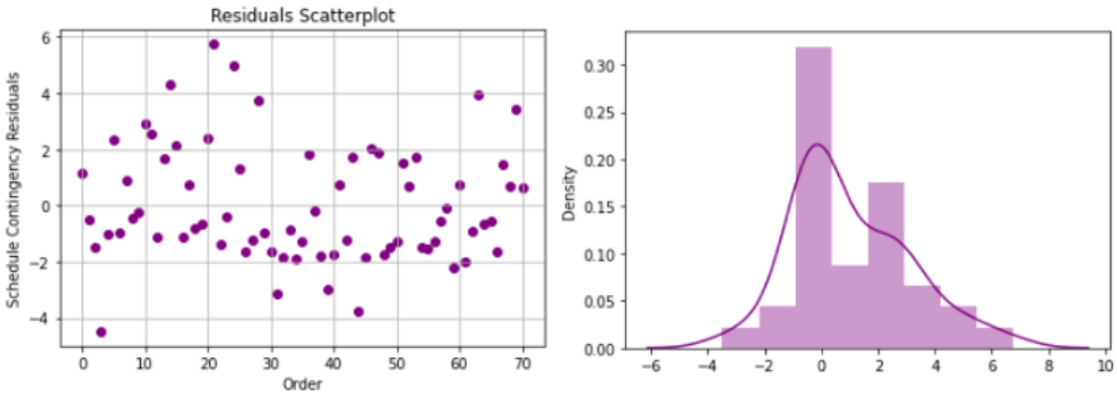

Figure 4-17. MLR Residual plot and histogram plot of residuals for the Schedule Contingency Dimension

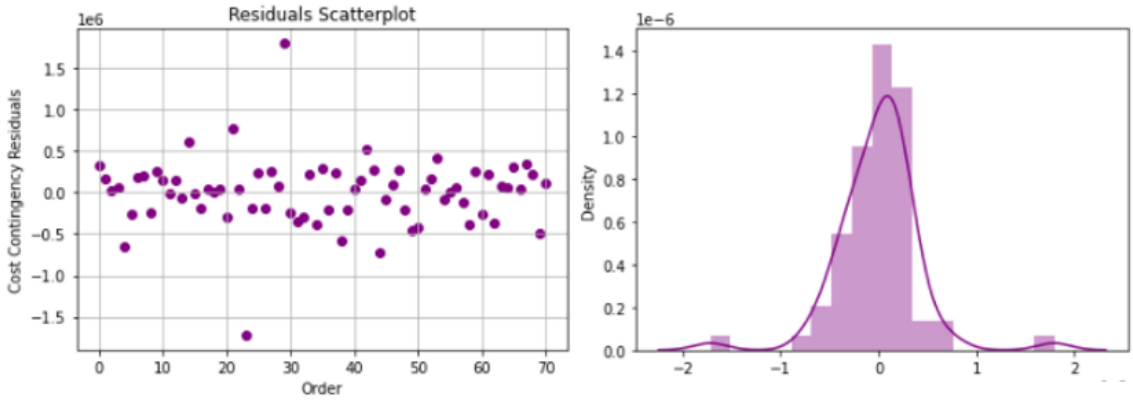

Figure 4-18. MLR Residual plot and histogram plot of residuals for the Cost Contingency Dimension

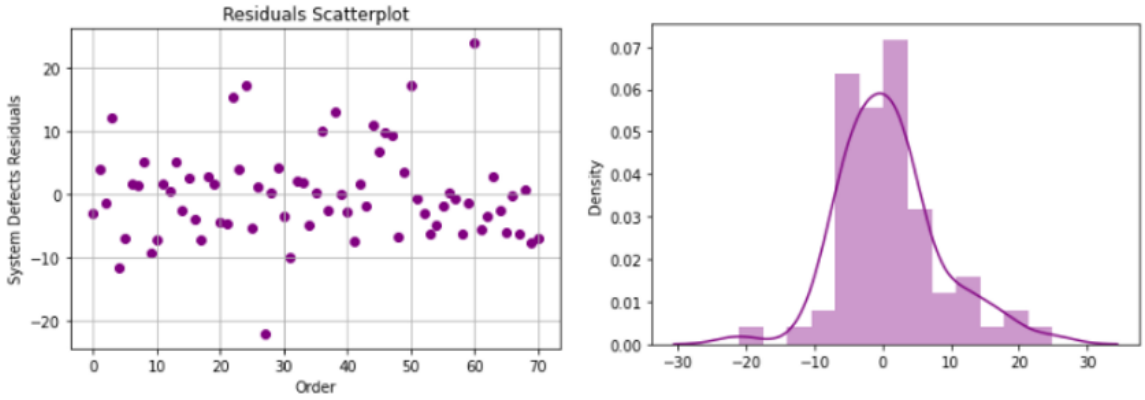

Figure 4-19. MLR Residual plot and histogram plot of residual for the System Defect Dimension

The MDPM was developed in a chained 3-step process as described in **Section 3.6** and **Figure 3-2**:

1. The First Step Trained Model (FSTM) predicts the schedule contingency dimension.
2. The Second Step Trained Model (SSTM) predicts the cost contingency dimension.
3. The Third Step Trained Model (TSTM) predicts the system defects dimension.

For each step of the MDPM model, **Figures 4-20, 4-21, and 4-22** compare the predicted values with the true values, respectively, during the testing phase for the Multiple Linear Regression model. Discussion of these figures will be included in **Section 5.1.2**.

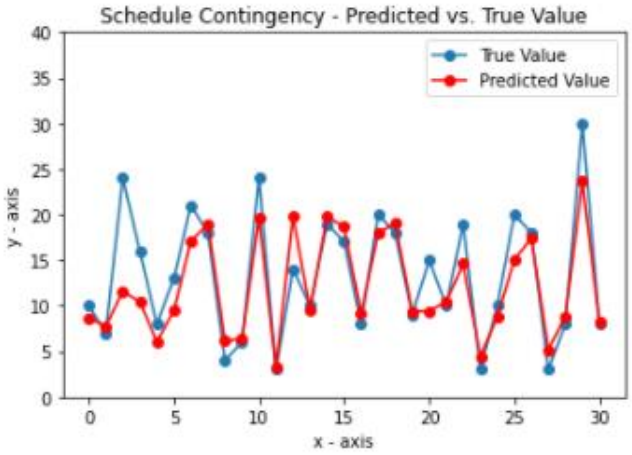

Figure 4-20. MLR Predicted vs. True Value for the Schedule Contingency Dimension

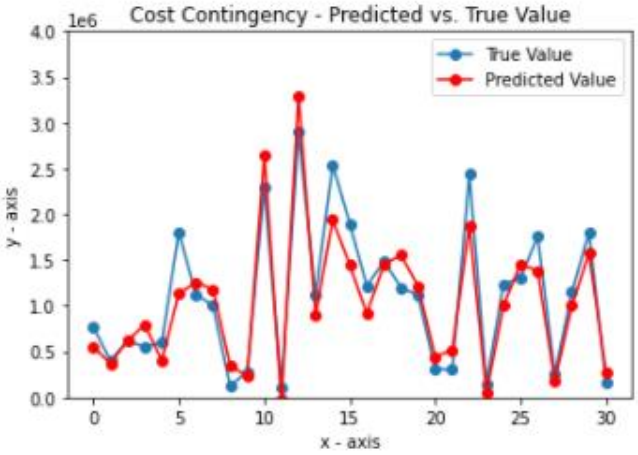

Figure 4-21. MLR Predicted vs. True Value for the Cost Contingency Dimension

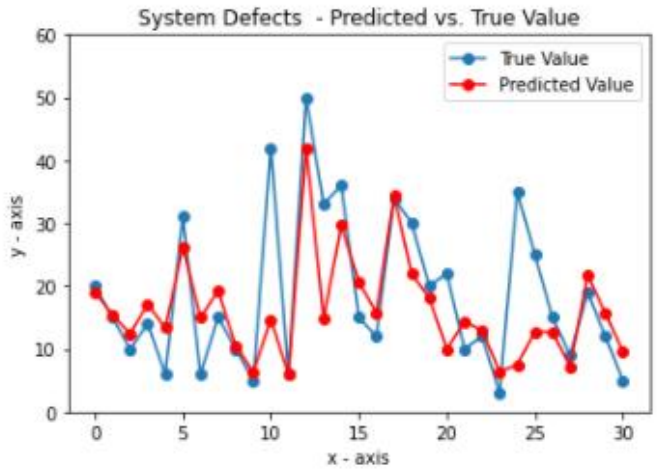

Figure 4-22. MLR Predicted vs. True Value for the System Defects Dimension

In addition to the predicted versus true value plots, **Figures 4-23, 4-24, and 4-25** include the run order residual plot, the residual histogram and the distribution curve in each of the dimensions in the testing phase, respectively. Discussion of these figures will be included in **Section 5.1.2**.

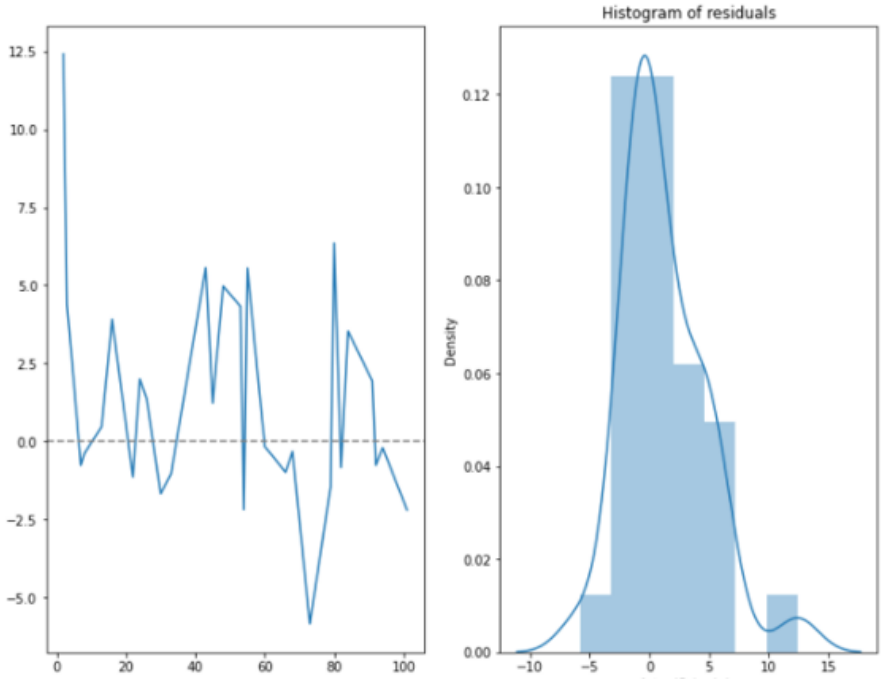

Figure 4-23. MLR Residual plot and histogram plot of residual for the Schedule Contingency Dimension

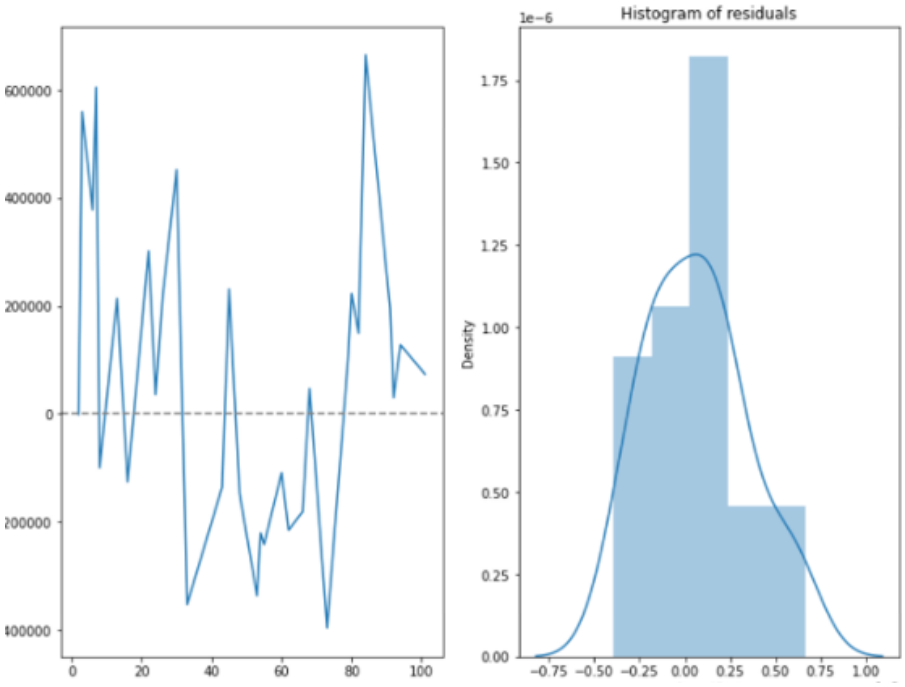

Figure 4-24. MLR Residual plot and histogram plot of residual for the Cost Contingency Dimension

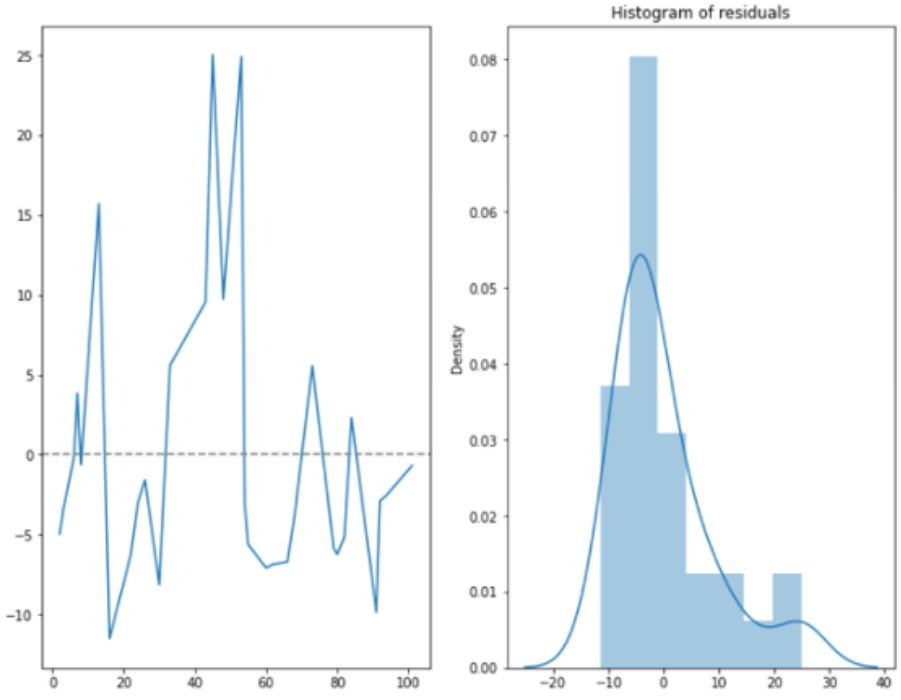

Figure 4-25. MLR Residual plot and histogram plot of residual for the System Defect Dimension

Using the **Equations 3-2** and **3-4** as described in **Section 3.8**, **Table 4-3** shows the model performance results for each output dimension of schedule contingency, cost contingency, and system defects, and the final combined MDPM model for the Multiple Linear Regression model using Ordinary Least Squares (OLS) and Multiple Linear Regression model using Least Absolute Shrinkage and Selection Operation (LASSO). Discussion of these results will be presented in **Section 5.1.2**.

Table 4-3. MLR Model Performance Results

Dimensions	Model Performance Evaluation Metrics			
	OLS		LASSO	
	NMAE	NRMSE	NMAE	NRMSE
Schedule Contingency	9.43%	13.51%	9.26%	12.67%
Cost Contingency	8.26%	10.16%	8.26%	10.16%
System Defects	15.13%	20.09%	15.09%	19.91%
Multi-Dimensional	10.94%	14.59%	10.87%	14.25%

4.5 Hypothesis 3 and Results

H3: *A Decision Tree model can be developed to predict multi-dimensional project outcomes with NMAE and NRMSE to be less than or equal to 20%.*

The MDPM is a chained 3-step process with the FSTM predicting the schedule contingency dimension, SSTM predicting the cost contingency dimension, and the TSTM predicting the system defects dimension. **Figures 4-26, 4-27, and 4-28** compare the predicted values with the true values in each of the dimension during the testing phase for the Decision Tree model. Discussion of these results is presented in **Section 5.1.3**.

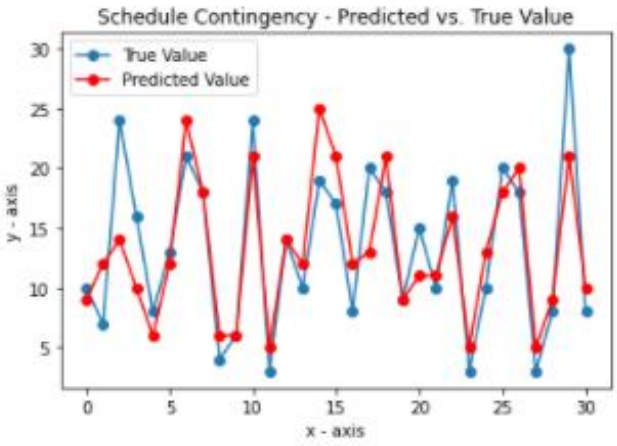

Figure 4-26. DT Predicted vs. True Value for the Schedule Contingency Dimension

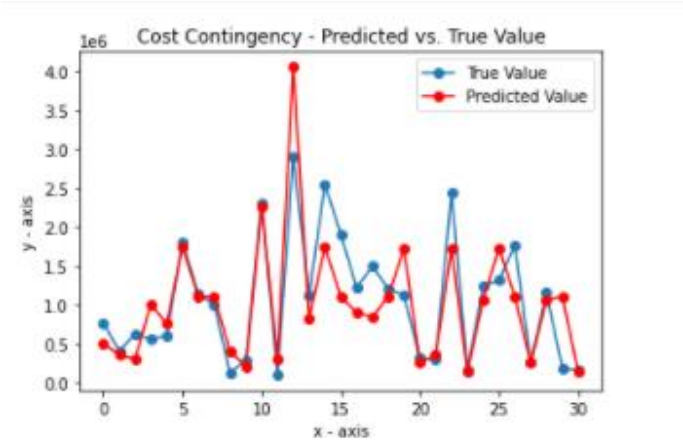

Figure 4-27. DT Predicted vs. True Value for the Cost Contingency Dimension

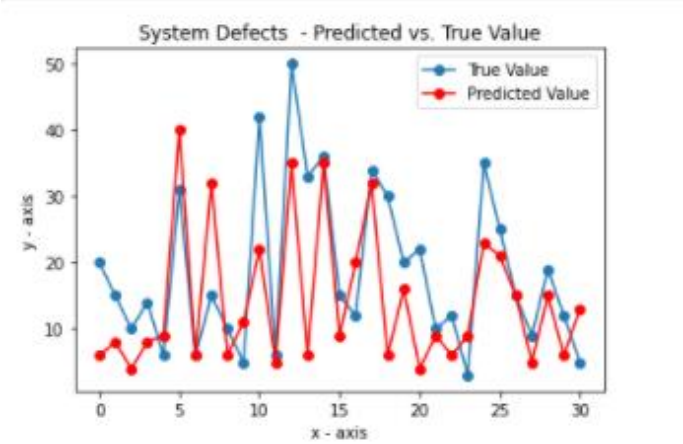

Figure 4-28. DT Predicted vs. True Value for the System Defects Dimension

Figures 4-29, 4-30, and 4-31 include the run order residual plot, the residual histogram and distribution curve in each dimension, respectively. Discussion of these results is described in **Section 5.1.3**.

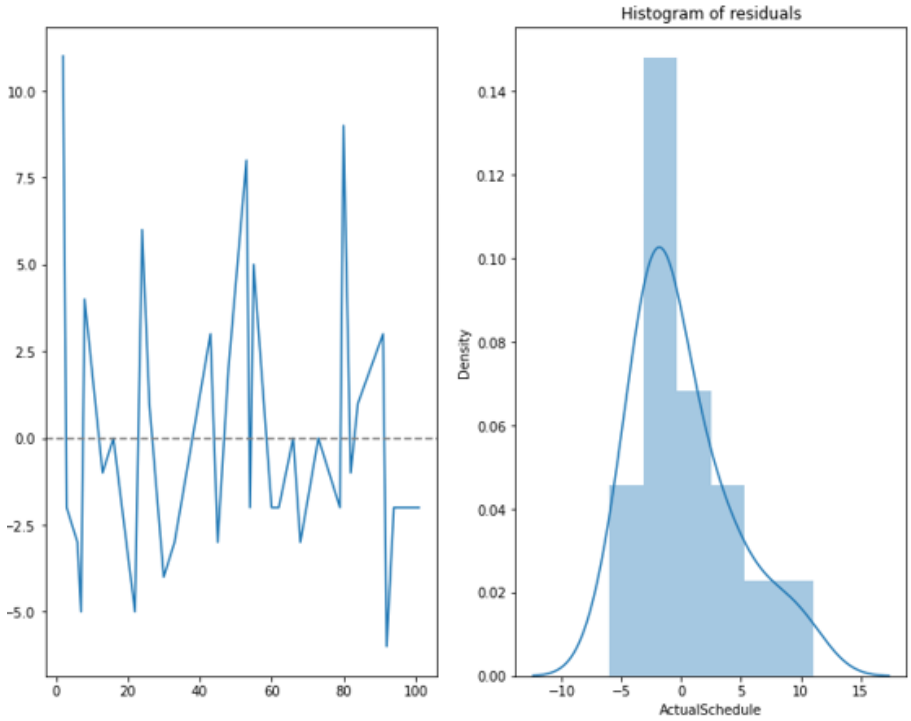

Figure 4-29. DT Residual plot and histogram plot of residual for the Schedule Contingency Dimension

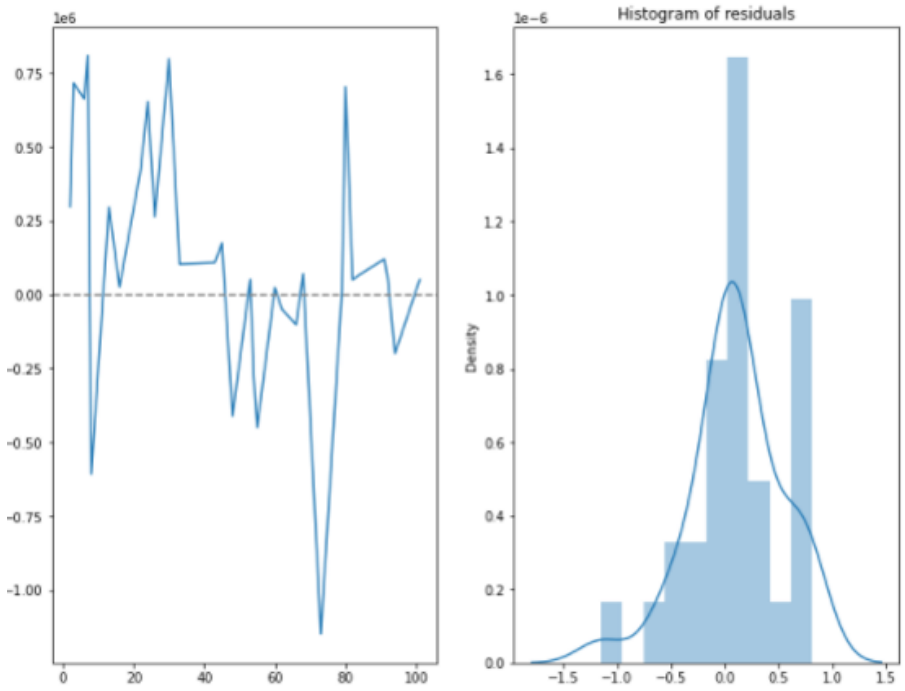

Figure 4-30. DT Residual plot and histogram plot of residual for the Cost Contingency Dimension

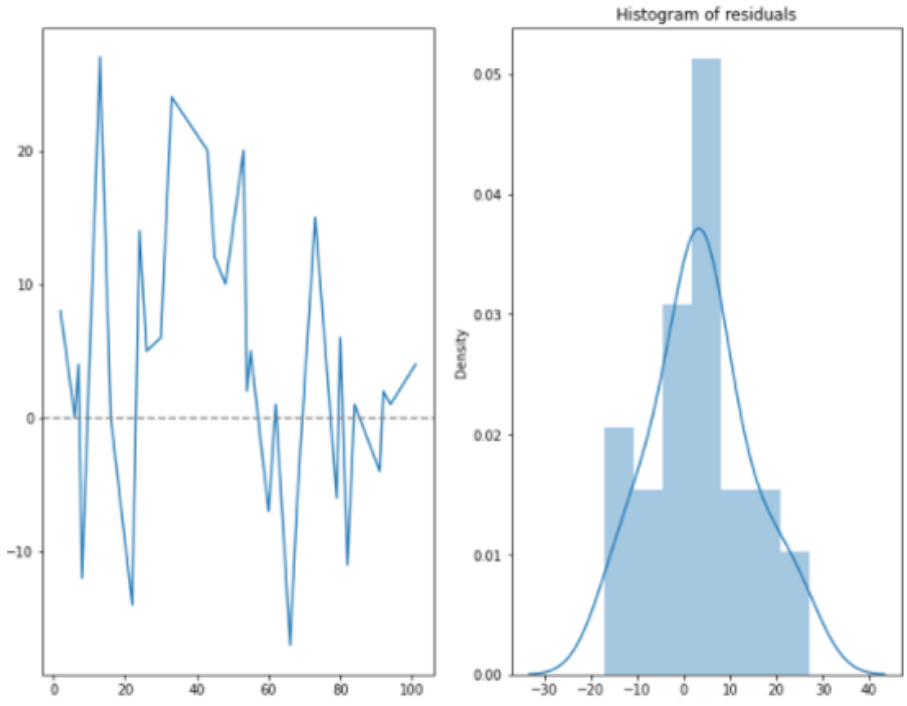

Figure 4-31. DT Residual plot and histogram plot of residual for the System Defects Dimension

Using the **Equations 3-2** and **3-4** as described in **Section 3.8**, **Table 4-4** displays the model performance results for each output dimension and the final combined MDPM model for the Decision Tree model. Discussion of these results will be presented in **Section 5.1.3**.

Table 4-4. DT Model Performance Results

Dimensions	Model Performance Evaluation Metrics	
	NMAE	**NRMSE**
Schedule Contingency	12.78%	16.97%
Cost Contingency	11.52%	17.94%
System Defects	19.15%	25.92%
Multi-Dimensional	14.48%	20.28%

4.6 Hypothesis 4 and Results

H4: *A Random Forest model can be developed to predict multi-dimensional project outcomes with NMAE and NRMSE to be less than or equal to 20%.*

The MDPM predicts the schedule contingency, cost contingency and the number of system defects following the FSTM, SSTM, and TSTM. **Figures 4-32, 4-33, and 4-34** compare the predicted values with the true values in each of the dimension during the testing phase for the Random Forest model. Discussion of these comparisons is presented in **Section 5.1.4**.

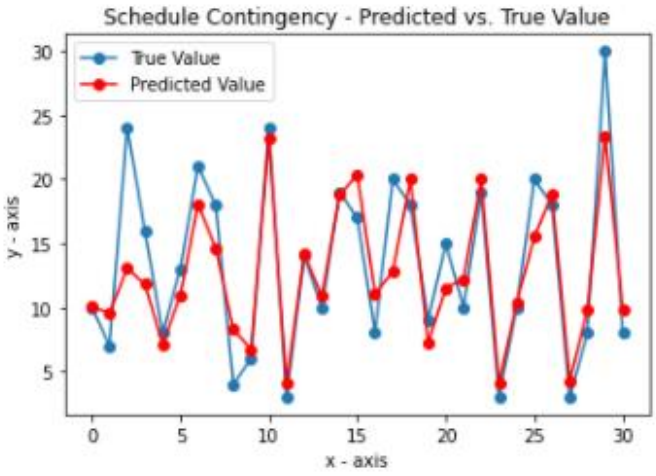

Figure 4-32. RF Predicted vs. True Value for the Schedule Contingency Dimension

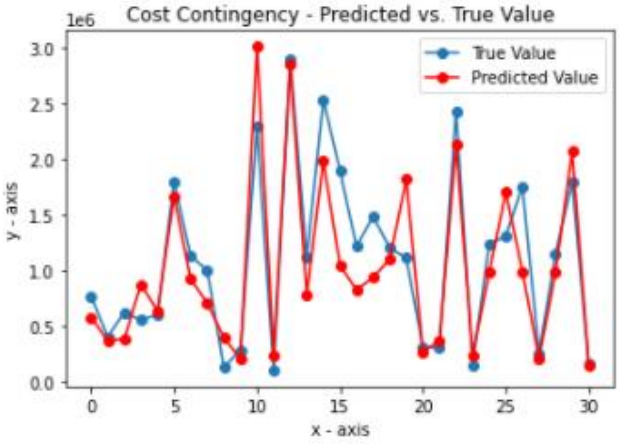

Figure 4-33. RF Predicted vs. True Value for the Cost Contingency Dimension

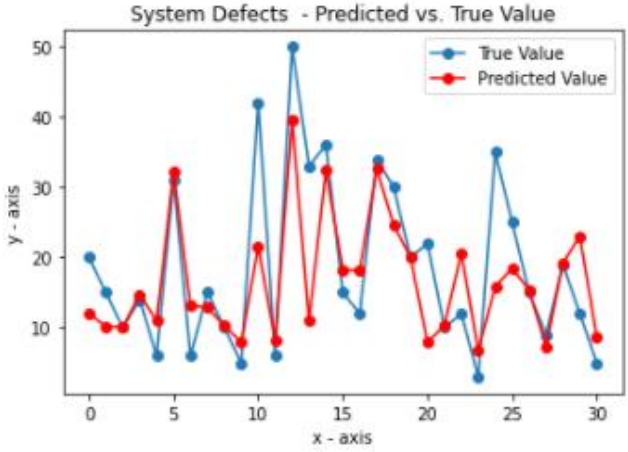

Figure 4-34. RF Predicted vs. True Value for the System Defects Dimension

Figures 4-35, 4-36, and 4-37 include the run order residual plot, the residual histogram and distribution curve in each dimension, respectively. Discussion of these figures is presented in **Section 5.1.4**.

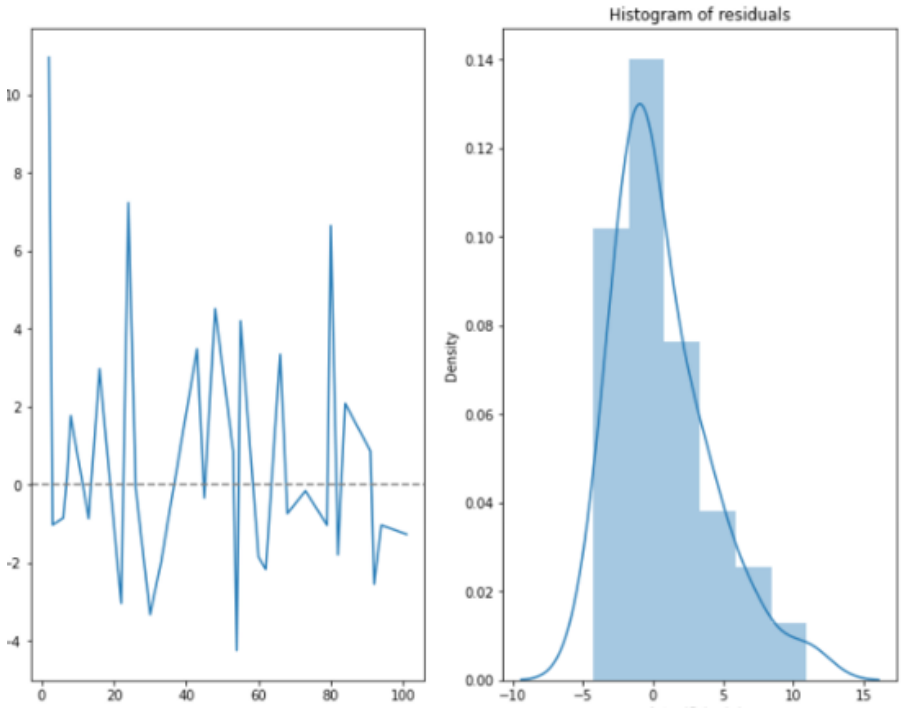

Figure 4-35. RF Residual plot and histogram plot of residual for the Schedule Contingency Dimension

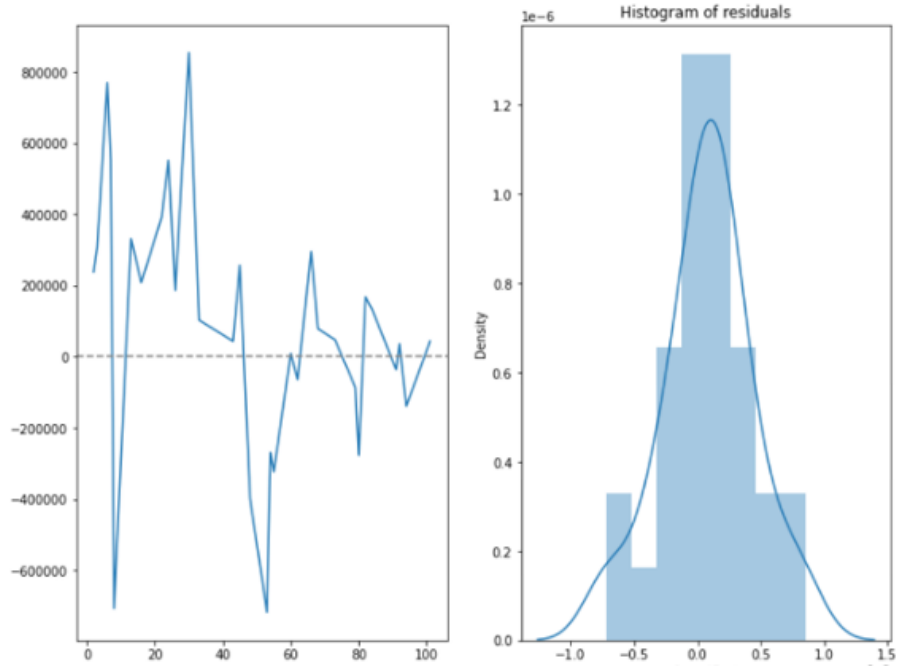

Figure 4-36. RF Residual plot and histogram plot of residual for the Cost Contingency Dimension

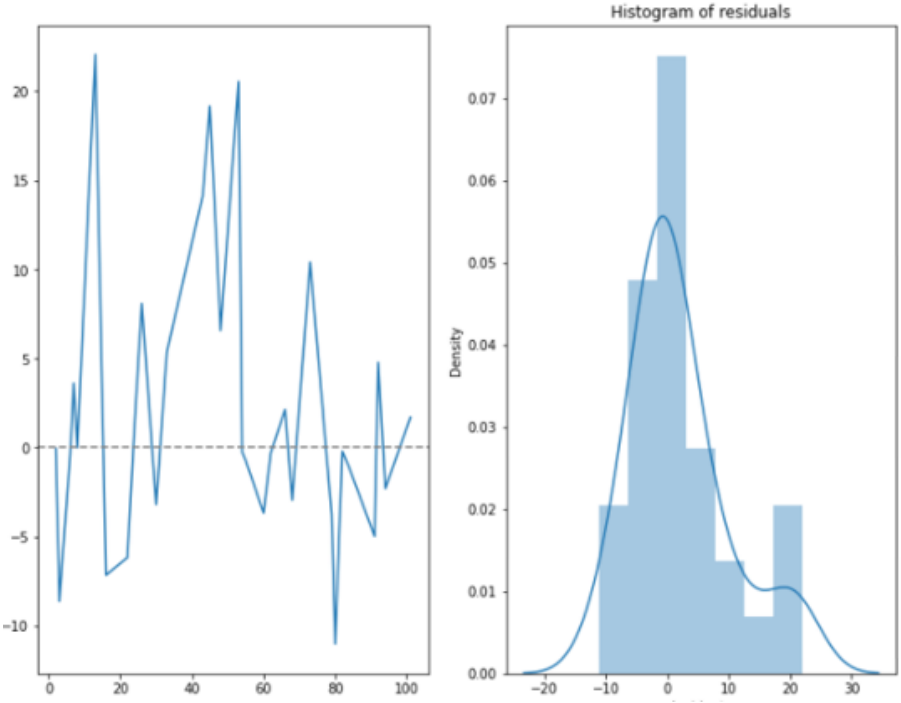

Figure 4-37. RF Residual plot and histogram plot of residual for the Cost Contingency Dimension

Using the **Equations 3-2** and **3-4** as described in **Section 3.8**, **Table 4-5** displays the model performance results for each output dimension and the final combined MDPM model for the Random Forest model. Discussion of these results is presented in **Section 5.1.4**.

Table 4-5. RF Model Performance Results

Dimensions	Model Performance Evaluation Metrics	
	NMAE	**NRMSE**
Schedule Contingency	9.86%	12.96%
Cost Contingency	10.44%	13.82%
System Defects	12.74%	17.60%
Multi-Dimensional	11.01%	14.79%

4.7 Hypothesis 5 and Results

H5: *A Neural Network model can be developed to predict multi-dimensional project outcomes with NMAE and NRMSE to be less than or equal to 20%.*

The MDPM predicts the schedule contingency, cost contingency and the number of system defects. **Figures 4-38, 4-39, and 4-40** compare the predicted values with the true values in each of the dimension during the testing phase for the Neural Network model. Discussion of these comparisons is presented in **Section 5.1.5**.

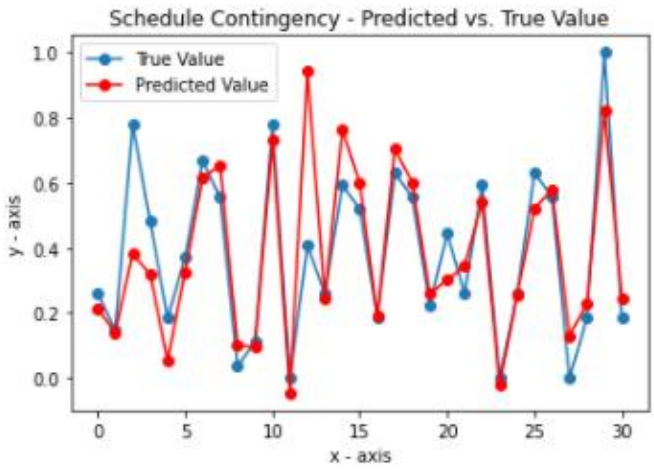

Figure 4-38. NN Predicted vs. True Value for the Schedule Contingency Dimension

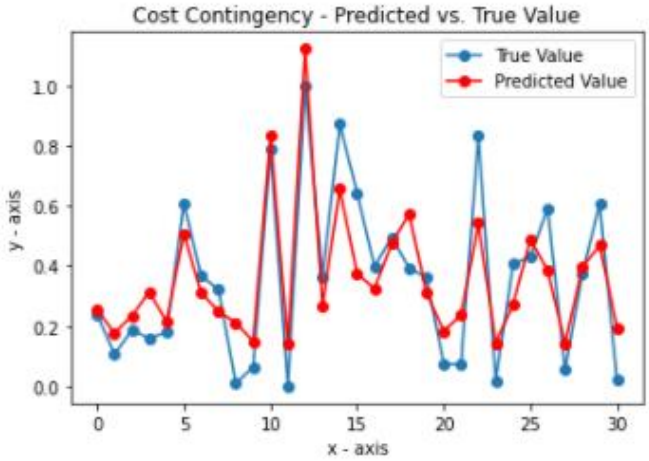

Figure 4-39. NN Predicted vs. True Value for the Cost Contingency Dimension

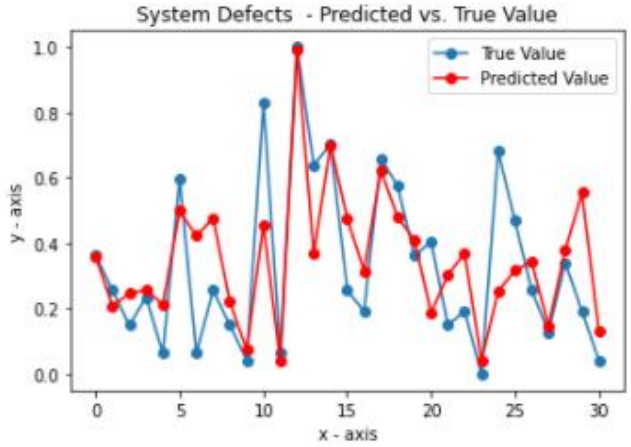

Figure 4-40. NN Predicted vs. True Value for the System Defects Dimension

Figures 4-41, 4-42, and 4-43 include the run order residual plot, the residual histogram and distribution curve in each dimension, respectively. Discussion of these figures is presented in **Section 5.1.5**

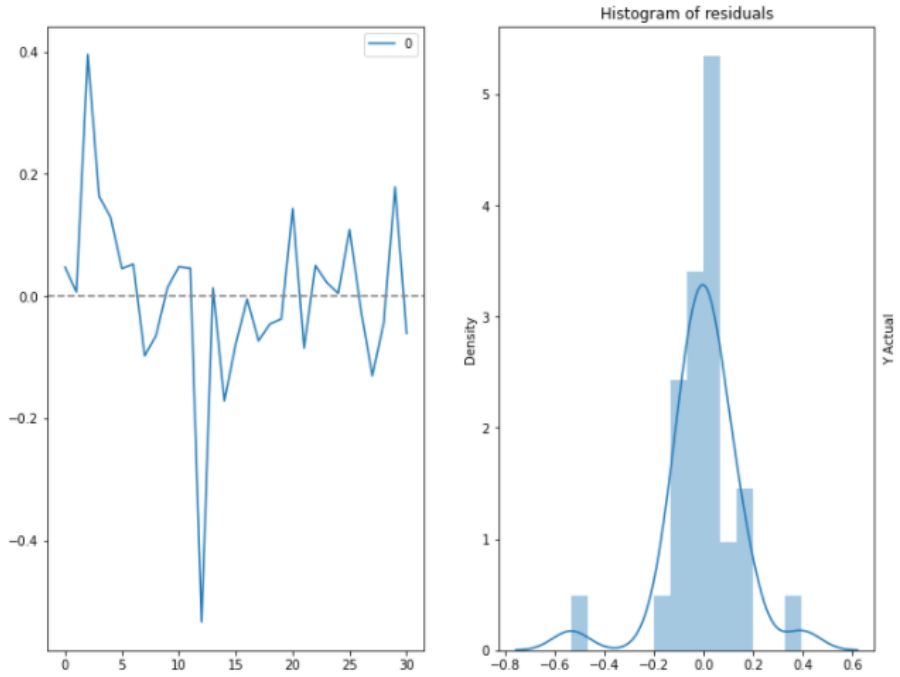

Figure 4-41. NN Residual plot and histogram plot of residual for the Schedule Contingency Dimension

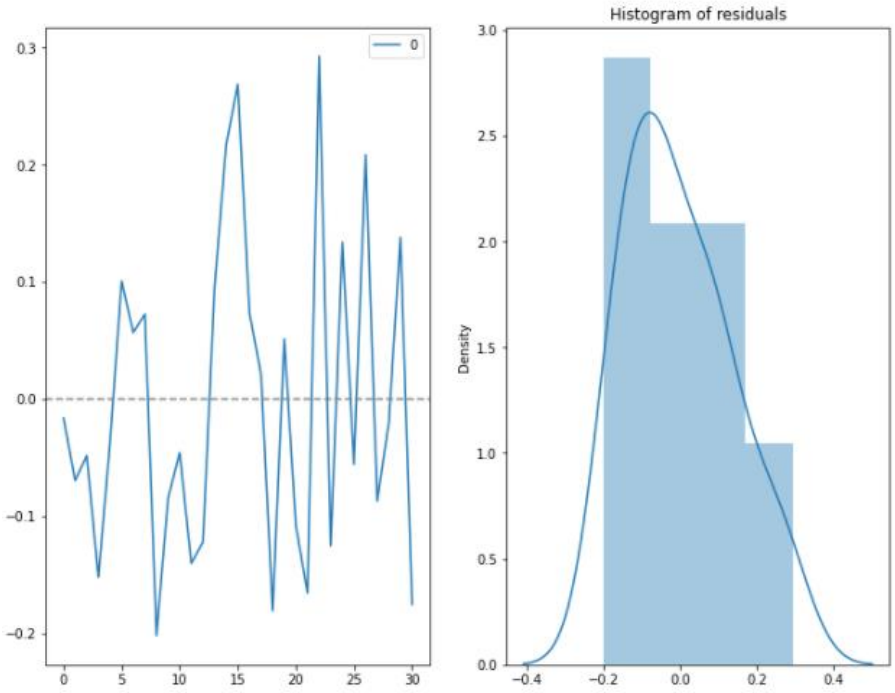

Figure 4-42. NN Residual plot and histogram plot of residual for the Cost Contingency Dimension

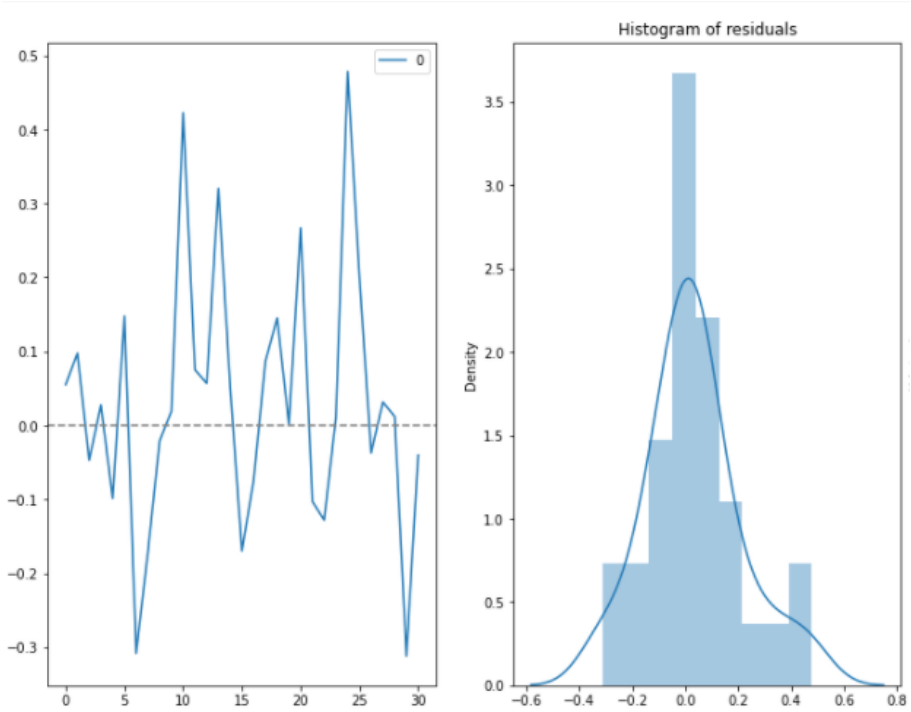

Figure 4-43. NN Residual plot and histogram plot of residual for the System Defects Dimension

Using the **Equations 3-2** and **3-4** as described in **Section 3.8**, **Table 4-6** displays the model performance results for each output dimension and the final combined MDPM model for the Neural Network model. Discussion of these result will be presented in **Section 5.1.5**.

Table 4-6. NN Model Performance Results

Dimensions	Model Performance Evaluation Metrics	
	NMAE	**NRMSE**
Schedule Contingency	9.42%	14.5%
Cost Contingency	13.86%	18.16%
System Defects	14.07%	18.62%
Multi-Dimensional	12.45%	17.09%

4.8 Model Validation

Histograms were generated for all variables in the data set to validate that the construct of selected projects is representative of all types of software projects in order to minimize selection bias. **Figure 4-44** displays the histogram generated from Python code using the Google Co-laboratory development environment. These histograms validated that the selected data set consists of a diverse group of software projects of different sizes, types, and constraints.

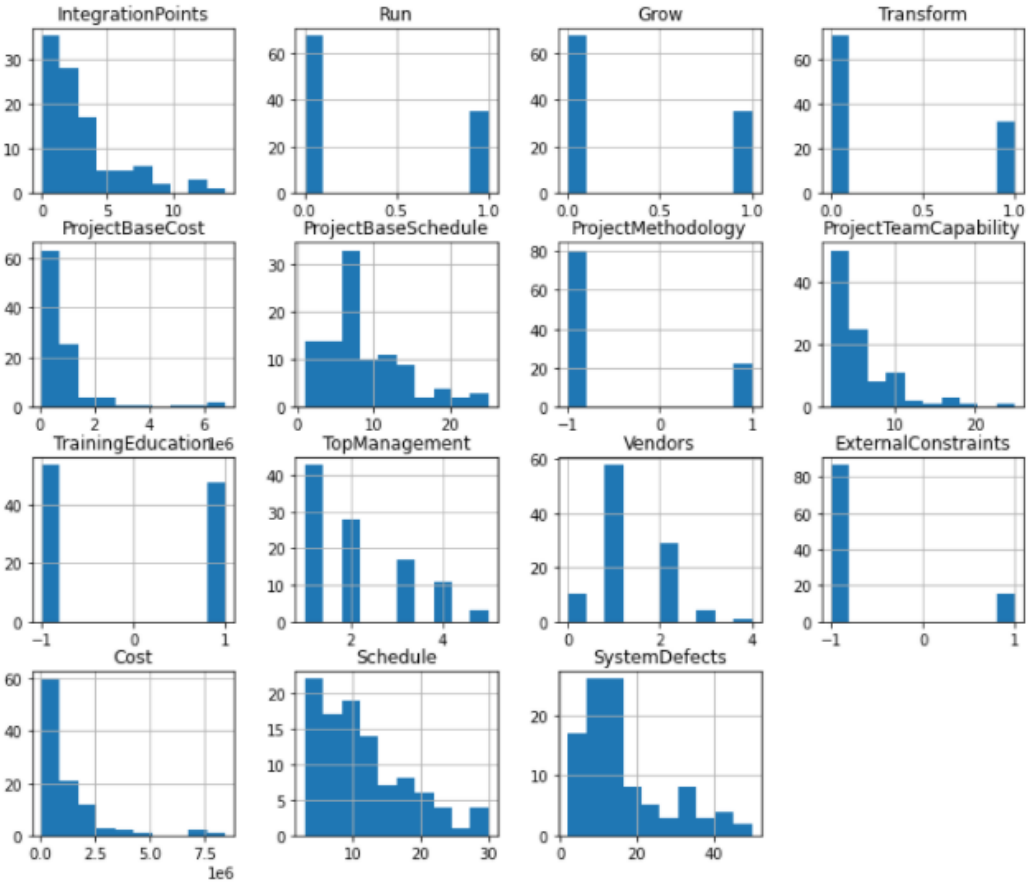

Figure 4-44. Data Input Histogram

4.8.1 Multiple Linear Regression Validation Results

As discussed in **Section 3.7**, an important model validation methodology in machine learning is cross-validation. In the Multiple Linear Regression model, 10-fold cross validation was performed using the training set which contained 70% of the data set. **Figures 4-45, 4-46, 4-47** are the box plots of the validation scores for the schedule contingency, cost contingency, and system defects dimensions, respectively. Based on the box plots, the distribution of the validation scores was satisfactory in all dimensions; there was only one outlier in the cost contingency dimension. In addition to cross validation, model performance results using **Equations 3-2 and 3-4** were applied to validate the accuracy of the machine learning models using the testing set which

contained the remaining 30% of the data set. The accuracy results are presented in **Table 4-3**. The evaluation of the results for the Multiple Linear Regression model suggests that the model provide sufficient validation and acceptable behavior.

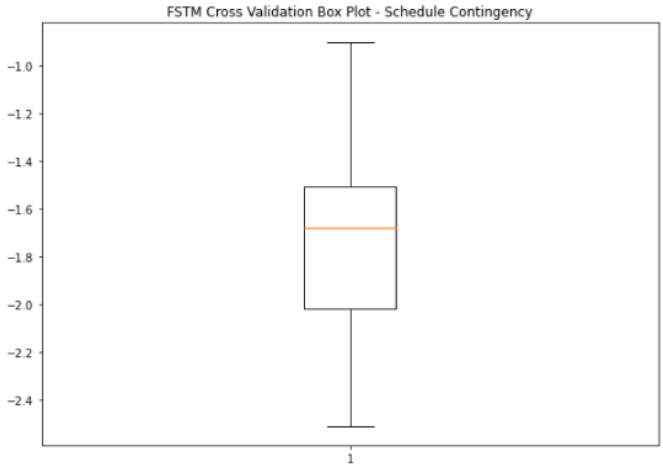

Figure 4-45. MLR Cross Validation Box Plot for the Schedule Contingency Dimension

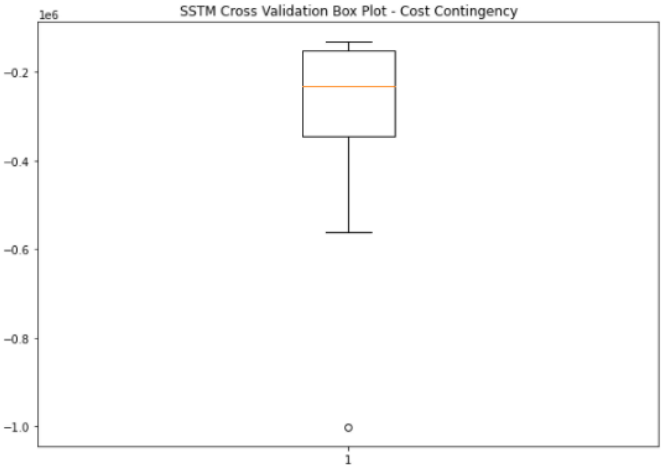

Figure 4-46. MLR Cross Validation Box Plot for the Cost Contingency Dimension

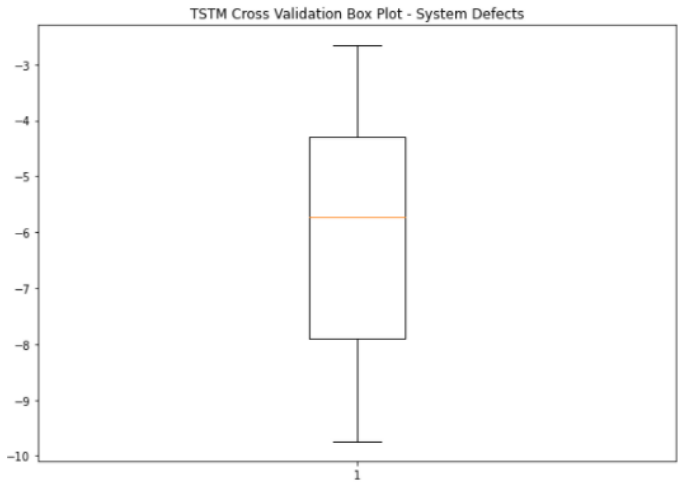

Figure 4-47. MLR Cross Validation Box Plot for the System Defects Dimension

4.8.2 Decision Tree Validation Results

For the Decision Tree model, 10-fold cross validation was performed using the training set. **Figures 4-48, 4-49, 4-50** are the box plots of the validation scores for the schedule contingency, cost contingency, and system defects dimensions, respectively. Based on the box plots, the distribution of the validation scores were satisfactory in all dimensions; the distributions for the schedule contingency and cost contingency dimensions were negatively skewed but there were no outliers in all dimensions. In addition to cross validation, model performance results using **Equations 3-2 and 3-4** were applied to validate the accuracy of the machine learning models using the testing set. The accuracy results are presented in **Table 4-4**. The evaluation of the results for the Decision Tree model suggests that the model provides sufficient validation and acceptable behavior.

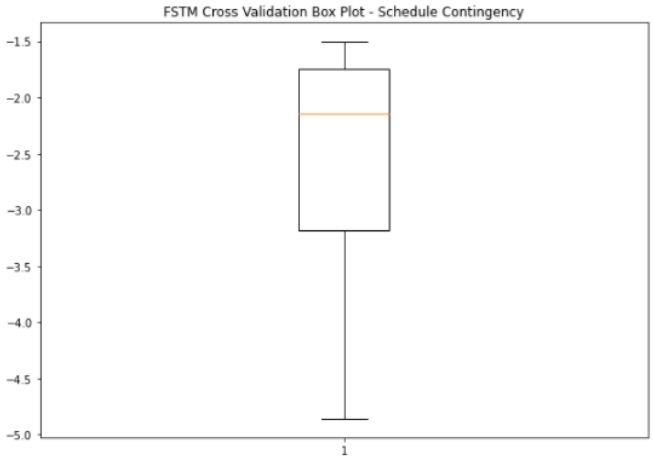

Figure 4-48. DT Cross Validation Box Plot for the Schedule Contingency Dimension

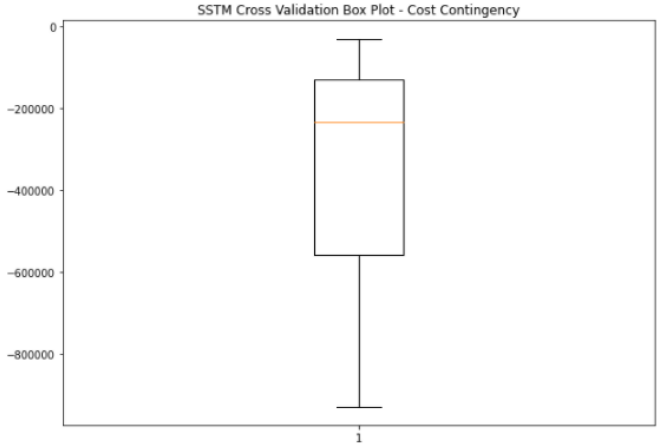

Figure 4-49. DT Cross Validation Box Plot for the Cost Contingency Dimension

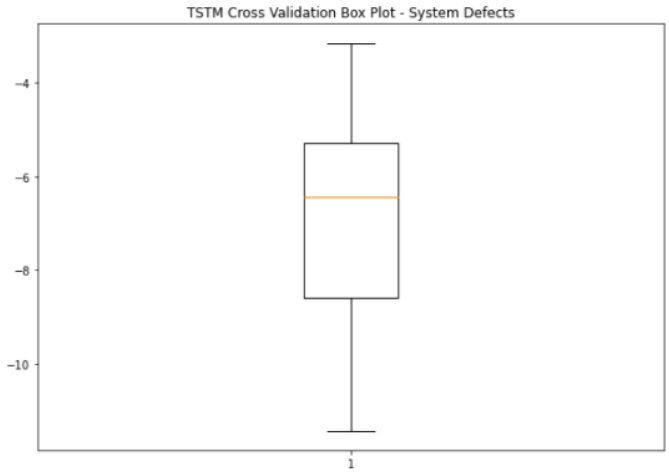

Figure 4-50. DT Cross Validation Box Plot for the System Defects Dimension

4.8.3 Random Forest Validation Results

For the Random Forest model, **Figures 4-51, 4-52, 4-53** are the box plots of the validation scores for the schedule contingency, cost contingency, and system defects dimensions, respectively. 10-fold cross validation was performed using the training set. Based on the box plots, the distribution of the validation scores were satisfactory in all dimensions; the distributions for the schedule contingency and cost contingency dimensions were negatively skewed but there were no outliers in all dimensions. In addition to cross validation, model performance results using **Equations 3-2 and 3-4** were applied to validate the accuracy of the machine learning models using the testing set. The accuracy results are presented in **Table 4-5**. The results for the Random Forest model suggest that the model provides sufficient validation and acceptable behavior.

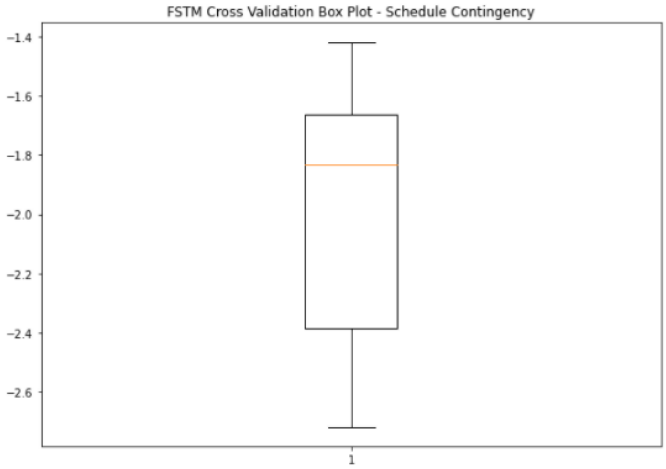

Figure 4-51. RF Cross Validation Box Plot for the Schedule Contingency Dimension

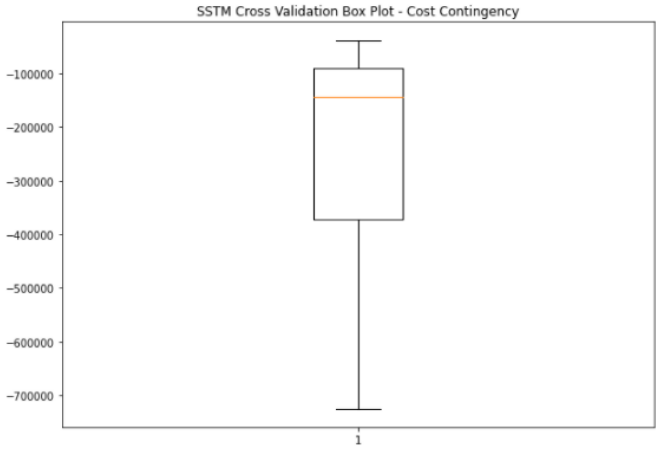

Figure 4-52. RF Cross Validation Box Plot for the Cost Contingency Dimension

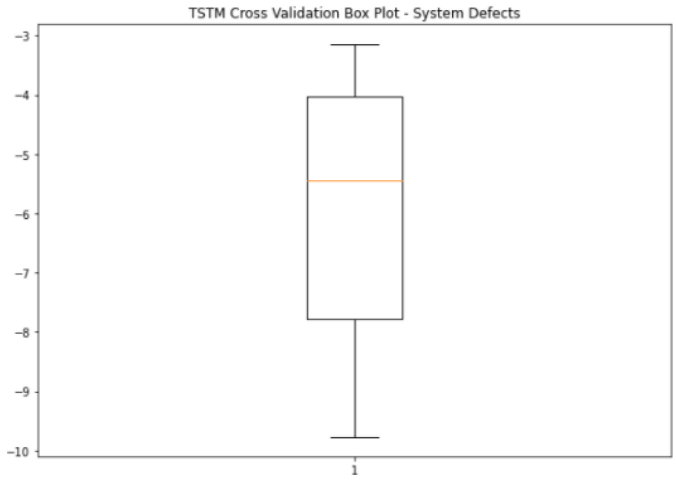

Figure 4-53. RF Cross Validation Box Plot for the System Defects Dimension

4.8.4 Neural Network Validation Results

For the Neural Network model, the *Keras* (Chollet, 2015) library in Python programming language as described in **Section 3.6.1** has pre-defined functions to perform validation and evaluate performance of the model using the validation subset. As described in **Section 3.7**, 20% of the training data is separated into a validation data set. **Figures 4-54, 4-55, 4-56** display the loss functions for both the training and validation data sets for the schedule contingency, cost contingency, and system defects dimensions, respectively. The loss function indicates how well the model is fitting the data. Evaluating and comparing the loss functions of the training data set and the validation data set over the 500 epochs in **Figures 4-54, 4-55, 4-56** validates that the model is not overfitting in all dimensions and provides an acceptable behavior.

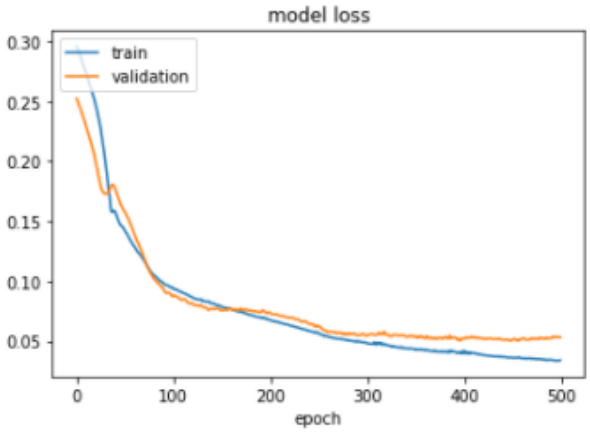

Figure 4-54. NN Validation Model Loss for the Schedule Contingency Dimension

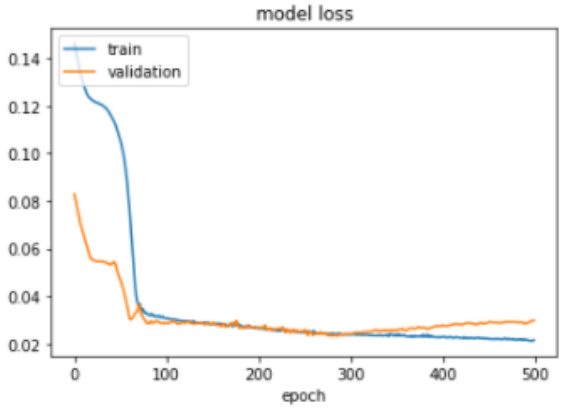

Figure 4-55. NN Validation Model Loss for the Cost Contingency Dimension

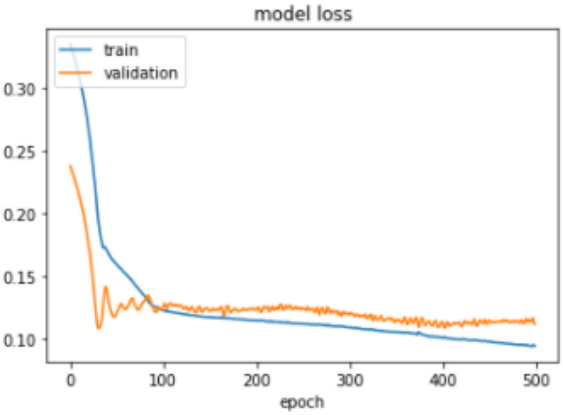

Figure 4-56. NN Validation Model Loss for the System Defects Dimension

Chapter 5—Discussion and Conclusions

5.1 Discussion

The results presented in **Chapter 4** suggest that the identified Critical Success Factors (CSFs) and the selected machine learning methods could be used for software project estimation to improve project management performance and reduce the likelihood of failed projects. Literature review in **Chapter 2** reveals that many approaches have been attempted to solve the problem of software projects being over budget, late or lacking the required functionality. However, previous literature focused primarily on predicting software performance in a single dimension, and did not emphasize the importance of the selection of CSFs that tailor to the different types of software projects and industries. In this praxis, 10 CSFs were identified as having significant impact on the project outcome, and the machine learning model predicted three output dimensions of schedule contingency, cost contingency and number of system defects.

The following sections review the results as they relate to the research questions and hypotheses. **Sections 5.1.1–5.1.6** discuss the hypotheses and the results from **Sections 4.3-4.7**. **Section 5.2** presents the conclusions and **Section 5.2.1** offers lessons learned from analyzing the results. **Section 5.3** suggests how this research contributes to the current body of knowledge, and **Section 5.4** recommends opportunities for future research.

5.1.1 Discussion of Hypothesis 1

H1: *The Critical Success Factors (Independent Variables) identified in this praxis have significant influence on the dependent variables.*

Hypothesis 1 was tested using 102 software projects obtained from a Canadian large-size client organization that implemented software projects in the Energy Sector from 2016 to 2020 using 10 CSFs as inputs and the multi-dimensional output variables. The results presented in **Section 4.3.5** suggest the identified CSFs could be used for multi-dimensional prediction of software projects. The top five CSFs with the most significant influence on the output variables were Integration of the System, Project Base Cost, Project Base Schedule, Project Team Capability, and Top Management Support. The remaining five CSFs (Technical Model, Project Methodology, Training and Education, Vendor Partnership, External Constraints) were features that also contributed to the prediction. Findings reveal that numeric features contributed more significantly to the multi-dimensional prediction than the categorical features. Results in **Section 4.3** suggest that in the Random Forest model, the features with high importance calculated using SHAP values were concentrated in the top five CSFs, whereas in the Neural Network model, feature importance values were more evenly distributed among all the CSFs.

Result: Hypothesis confirmed.

5.1.2 Discussion of Hypothesis 2

H2: *A Multiple Linear Regression model can be developed to predict multi-dimensional project outcomes with Normalized Mean Absolute Error (NMAE) and Normalized Root Mean Squared Error (NRMSE) to be less than or equal to 20%.*

Hypothesis 2 was tested using 102 software projects obtained from a Canadian large-size client organization that implemented software projects in the Energy Sector from 2016 to 2020 using 10 CSFs as inputs and the multi-dimensional output variables.

Figures 4-19, 4-20, and **4-21** illustrate that the predicted values were close to the true value with a few projects having a larger variance. The gap between the predicted values and true values was highest in the System Defects dimension as shown in **Figure 4-21**. **Figures 4-22, 4-23**, and **4-24** suggest that the residuals had random variance and the distribution had a mean close to 0. In **Figure 4-22**, outliers were observed for the Schedule Contingency dimension. Results in **Table 4-2** suggest that Multiple Linear Regression model using Ordinary Least Squares (OLS) can accurately predict multi-dimensional project outcomes with NMAE and NRMSE of less than 20%. Multiple Linear Regression model using Least Absolute Shrinkage and Selection Operation (LASSO) only improved the prediction accuracy by an insignificant percentage of less than 1%. Observing the errors in each dimension, System Defects dimension had the largest error percentages with NMAE of 15.13% and NRMSE of 20.09% using the OLR approach. Cost Contingency dimension had the lowest error percentages with NMAE of 8.26% and NRMSE of 10.16% using the OLR approach.

Result: Hypothesis confirmed.

5.1.3 Discussion of Hypothesis 3

H3: *A Decision Tree model can be developed to predict multi-dimensional project outcomes with NMAE and NRMSE to be less than or equal to 20%.*

Hypothesis 3 was tested using 102 software projects obtained from a Canadian large-size client organization that implemented software projects in the Energy Sector from 2016 to 2020 using 10 CSFs as inputs and the multi-dimensional output variables. **Figures 4-25, 4-26,** and **4-27** illustrate that there were variances between the predicted and the true values. The gap between the predicted values and true values was highest in

the System Defects dimension as shown in **Figure 4-27**. **Figures 4-28, 4-29**, and **4-30** suggest that the residuals had random variance and the distribution had a mean close to 0. In **Figure 4-29**, outliers were observed for the Cost Contingency dimension. Results in **Table 4-3** suggest that Decision Tree cannot accurately predict multi-dimensional project outcomes with NMAE and NRMSE of less than or equal to 20%. The calculated multi-dimensional NRMSE was 20.28%. Observing the errors in each dimension, the error percentages were fairly high in all dimensions, especially in the System Defects dimension with a NMAE of 19.15% and NRMSE of 25.92%.

Result: Hypothesis not confirmed.

5.1.4 Discussion of Hypothesis 4

H4: *A Random Forest model can be developed to predict multi-dimensional project outcomes with NMAE and NRMSE to be less than or equal to 20%.*

Hypothesis 4 was tested using 102 software projects obtained from a Canadian large-size client organization that implemented software projects in the Energy Sector from 2016 to 2020 using 10 CSFs as inputs and the multi-dimensional output variables. **Figures 4-31, 4-32,** and **4-33** illustrate that the predicted values were close to the true value with a few projects having larger variances. The gap between the predicted values and true values was highest in the System Defects Dimension as shown in **Figure 4-33**. **Figures 4-34, 4-35**, and **4-36** suggest that the residuals had random variance and the distribution had a mean close to 0. In **Figure 4-36**, outliers were observed for the System Defects dimension. Results in **Table 4-4** suggest Random Forest can accurately predict multi-dimensional project outcomes with NMAE and NRMSE of less than 20%.

Observing the errors in **Table 4-4**, the NMAE ranged between 9% to 13% and the NRMSE ranged between 12% and 18% in all dimensions.

Result: Hypothesis confirmed.

5.1.5 Discussion of Hypothesis 5

H5: *A Neural Network model can be developed to predict multi-dimensional project outcomes with NMAE and NRMSE to be less than or equal to 20%.*

Hypothesis 5 was tested using 102 software projects obtained from a Canadian large-size client organization that implemented software projects in the Energy Sector from 2016 to 2020 using 10 CSFs as inputs and the multi-dimensional output variables. **Figures 4-37, 4-38,** and **4-39** illustrate that the predicted values were close to the true value with a few projects having larger variances. The gap between the predicted values and true values was highest in the System Defects Dimension as shown in **Figure 4-39**. **Figures 4-40, 4-41**, and **4-42** suggest that the residuals had random variance and the distribution had a mean close to 0. In **Figures 4-40 and 4-42**, outliers were observed for the Schedule Contingency and System Defect dimensions. Results in **Table 4-5** suggest Neural Network can accurately predict multi-dimensional project outcomes with NMAE and NRMSE of less than 20%. Observing the errors in each dimension, the error percentages were fairly consistent in all dimensions with the highest NMAE at 14.07% and lowest at 9.42%, and the highest NRMSE at 18.62% and lowest at 14.50%.

Result: Hypothesis confirmed.

5.1.6 Discussion of Model Comparison

Three out of the four machine learning models produced reasonable predictions on a multi-dimensional perspective with an acceptable degree of accuracy and a strong

correlation between the actual and predicted. The Multiple Linear Regression and Random Forest were identified as the top two machine learning models with excellent predictive performance and low error rates. Multiple Linear Regression had the lowest error rate with an overall NMAE and NRMSE of 10.94% and 14.59%, respectively, from a multi-dimensional perspective. As discussed in **Section 2.5.2**, Multiple Linear Regression is a simple and effective model and is often a preferable model because of the model's interpretability and low cost. However, even though the overall multi-dimensional error rate was low for the Multiple Linear Regression model, the prediction in the dimension of system defects had a high error percentage with a NRMSE of > 20%. As discussed in **Section 2.3.1**, the quality of the delivered scope is instrumental as it describes whether the end-product is functioning as designed without major defects (Jun et al., 2011).

Random Forest performed consistently in all dimensions as presented in **Table 4.4** and the overall multi-dimensional NMAE and NRMSE were 11.01% and 14.74%, respectively. Therefore, Random Forest is the most effective prediction model in this praxis. However, as discussed in **Section 2.5.6**, Random Forest model is considered a more complex model and lacks interpretability. More effort is required to provide explanations for the predictions to project sponsors and decision makers using this model in the organization's end-to-end business planning process (**Figure 3-1**).

5.2 Conclusions

In this research, we identified two research questions to help address the problem statement proposed in this research.

RQ1: What features are effective predictors of multi-dimensional project outcomes from a client perspective?

RQ2: Which proposed prediction model is the most effective in estimating multi-dimensional project outcomes based on prior project performance data?

Firstly, 10 CSFs (**Table 3-1**) were identified to train and test the four machine learning models (Multiple Linear Regression, Decision Tree, Random Forest, Neural Network). The findings discussed in **Sections 4.3** and **5.1.1** addressed our first research question and demonstrated that the selected CSFs were effective predictors of the multi-dimensional project outcomes from a client perspective. The second research question was addressed with the evaluation of the results detailed in **Sections 4.4 – 4.7** and discussion of model comparison in **Section 5.1.6** revealed that the Random Forest model was the most effective in estimating multi-dimensional project outcomes of project schedule contingency, cost contingency, and system defects based on prior project performance data. The following list presents the key conclusions of the models and results:

- The proposed Multi-Dimensional Prediction Model (MDPM) in **Figure 3-4** is a required step in the end-to-end business planning process (**Figure 3-1**) in an organization to support project managers and sponsors to make unbiased and informed decisions.

- The top five CSFs with the most significant influence on project performance were Integration of the System, Project Base Cost, Project Base Schedule, Project Team Capability, and Top Management Support.
- Multiple Linear Regression model was a simple and effective model, and it achieved the best overall multi-dimensional accuracy. However, when examining the system defects dimension independently, it had an unacceptable high error rate. Therefore, this method was not recommended as the preferred model in this praxis.
- Random Forest model achieved the second best overall multi-dimensional accuracy, and the model accuracy was consistent in all dimensions. This method was selected to be the most effective method in estimating multi-dimensional project outcomes based on prior project performance data in this praxis.

5.2.1 Lessons Learned

There were two key experiences learned from analyzing the data set and building the machine learning models during this praxis that should be incorporated into future analyses:

1. The data preprocessing step highlighted a gap in the data collection process with a subset of historical software projects having missing data or insufficient data quality. 102 out of a total of 208 projects were included in this praxis based on the criteria detailed in **Section 3.3**. Machine learning models are trained on data; thus, good data quality and sufficient data size are required to achieve the desired model outcomes. Organizations need to

emphasize the importance of data collection in the end-to-end business planning process (**Figure 3-1**), specifically during the steps of Business Case (Baseline), Project Execution, and System Production Sustainment.

2. The proposed MDPM aimed to support project sponsors in their strategic planning and to make calculated decisions early in the project lifecycle. The MDPM is an effective and unbiased tool that is required to address project complexity leading to a positive impact on the overall project performance. However, although the MDPM minimizes the human interaction needed, it cannot completely replace all human judgement in the decision-making process. Results in **Figures 4-22, 4-29, 4-36, 4-40, and 4-42** suggest that although the majority of the software projects had accurate predictions, there were a small number of outliers and exception cases.

5.3 Contributions to Body of Knowledge

The model developed in this praxis provides a multi-dimensional approach to support decision-making during the planning phase of a project and to improve project performance, specifically in the Canadian Energy Sector. By combining the optimum CSFs and machine learning models, the proposed MDPM predicts, within a 20% margin of error, the schedule and cost contingencies required to manage project uncertainties and risks, and the number of system defects required to deliver a quality end product. This model assists organizations to handle complexity in software systems, reduces the likelihood of projects becoming over budget, late or lacking the required functionality, and helps the organization to be financially successful.

5.4 Recommendations for Future Research

To further advance the body of knowledge with the goal of improving the multi-dimensional prediction performance, there are two main recommendations:

1. Future research should consider a multi-dimensional prediction model that extends beyond the Canadian market and the Energy Sector. Using a larger data set with more historical projects could help identify additional refinements to the selection of CSFs and machine learning models.
2. Future research should investigate the multi-dimensional problem using additional machine learning methods such as K-Nearest Neighbor (KNN) model and Support Vector Machines (SVM), which could contribute to the improvement of prediction accuracy.

References

AACE (American Association of Cost Engineers). (2000). AACE International's Risk Management Dictionary. *Cost Engineering, 42*(4), pp. 28–31.

Adadi, A. & Berrada, M. (2018). Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access, 6*, 52138-52160. https://doi.org/10.1109/ACCESS.2018.2870052

Adnams, S., Mok, L., & Curtis, D. (2018). *CIOs Can Use the Run-Grow-Transform Model to Align IT Functions and Roles to Strategic Business Priorities* (ID: G00325938). Gartner. https://www.gartner.com

Agamirzian, I. et. al. (2021). *Forecast: Enterprise IT Spending by Vertical Industry Market, Worldwide, 2019-2025, 1Q21 Update* (ID: G00744038). Gartner. https://www.gartner.com

Ahimbisibwe, A., Cavana, R.Y., & Daellenbach, U. (2015) A contingency fit model of critical success factors for software development projects: A comparison of agile and traditional plan-based methodologies. *Journal of Enterprise Information Management, 28*(1), 7-33. https://doi.org/10.1108/JEIM-08-2013-0060

Ahmed, F., Bouktif, S., Serhani, A., & Khalil, I. (2008). Integrating Function Point Project Information for Improving the Accuracy of Effort Estimation. *The Second International Conference on Advanced Engineering Computing and Applications in Sciences*, 193–198. https://doi.org/10.1109/advcomp.2008.42

Asheeri, M. M. & Hammad, M. (2019). Machine learning models for software cost estimation. *Paper presented at the - 2019 International Conference on Innovation*

and Intelligence for Informatics, Computing, and Technologies (3ICT), 1-6. https://doi.org/10.1109/3ICT.2019.8910327

Asiedu, R. O., Frempong, N. K., & Alfen, H. W. (2017). Predicting likelihood of cost overrun in educational projects. *Engineering, Construction, and Architectural Management, 24*(1), 21-39. https://doi.org/10.1108/ECAM-06-2015-0103

Baccarini, D. (2005). Understanding project cost contingency – a survey, presented at Cobra International Construction Conference, Brisbane, 2005. Brisbane, Qld: Queensland University of Technology.

Barraza, G., & Bueno, R. (2007). Cost contingency management. *Journal of Management in Engineering*, *23*(3), 140-146. https://doi.org/10.1061/(ASCE)0742-597X(2007)23:3(140)

Blanchard, B. S. (2003). *Logistics Engineering and Management 6th Edition*. Prentice Hall.

Boehm, B. et al. (2000) *Software Cost Estimation with COCOMO II*. Prentice-Hall.

Borchani, H., Varando, G., Bielza, C., & Larranaga, P. (2015). A survey on multi-output regression. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(5), 216–233. https://doi.org/10.1002/widm.1157

Bouayed Z. (2016). Using monte carlo simulation to mitigate the risk of project cost overruns. *International journal of safety and security engineering, 6*(2):293-300. https://doi.org/10.2495/SAFE-V6-N2-293-300

Bowen, D., & Ungar, L. (2020). Generalized SHAP: Generating Multiple Types of Explanations in Machine Learning. *arXiv preprint arXiv:2006.07155*

Broniatowski, D., & Tucker, C. (2017). Assessing Causal Claims about Complex Engineered Systems with Quantitative Data: Internal, External, and Construct Validity. *Systems Engineering, 20,* 483-496. https://doi.org/10.1002/sys.21414

Brucker, P. et al. (1999). Resource-Constrained Project Scheduling: Notation, Classification, Models and Methods. *European Journal of Operational Research, 112,* 3-41. https://doi.org/10.1016/S0377-2217(98)00204-5

Canada Energy Regulator. (2019). Canadian Energy Regulatory Act. (S.C. 2019, c. 28, s. 10). Retrieved from Government of Canada website: https://laws-lois.justice.gc.ca/PDF/C-15.1.pdf

Chen, H. L., Chen, W. T., & Lin, Y. L. (2016). Earned value project management: Improving the predictive power of planned value. *International Journal of Project Management, 34*(1), 22-29. https://doi.org/10.1016/j.ijproman.2015.09.008

Chen, R., Bloomfield, P., & Fu, J. (2003). An Evaluation of Alternative Forecasting Methods to Recreation Visitation. *Journal of Leisure Research, 35*(4), 441-454. https://doi.org/10.1080/00222216.2003.11950005

Chen, W. & Zhang, J. (2013). Ant colony optimization for software project scheduling and staffing with an event-based scheduler. *IEEE Transactions on Software Engineering, 39*(1), 1–17. https://doi.org/10.1109/TSE.2012.17

Chollet, F. et al. (2015). Keras. *GitHub*. https://github.com/fchollet/keras

Chow T., & Cao, T.B. (2008). A survey study of critical success factors in agile software projects. *Journal of Systems and Software, 81*(6), 961–971. https://doi.org/10.1016/j.jss.2007.08.020

Chu, C., Hsu, A.L., Chou, K.H., Bandettini, P., & Lin, C. (2012). Does feature selection improve classification accuracy? Impact of sample size and feature selection on classification using anatomical magnetic resonance images. *Neuroimage, 60*, 59–70. https://doi.org/:10.1016/j.neuroimage.2011.11.066

Costantino, F., Di Gravio, G., & Nonino, F. (2015). Project selection in project portfolio management: An artificial neural network model based on critical success factors,'' *International Journal of Project Management, 33*(8), 1744–1754. https://doi.org/10.1016/j.ijproman.2015.07.003

Cui, Z. & Gong, G. (2018). The Effect of Machine Learning Regression Algorithms and Sample Size on Individualized Behavioral Prediction with Functional Connectivity Features. NeuroImage, 178, 622-637. https://doi.org/10.1016/j.neuroimage.2018.06.001

Dalcher, D. (2014). Rethinking success in software projects: Looking beyond the failure factors. *Software Project Management in a Changing World*, 27-49. https://doi.org/ 10.1007/978-3-642-55035-5_2

Dikert, K., Paasivaara, M., & Lassenius, C. (2016). Challenges and success factors for large-scale agile transformations: a systematic literature review. *Journal of Systems and Software, 119*, 87–108. https://doi.org/10.1016/j.jss.2016.06.013

Elragal, A. & Al-Serafi, A.M. (2011). The effect of ERP system implementation on business performance: An exploratory case-study. *Communications of the IBIMA*, 1-19. https://doi.org/10.5171/2011.670212

Elragal, A. & Haddara, M. (2013). The Impact of ERP Partnership Formation Regulations on the Failure of ERP Implementations. *Procedia Technology*, *9*, 527-535. https://doi.org/10.1016/j.protcy.2013.12.059

Fayaz, A., Kamal, Y., Amin, S., & Khan, S. (2017). Critical Success Factors in Information Technology Projects. *Management Science Letters*, *7*(2), 73–80. https://doi.org/10.5267/j.msl.2016.11.012

Flyvbjerg, B. (2014). What you should know about megaprojects and why: An overview. *Project Management Journal*, *45*(2), 6-19. https://doi.org/10.1002/pmj.21409

Fryer, D., Strumke, I., and Nguyen, H. (2021). Shapley values for feature selection: The good, the bad, and the axioms. arXiv preprint arXiv: 2012.10936v1

GAO-21-306. (2021). *Report to congressional committees, NASA Assessments of Major Projects, GAO-21-306.* Washington, D.C.: Government Accountability Office.

Garces, I., Cazares, M. F., & Andrade, R.O. (2019). Detection of Phishing Attacks with Machine Learning Techniques in Cognitive Security Architecture. *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, 366-370, https://doi.org/10.1109/CSCI49370.2019.00071

Gemino, A., Sauer, C., & Reich, B. H. (2010). Using classification trees to predict performance in information technology projects. *Journal of Decision Systems*, *19*(2), 201-223. https://doi.org/10.3166/jds.19.201-223

Guillaume-Joseph, G. & Wasek, J. (2015). Improving software project outcomes through predictive analytics: Part 2. *IEEE Engineering Management Review*, *43*(3), 39-49. https://doi.org/10.1109/EMR.2015.2469471

Hammad, M.W., Abbasi, A., & Ryan, M.J. (2015). A new method of cost contingency management. *2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, 38-42. https://doi.org/10.1109/IEEM.2015.7385604

Highsmith, J. (2013). *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems.* Addison-Wesley.

Hoerl, A. E., & Kennard, R. W. (2000). Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics, 42*(1), 80–86. https://doi.org/10.2307/1271436

Huang, J., Li, Y., & Xie, M. (2015). An empirical analysis of data preprocessing for machine learning-based software cost estimation. *Information and Software Technology, 67*, 108-127. https://doi.org/10.1016/j.infsof.2015.07.004

Ika, L. (2009). Project Success as a Topic in Project Management Journals. *Project Management Journal, 40*(4), 6-19. https://doi.org/10.1002/pmj.20137

Imreh, R. & Raisinghani, M. (2011). Impact of agile software development on quality within information technology organizations. *Journal of Emerging Trends in Computing and Information Science, 10*(10), 460-475.

Jafarzadeh, R., Wilkinson, S., González, V., Ingham, J., & Amiri, G. (2014). Predicting Seismic Retrofit Construction Cost for Buildings with Framed Structures Using Multilinear Regression Analysis. *Journal of Construction Engineering and Management, 140*(3), 1943-7862. https://doi.org/10.1061/(ASCE)CO.1943-7862.0000750

Jun, L., Qiuzhen, W., & Qingguo, M. (2011). The Effects of Project Uncertainty and Risk Management on is Development Project Performance: a Vendor Perspective. *International Journal of Project Management, 29*(7), 923-933. https://doi.org/10.1016/j.ijproman.2010.11.002

Kerzner, H. (2018). *Project Management Best Practices: Achieving Global Excellence*. Wiley.

Khan, M.A., Mahmood, S. (2015) Measuring Flexibility in Software Project Schedules. *Arabian Journal for Science and Engineering*. *40*, 1343–1358. https://doi.org/10.1007/s13369-015-1597-x

Khanna, S. & Das, W. (2020). A Novel Application for the Efficient and Accessible Diagnosis of ADHD Using Machine Learning. *2020 IEEE / ITU International Conference on Artificial Intelligence for Good (AI4G)*, 51-54, http://doi.org/10.1109/AI4G50087.2020.9311012

Kuhn, M., & Johnson, K. (2018). *Applied predictive modeling*. Springer.

Kumari, S., & Pushkar, S. (2018). Cuckoo search-based hybrid models for improving the accuracy of software effort estimation. *Microsystem Technologies: Sensors, Actuators, Systems Integration, 24*(12), 4767-4774. https://doi.org/10.1007/s00542-018-3871-9

Leon, H., Osman, H., Georgy, M., & Elsaid, M. (2018). System Dynamics Approach for Forecasting Performance of Construction Projects. *Journal of Management in Engineering*, *34*(1), 1943-5479. https://doi.org/10.1061/(ASCE)ME.1943-5479.0000575

Lewis, C.D. (1982). *Industrial and business forecasting methods: A practical guide to exponential smoothing and curve fitting*. Butterworth Scientific.

Love, P.E., Edwards, D.J. and Irani, Z. (2012), Moving beyond optimism bias and strategic misrepresentation: an explanation for social infrastructure project cost overruns. *Engineering Management, IEEE Transactions*, *59*(4), 560-571. https://doi.org/10.1109/TEM.2011.2163628

Lundberg, S. M. et al. (2020). From Local Explanations to Global Understanding with Explainable AI for Trees. *Nature Machine Intelligence. 2*(1), 2522-5839. https://doi.org/10.1038/s42256-019-0138-9

Lundberg, S. M., & Lee, S.I. (2017). A Unified Approach to Interpreting Model Predictions. *Advances in neural information processing systems.* 4765-4774.

Malcolm, D. G., Roseboom, J. H., Clark, C. E., & Fazar, W. (1959). Application of a technique for research and development program evaluation. *Operations Research, 11*(5), 646–669. https://doi.org/10.1287/opre.7.5.646

Mansor, Z., Yahya, S. & Arshad, N.H. (2011). Towards the development success determinants charter for agile development methodology. *International Journal of Information Technology and Engineering*, *2*(1), 1-7.

Maroufkhani, P. et al. (2019). Big Data Analytics and Firm Performance: A Systematic Review. *Information (Switzerland). 10*(7), 1-21. https://doi.org/10.3390/info10070226

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

Messalas, A., Kanellopoulos, Y. & Makris, C. (2019). Model-Agnostic Interpretability with Shapley Values. *2019 10th International Conference on Information, Intelligence, Systems and Applications*, 1-7, https://doi.org/10.1109/IISA.2019.8900669

Misra, S. C., Kumar, V., & Kumar, U. (2009). Identifying some important success factors in adopting agile software development practices. *Journal of Systems and Software, 82*(11), 1869–1890. https://doi.org/10.1016/j.jss.2009.05.052

Mitchell, J., Mitchell, S., & Mitchell, C. (2020). Machine Learning for Determining Accurate Outcomes in Criminal Trials. *Law, Probability and Risk, 19*:1, 43-65. https://doi.org/10.1093/lpr/mgaa003

Mitrovic, Z. M. et al. (2020). Systems thinking in software projects-an artificial neural network approach. *IEEE Access, 8*, 213619-213635. https://doi.org/10.1109/ACCESS.2020.3040169

Mittas, N & Angelis, L. (2013). Ranking and Clustering Software Cost Estimation Models through a Multiple Comparisons Algorithm. *IEEE Transactions on Software Engineering*, *39*(4), 537-551. https://doi.org/10.1109/TSE.2012.45

Myrtveit, I., Stensrud, E., & Shepperd, M. (2005). Reliability and Validity in Comparative Studies of Software Prediction Models. *IEEE Transactions on Software Engineering*, *31*(5), 380-391. https://doi.org/10.1109/TSE.2005.58

Nasir, M.H., & Sahibuddin, S. (2011). Critical Success Factors for Software Projects: A Comparative Study. *Scientific Research and Essays, 6*(10), 2174-2186. https://doi.org/10.5897/SRE10.1171

Naur, P. & Randell, B. (1969). Software Engineering, presented at NATO Software Engineering Conference, Germany, 1968. Brussels, Belgium: Scientific Affairs Division.

Nelson, L.S. (1998). The Anderson-Darling Test for Normality. *Journal of quality technology, 30*(3), 298–299. https://doi.org/10.1080/00224065.1998.11979858

Pedregosa, F. et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*(2011), 2825–2830.

Pospieszny, P., Czarnacka-Chrobot, B., & Kobylinski, A. (2018). An Effective Approach for Software Project Effort and Duration Estimation with Machine Learning Algorithms. *Journal of Systems and Software, 137*, 184-196. https://doi.org/10.1016/j.jss.2017.11.066

Prabhakar, G.P. (2008). What Is Project Success: A Literature Review. *International Journal of Business and Management*, *3*(9), 3-10. https://doi.org/10.5539/ijbm.v3n9p3

Predescu, E., Stefan A., & Zaharia, A. (2019). Software effort estimation using multilayer perceptron and long short-term memory. *Informatica Economica*. *23*(2), 76-87. https://doi.org/10.12948/issn14531305/23.2.2019.07

Project Management Institute. (2017). *A guide to the Project Management Body of Knowledge – PMBOK Guide, sixth ed.* Project Management Institute PMI Book.

Pushphavathi, T.P. (2017). An approach for software defect prediction by combined soft computing. *2017 International Conference on Energy, Communication, Data Analytics and Software Computing*, 3003-3006. https://doi.org/10.1109/ICECDS.2017.8390007

Raschka, S. & Mirjalili, V. (2019). *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2, 3rd Edition.* Packt Publishing.

Rodriguez-Perez, R. & Bajorath, J. (2020) Interpretation of machine learning models using shapley values: application to compound potency and multi-target activity predictions. *Journal of Computer Aided Molecular Design, 34,* 1013–1026. https://doi.org/10.1007/s10822-020-00314-0

Roth, A. (1988). Introduction to the shapley value. *The Shapley value,* 1-27.

Ryan, J., Sarkani, S., & Mazzuchi, T. (2014). Leveraging Variability Modeling Techniques for Architecture Trade Studies and Analysis. *Systems Engineering: The Journal of INCSOE, 17*(1), 10-25. https://doi.org/10.1002/sys.21247

Saarela, M. & Jauhiainen, S. (2021). Comparison of feature importance measures as explanations for classification models. *SN Applied Sciences, 3,* 272. https://doi.org/10.1007/s42452-021-04148-9

Sabilla, S., Sarno, R., & Triyana, K. (2019). Optimizing Threshold using Pearson Correlation for Selecting Features of Electronic Nose Signals. *International Journal of Intelligent Engineering and Systems, 12*, 81-90. https://doi.org/10.22266/ijies2019.1231.08

Shawky, D. (2014). Traditional vs Agile Development - A Comparison Using Chaos Theory. *In Proceedings of the 9th International Conference on Software Paradigm Trends,* 109-114. https://doi.org/10.5220/0005096501090114

Shtub, A., Bard, J.F., & Globerson, S. (2005) *Project Management: Processes, Methodologies, and Economics, Second Edition.* Prentice Hall.

Sommerville, I. & Kotonya, G. (1998). *Requirements Engineering:Processes and Techniques.* John Wiley & Sons, Inc.

Sudhakar, G. (2012). A model of critical success factors for software projects. *Journal of Enterprise Information Management, 25*(6), 537-558. https://doi.org/10.1108/17410391211272829

Suliman, S. & Kadoda, G. (2017). Factors that influence software project cost and schedule estimation. *Sudan Conference on Computer Science and Information Technology*, 1-9, https://doi.org/10.1109/SCCSIT.2017.8293053

Svejvig, P. & Andersen, P. (2015). Rethinking project management: A structured literature review with a critical look at the brave new world. *International Journal of Project Management*, *33*(2), 278–290. https://doi:10.1016/j.ijproman.2014.06.004

Tam, C., Moura, E., Oliveira, T., & Varajao, J. (2020). The Factors Influencing the Success of On-going Agile Software Development Projects. *International Journal of Project Management, 38*(3), 165-176. https://doi.org/10.1016/j.ijproman.2020.02.001

The Standish Group. (2020). *CHAOS 2020: Beyond Infinity*. The Standish Group.

Tibshirani, R. (1996). Regression Shrinkage and Selection via the lasso. *Journal of the Royal Statistical Society Series B (Methodological)*, 58(1), 267–288. http://www.jstor.org/stable/2346178

Tishya, M., Aleena, S., & Moloud, A. (2019). Decision Tree Predictive Learner-Based Approach for False Alarm Detection in ICU. *Journal of Medical Systems, 43*(7), 1-13. https://doi.org/10.1007/s10916-019-1337-y

Tiwana, A., & Keil, M. (2004). The One-Minute Risk Assessment Tool. *Communications of the ACM, 47*(11), 73-77. http://doi.org/10.1145/1029496.1029497

Wang, K. et al. (2021). Software defect prediction model based on LASSO–SVM. *Neural Computing & Applications,* 33(14), 8249–8259. https://doi.org/10.1007/s00521-020-04960-1

Waskom, M. L. (2021). Seaborn: Statistical Data Visualization. *Journal of Open Source Software, 6*(60), 3021. https://doi.org/10.21105/joss.03021

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). Data Mining: Practical Machine Learning Tools and Techniques (4th ed.). Morgan Kaufmann.

Wojtas, M.A. & Chen K. (2020). Feature Importance Ranking for Deep Learning. arXiv preprint arXiv:2010.08973

Xu, D. et al. (2019). A Survey on Multi-output Learning. *arXiv preprint a*rXiv:1901.00248v2

Yusop, M. (2015). Understanding Usability Defect Reporting in Software Defect Repositories. *Proceedings of the ASWEC 2015 24th Australasian Software Engineering Conference*, 134–137. https://doi.org/10.1145/2811681.2817757

Zhang, M. & Zhou, Z. (2014). A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8), 1819–1837. https://doi.org/10.1109/TKDE.2013.39