12-16-2020

EDUCATION RESEARCH USING DATA MINING AND MACHINE LEARNING WITH COMPUTER SCIENCE UNDERGRADUATES

William Gregory Johnson
Georgia State University

EDUCATION RESEARCH USING DATA MINING AND MACHINE LEARNING WITH COMPUTER SCIENCE UNDERGRADUATES

by

WILLIAM GREGORY JOHNSON

Under the Direction of Anu G. Bourgeois, Ph.D.

ABSTRACT

In recent decades, we are witness to an explosion of technology use and integration of everyday life. The engine of technology application in every aspect of life is Computer Science (CS). Appropriate CS education to fulfill the demand from the workforce for graduates is a broad and challenging problem facing many universities. Research into this 'supply–chain' problem is a central focus of CS education research.

As of late, Educational Data Mining (EDM) emerges as an area connecting CS education research with the goal to help students stay in their program, improve performance in their program, and graduate with a degree. We contribute to this work with several research studies and future work focusing on CS undergraduate students relating to their program success and course performance analyzed through the lens of data mining.

We perform research into student success predictors beyond diversity and gender. We examine student behaviors in course load and completion. We study workforce readiness with creation of a new teaching strategy, its deployment in the classroom, and the analysis shows us relevant Software Engineering (SE) topics for computing jobs. We look at cognitive learning in the beginning CS course its relations to course performance. We use decision trees in machine learning algorithms to predict student success or failure of CS core courses using performance and semester span of core curriculum. These research areas refine pathways for CS course sequencing to improve retention, reduce time-to–graduation, and increase success in the work field.

INDEX WORDS: Education research, Educational data mining, CS student performance, Software engineering education, Classification, Decision trees

EDUCATION RESEARCH USING DATA MINING AND MACHINE LEARNING WITH COMPUTER SCIENCE UNDERGRADUATE

by

WILLIAM GREGORY JOHNSON

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2020

EDUCATION RESEARCH USING DATA MINING AND MACHINE LEARNING WITH COMPUTER SCIENCE UNDERGRADUATE

by

WILLIAM GREGORY JOHNSON

DEDICATION

This dissertation is dedicated to the memory of innocent souls lost and acknowledgement of irreparable damages from the COVID-19 pandemic. I honor their interrupted lives, unrealized potentials, and undiscovered purposes taken from our world. I also honor the selfless love and unrelenting dedication from front-line workers that risk physical and mental health each day to care for those in our society struggling with this virus and its lingering effects. I dedicate my work for their testament of perseverance in the face of unabating challenge to cope and live in our world crippled and forever changed in many ways by this virus.

I honor all my family, friends, and colleagues supporting me in this long and trying journey. May we all emerge from this dark time in a better place and with wisdom to reflect and learn from our work and from the work and love of others.

ACKNOWLEDGMENTS

I would like to acknowledge Jamie Hayes for his help in sourcing the dataset of computer science students from our university system I used in the research of my dissertation. His work has given me avenues of new knowledge and understanding. He is the consummate professional in the CS department at Georgia State University (GSU).

I would like to thank my advisors for their guidance and sponsorship. It was a significant help in this complex process of organization, discovery, and coping with pressures of undertaking a terminal degree in computer science. I would like to thank Anu for her patient voice and ability to teach me to see that research is larger and more focused than the sum of their smaller parts. I would like to thank Raj for his subtle and cheerful wisdom that has guided me since 2001 in the CS department and opened many doors that have supported my transition into academe. I would like to thank Yubao for his contagious enthusiasm of data mining and his instruction and guidance that has expanded my thinking with innovations of using data and machine learning. I would like to thank Melinda for agreeing to be part of my committee at the request of my husband, but more importantly I thank her for sharing her passionate interest in data science that has helped me realize the value and necessity of statistics in research.

I would like to thank the review committee from the 13th annual International Conference of Education, Research and Innovation (iCERi 2020) for their review, feedback, and acceptance of my paper featured in Chapter 5. Also, I would like to thank the review committee from the 9th Computer Science Education Conference (CESRC 2020) for their review, feedback, and acceptance of my paper featured in Chapter 6.

Lastly, I would like to thank my high school guidance counselors. If were not for their interest in my future education and determination to convince my parents that I should get a 4-year degree, I would not have attended college. Nor would I have transitioned from a small farm in rural Alabama to a large urban city in Georgia, and never would I have thought of pursuing a terminal degree in computer science with the dream of teaching computer science in a university. I thank them and my parents for confidence in my abilities and promise of a brighter future in industry and academe.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

- ACM - Association of Computing Machinery
- AI - Artificial Intelligence
- CART - Classification and Regression Trees
- CP - Complexity Parameter
- CS - Computer Science
- CSCC - Computer Science Core Curriculum
- DM - Discrete Math
- DS - Discrete Structures
- EDM - Educational Data Mining
- FERPA - Family Educational Rights and Privacy Act
- GAN - Generative Adversarial Network
- GPA - Grade Point Average
- GSU - Georgia State University
- ID3 - Iterative Dichotomiser 3
- IRB - Institutional Review Board
- KA - Knowledge Area
- KDD - Knowledge Discovery and Data Mining
- LMS - Learning Management System

- MACROVR - MAchine learning to select project team members; Cloud technologies required for project control, code versioning, and team communications; ROtational schedules in Agile roles; an individual Video of the team project story board; and Rubrics for all presentations
- ML - Machine Learning
- MOOC - Massive Open On-Line Courses
- PBL - Project–Based Learning
- STEM - Science, Technology, Engineering, and Math
- TTA - Time–To–Algorithms
- TTG - Time–To–Graduation

1 INTRODUCTION

Across all industries in most societies, computing usage is prevalent without question. With this pervasive integration, a resulting large demand from our workforce for highly skilled and knowledgeable Computer Science (CS) graduates creates an evident shortage. While CS student enrollments have grown, universities are facing a difficult time to keep up with increasing their faculty and other resources to respond in kind. With a large student population, there exists a diverse set of learning aptitudes, leading to traditional methods of CS content delivery, however, failing to meet the academic needs of this population [1–3].

Many have tried to address the need by using various techniques found in research such as blended classrooms, hybrid learning, flipped classrooms, group learning, and Massive Open On–Line Courses (MOOC) [4–6]. This has not been enough. Instead of focusing on pedagogy and content delivery, we can instead look at the CS student factors like race, sex, transfer status, prerequisite grades, and CS course sequencing to identify ways of improving performance and educational experiences and outcomes.

Our focus is to observe our diverse student body, and through a variety of modalities, identify predictors of student performance - good and bad. Recent CS educational research is proving to offer more accurate findings to help design intervention systems and more accurate advising [7–9]. Our discovery can lead to developing customized interventions and learning techniques leading towards pathways of student success.

Georgia State University (GSU) has recently been recognized as one of the most innovative universities in the nation with respect to transforming education experiences through analysis of student populations [10]. Recently in a 2021 U.S. News and World Report[1], they rank GSU third in the United States as most innovative and in the top ten over the last five years. Additionally, GSU ranks tenth in the United States as having one of the most diverse student populations [11]. The main reason for their achievements is the use of data analytics to improve student retention, progression, and graduation rates for the general population across all majors at GSU. This approach shows value from results seen in the very populations that undergraduate education has traditionally failed - first generation, underrepresented groups, and low income. We take this approach of using data analytics focusing on the CS majors, and more specifically, the part of the program that is CS specific and not the general core classes. In this manner, we can identify uniqueness for the CS discipline, rather than being generalized across the entire university with all academic programs. Our approach can be tailored for analysis in any other specific academic program or major to help identify performance predictors for their students.

Our first analysis is examining the performance impact of the CS course load in a given semester between two cohorts of CS students - transfer and native (non-transfer). Our university supports a large number of transfer students from several two-year colleges that offer an Associate Degree in CS. Recently in 2016, our university merged with a two-year college that offers this degree and we anticipate an even larger number of transfer CS students. It has been observed these students have fewer completed lower level CS classes, but most all of the general core classes related to a Baccalaureate Degree.

Our second analysis investigates student predictors of performance in the CS algorithms course. In particular, we focus on various factors to identify which ones are predictors of their performance in the algorithms course, namely, race, sex, prerequisite GPA, transfer status, and semester span between passing a prerequisite course and passing the algorithms course. Using regression analysis with numerous models, we determine the most influential predictors for a successful GPA in the algorithms course.

[1]https://www.usnews.com/best-colleges/rankings/national-universities/innovative (Accessed in November 2020).

Our third analysis centers on a survey study involving our newly developed and novel teaching approach in a Software Engineering (SE) course. We call our strategy and approach to teaching SE, MACROVR: MAchine learning to select project team members; Cloud technologies required for project control, code versioning, and team communications; ROtational schedules in Agile roles; an individual Video of the team project story board; and Rubrics for all presentations (MACROVR). As this class is often taken towards the end of a four-year CS degree program, the goal of our study is to determine if our novel approach better prepares senior and graduating CS students for computing job interviews. We deliver a five–point Likert scale survey to the students and assess the impact of our new teaching strategies, pedagogical innovations, and project–based team learning.

In our fourth analysis, we examine the performance of a CS1 course relating to cognitive thinking load. We ask students several interpretive questions relating to the CS major's qualifying chart. Using inferential analysis, we investigate if any correlation between survey scores, race, gender, high school CS course offering, SAT math score, public/private high school, and high school population numbers (independent variables) impact the student's final course grade (dependent variable) could be found. Our results support the null hypothesis: there are no correlative factors from these independent variables and the course final grade. This research shows a non–positive outcome. We discuss this work in our conclusions.

Our last research and analysis involves the interpretation of the CS core curriculum related to classification and predictive models. In this work, we use machine learning decision tree algorithms to examine prerequisite course performance and semester span (sequencing) to build optimal paths for success or failure in a variety of CS core curriculum courses. This data gives us new insight to add to domain knowledge for intelligent advisement, CS course scheduling, and proactive alerting systems for increasing student retention and completion of the CS program's core curriculum.

2 BACKGROUND

CS education research is a widely studied topic mostly focusing on classroom experience reports, acumen of subject content, pedagogical improvements, and innovations to increase student participation. There exists measurements of outcomes in these research topics and as of late, data mining and machine learning enter the collective work as tools to make predictions and observe new models of CS student success and identify points of struggle or failure. This new area of knowledge extraction, Educational Data Mining (EDM), is an analysis process with interdisciplinary appeal where data mining and machine learning are used to expand knowledge for stakeholders in education settings [12].

Most EDM research centers around classification and prediction with large groups of students across many academic programs as found in [13–17]. Another area of EDM application is the K–12 education environments and mining student dynamic data from online education experiences [18]. Some of the more recent EDM works include deep learning and using tensorflow for prediction models with focus on grouped STEM academic programs [19]. Using these more advanced types of EDM techniques require large amounts of data, substantial computing resources, and a high level of secondary CS education to effectively apply the tools.

The literature does not show much with EDM applied specifically to the CS academic program. Our research is uniquely positioned to add work in this field particular to CS students at GSU. With our recent works, the focus is trying to determine factors that affect CS student performance with attributes such as CS or math course grade, semester span between a prerequisite CS course as well as the applied CS course, and demographic identifiers. We use several tools like Stata15, R, Excel, Python, and Java to gather descriptive and inferential statistics and then apply data mining techniques to relatively small data sets for predictive models. It is our goal to expand this work with generative modeling that involves automatically discovering and learning the regularities or patterns in our approximately 49k CS students dataset.

Using this technique of modeling, we can generate very large amounts of synthetic data that plausibly is drawn from our original dataset. We will discuss this in more detail in the conclusions.

3 RESEARCH DATA SETS

The data used in our analysis all originate from two categories: 1) the university's student data system, and 2) CS student surveys. The first category consists of undergraduate students with a declared major of CS. The second category consists of a sub-group in the first category's population, but sources from online and paper-based surveys. The online survey is created and delivered via email from the university's Qualtrics system and our paper-based survey is delivered in-person from a classroom setting.

Processing our datasets involves numerous steps to reach interpretations related to our goals. These steps require preprocessing actions for machine learning analysis in Chapter 7, and different transformations to reflect the descriptive and inferential analysis in Chapters 4, 5, and 6. Achieving an outcome of useful information from our data can be visualized as a continuum beginning with unorganized and sparse data that we clean and transform. Next, we apply data mining techniques for organization and analysis, and lastly we realize patterns that deliver knowledge. This concept, known as Knowledge Discovery and Data Mining (KDD) is shown in Figure 3.1.

In our Chapters 4, 5, and 7, the data originates from 239K records of student events, spanning from 2008 - 2018, representing each individual course performance by a CS student. Using a variety of transformation programs, the data was amalgamated to approximately 49K records of individual students that reflect their CS academic program activity across math and CS courses. The data includes performance values from prerequisite courses in mathematics and any completed CS course. Additional to demographics of gender and race, we collect the transfer status and the semester (year, month) corresponding to an acquired grade.

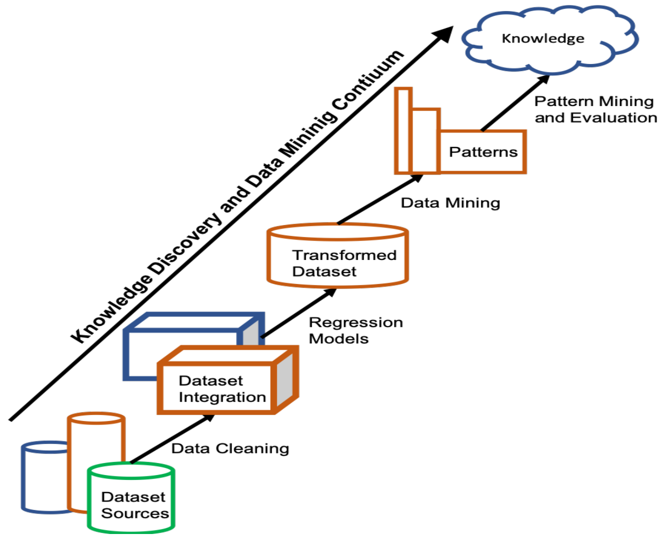


Figure 3.1: Knowledge Discovery and Data Mining Continuum.

The second category of data is used for Chapter 6, where we perform an inferential analysis related to teaching strategies in Software Engineering (SE) education. This data is represented with a five-point Likert scale and sourced from the GSU Qualtrics system. The anonymous participants were selected from students that completed the SE course and were seeking a computing industry job. The survey reached a total of n=316 and we received x=116 to be analyzed in our study. Additionally, the second category is used in another study with beginning CS students. A paper-based survey was given to freshman/beginning CS students taking our 'CS1' course. We collected n=238 survey responses with x=198 being completed. This study, while not producing a positive outcome supporting our intuition and hypothesis, is discussed in the conclusions and insight for future use is offered.

3.1 University's Student Data System Composition

GSU's student population, as reported in 2018, is highly diverse according to [11]. Over 55% of undergraduate students are non-white and receiving Pell Grants. Our dataset of CS student population is reflective of this diversity with approximately 68% of undergraduates in the non-white category as shown in Table 3.1. The time span for this dataset begins with the Spring semester of 2018, and includes all three academic year semesters going back ten years.

We find our CS student dataset's racial composition beneficial for comparison to GSU and most urban universities in the United States. With respect to diversity, GSU ranks twelfth in the nation as reported by US News and World Report [20]. As shown in Table 3.1, the race groups are Asian, Black, other, and White. We group all races of American Indian or Native Alaskan, Native Hawaiian or Pacific Islander, Multiracial, and Not-reported as 'other' in our categories. We find racial groups in our CS population are higher in Asian and White, but lower in Black and 'other' groups. This may be a result of a well known and researched problem of the lower number of underrepresented groups in a Science, Technology, Engineering, and Math (STEM) academic program.

From Table 3.2, the CS student dataset is lower for female and higher for male compared to the GSU composition. Again, this may be a result of a well known and researched problem of the lower number of underrepresented groups in a STEM academic program. Interestingly, we discover the transfer and native percentages between the CS dataset and GSU population recordings have a significant difference, comparatively. This may be a result of our geographic location related to a large number of colleges granting two-year CS degrees and the recent merger with Georgia Perimeter College in 2016[1].

Table 3.1: Racial Student Population Comparisons.

Source	Asian	Black	Other	White
CS Data	**25.4%** (12,425)	**31.3%** (15,310)	**13.3%** (6,505)	**30%** (14,675)
GSU[2]	**13.2%** (3,594)	**42%** (11,437)	**19.8%** (5,391)	**25%** (6,809)

[1]https://consolidation.gsu.edu/ (Accessed in October 2020).

[2]https://www.collegefactual.com/colleges/georgia-state-university/student-life/diversity (Accessed in October 2020).

Table 3.2: Other Student Population Comparisons.

Source	Female	Male	Transfer	Native
CS Data	**24.5%** (11,994)	**75.5%** (36,921)	**25%** (12,315)	**75%** (36,600)
GSU[3]	**58.9%** (16,027)	**41.1%** (11,204)	**7%** (1,839)	**93%** (25,392)

3.2 Survey Data Composition

The data used in our secondary category consists of two subgroups of our CS student population. One is from an online survey delivered from the university's Qualtrics system and the other is a paper-based survey delivered in-person from a classroom setting. The online survey was given anonymously to the SE students over a period of time starting in Fall 2017 until Spring 2019. The purpose of the online survey was not dependent on attributes of the student demographics, but focused on the teaching strategies, pedagogical content, and project-based team learning.

The paper-based survey given to the CS1 students was not anonymous therefore we extracted demographic and pre-college data from the university's data system and merged this with the survey results. This study's demographic composition is shown in Table 3.3. We see the CS1 student population being significantly different from the CS dataset and GSU's recordings in the Asian, White, female, male, transfer, and native categories. This may be the result of the CS1 course is taken from students other than declared CS majors, thus the differences between the survey and CS dataset. We discuss this data and its impact in the conclusions.

[3]https://www.collegetuitioncompare.com/edu/139940/georgia-state-university/enrollment (Accessed in October 2020).

Table 3.3: Dataset Student Population Comparisons.

Source	Asian	Black	Other	White	Female	Male	Transfer	Native
CS Data	**16%**	**39%**	**13%**	**32%**	**44%**	**56%**	**25%**	**75%**
CS1 Survey	**40.1%**	**26.9%**	**12%**	**21%**	**31%**	**69%**	**12%**	**88%**
GSU	**13.2%**	**42%**	**19.8%**	**25%**	**58.9%**	**41.1%**	**7%**	**93%**

4 PERFORMANCE TRENDS COMPARING TRANSFER AND NON-TRANSFER COMPUTER SCIENCE STUDENTS

The measures of undergraduate student success factors in CS are researched in a variety of methods. Most rely on pedagogical interventions and the measurement of impact from hybrid learning, flipped classrooms, and activities of face-to-face delivery. Using Machine Learning (ML) and data mining to analyze and predict student success has mostly relied on statistical and regression models based on grades and to some degree Learning Management System (LMS) activity, while other research incorporates intelligent tutoring systems (ITS) and recommender systems data. By analyzing eight years of CS course data from our university's student data system of Fall 2008 through Spring 2016, we use descriptive analysis to compare the transfer with non-transfer student. The demographic and course performance data is related to each Fall or Spring semester and we remove the Summer semesters because GSU does not offer the full array of undergraduate CS courses in the Summer semester. We show that transfer students tend to take a higher load of CS classes per semester and their performance is consistently lower than that of non-transfer (native) students. The data corpus includes demographics in gender diversity and underrepresented groups. Our initial findings of *pass/fail* rates related to course load and transfer status are reported in this chapter and we offer further research to incorporate more features with ML techniques and data mining algorithms.

4.1 Introduction

The analysis of student success factors in undergraduate CS education are researched in a variety of modalities. Using statistical regression, ML techniques, and data mining algorithms, many influencing features are found that result in using more advanced methods to determine impact of the feature(s). CS educational research is proving to offer more accurate findings for prediction and help with design of intervention systems and more accurate advising to increase CS student retention and increase graduation rates [7–9].

This chapter focuses on the impact to *pass/fail* rates by examining two independent features: 1) CS course load and 2) transfer versus non-transfer status. We indicate a *pass* as an **A+** to a **C**. We do not assign a grade of **C-** and the grades **D**, **F**, **W** and **WF** are considered as *fail* status for the CS course. This is explained in Section 4.3. Our experience in the department is that a majority of transfer students enroll in more than two CS courses in a Fall or Spring semester. Our data shows a negative impact in taking more CS courses in a semester and indicates a higher percentage of transfer students enroll in more than two per semester. The observations of CS course load and the impact on *pass/fail* rates of CS students is an area not readily found in the current literature. Searching for research of CS course load impact to transfer and non-transfer students we find references in course areas of accounting, natural resources and sciences, and wildlife related to US universities, but none in CS education particular to CS course load [21–24].

Our university supports a large number of transfer students from several two-year colleges that offer an Associate Degree in CS. Recently in 2016, our university merged with a two-year college that offers this degree and we anticipate even larger numbers of transfer CS students. It has been observed these students have fewer completed lower level CS classes, but most all of the general core classes related to a Baccalaureate Degree, as shown in Figure 4.1. Students with most all of their general core classes completed creates a concern for finishing their degree in four years by taking more than two CS courses a semester versus doing well in the CS courses and giving them a chance for a higher GPA. From our current CS course requirement sequence, once a CS student has completed data structures and systems programming, the pool of available CS courses is large and this is where concern arises for the effectiveness of CS learning by taking more than two CS courses in a semester. This fact was made evident by an increasing population of transfer students as shown in Figure 4.2.

Our conjecture is based upon observations of our undergraduate Director and her interactions with students. There was also an increase in requests for an overload waiver to take beyond the maximum eight credit hours in a semester.

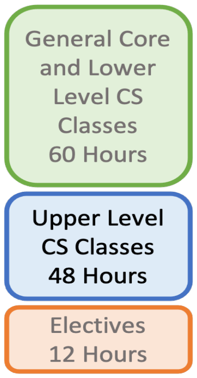

General Core and Lower Level CS Classes 60 Hours
Upper Level CS Classes 48 Hours
Electives 12 Hours

Figure 4.1: Regular and CS Course Areas for BS Degree.

4.2 Methods

The dataset used for this research was created from the university's database of CS student enrollment and end of semester reporting. We extracted data containing the CS course taken, the hours of the course, and the letter grade given to the student for each Fall and Spring semester over the past ten academic years. The entire strata of student classification is combined in our data (freshman, sophomore, junior, senior) and we excluded all graduate student data.

The data has been de-identified by using a Java based program that initially extracted the data from CSV files, then transformed in memory only, and finally loaded into the database from each semester's data file. The student ID was given a software generated index while being transformed in memory, thus ensuring no reverse identification data would be recorded somewhere or loaded into our database. As the extract, transform, loading (ETL) program ran, data attributes were stored to allow only a CS GPA calculation be acquired. However, the student's overall school grade point average (GPA) cannot be calculated as only the CS course grades are examined and CS GPA is reported. The course points were assigned based on Table 4.1 and the **WF** valued as a zero is used in the GPA computation.

In order to indicate a student record for a semester, it consists of the recorded semester and year (”08” and ”01” for Fall or Spring, respectively), the record index, all course hours with their assigned points, a DFWWF flag indicating the student has one or more *fail* (”1” is true or ”0” is false), and a transfer flag with the same meaning as DFWWF flag. The **WF** is included in the computation as value zero, but the **W** is not. In order to indicate what type of CS course failure occurred and whether or not to include that failure into the CS GPA, we store a **W** with point value of negative one and **WF** with point value of negative two. Our schema allows for up to six CS courses taken in one semester to be in the study. The CS student populations in the five and six CS course loads were very low, so these are considered as outliers as indicated in Table 4.3. A single tuple in our dataset for a **transfer** student with four courses (each as four hours), an **A**, **B-**, **W**, and **WF**, is shown in Table 4.2. The tuple in Table 4.2, indicates in the Fall of 2008, the student withdrew from course 3 (assigned a **W**), and withdrew failing from course 4 (assigned a **WF**), thus indicating a failure in the **FAIL** field. When transforming the data to analyze *fail* versus *pass* rates we use this DFWWF indicator to reduce computational complexity. The **TX** field is used to indicate if a student has matriculated as a transfer student from another college or university (”1” is true or ”0” is false).

Table 4.1: The Grading Scale.

Grade	**A+**	**A**	**A-**	**B+**	**B**	**B-**	**C+**	**C**	**C-**	**D**	**F**	**WF**
Value	4.3	4.0	3.7	3.3	3.0	2.7	2.3	2.0	N/A	1.0	0.0	0.0

Table 4.2: A Single Tuple.

SEM	**IDX**	**C1**	**P1**	**C2**	**P2**	**C3**	**P3**	**C4**	**P4**	**C5**	**P5**	**C6**	**P6**	**FAIL**	**TX**
08_2008	74	4	4.0	4	2.7	4	-1	4	-2	0	0	0	0	1	1

4.3 Results

These initial findings validate our intuition concerning the CS course load and transfer status related to *pass/fail* rates.

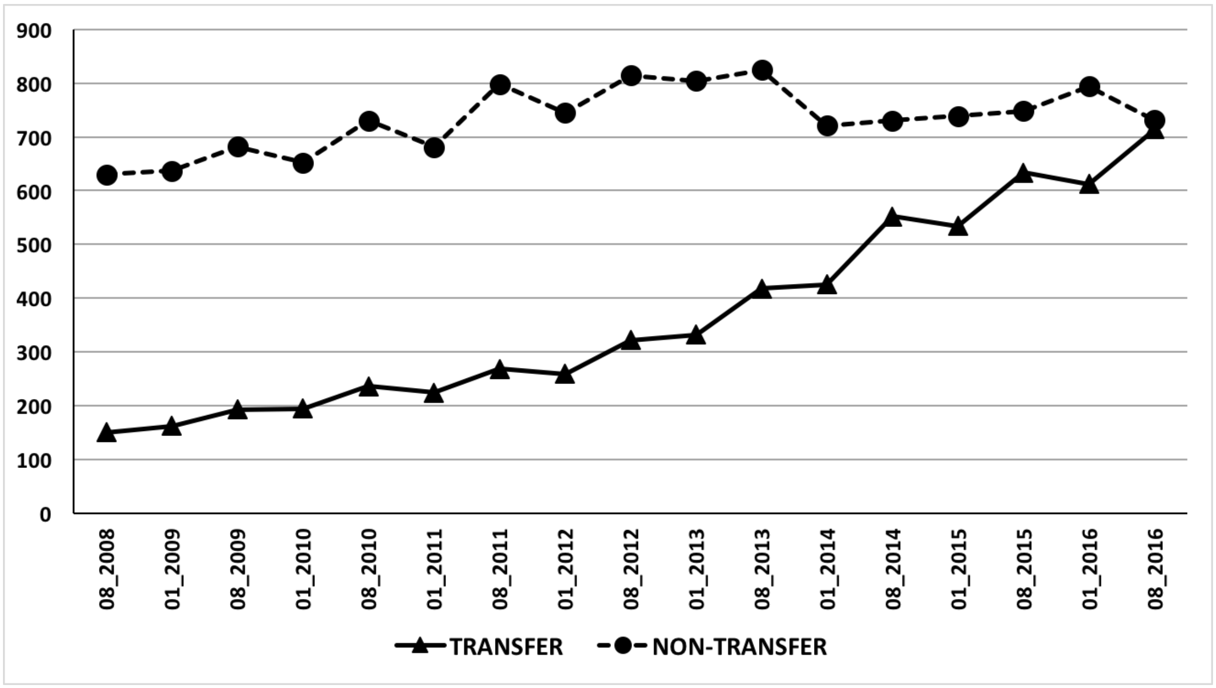


Figure 4.2: CS Student Population (Merger in 08/2016).

We analyze only the Fall and Spring corpus for knowledge discovery, because as stated earlier, the Summer semester is a sub-group of the undergraduate CS course offerings. The graph shown in Figure 4.3, indicates the percentage of each cohort relative to their total population in the data corpus. Table 4.3 shows the CS population composition across the CS course loads. Starting with two CS courses each semester, we find the transfer students are registering for more than non-transfer with 27% compared to 12%. This data does not indicate CS student classifications (freshman, sophomore, junior, senior), nor the level of the CS course indicated in Figure 4.1.

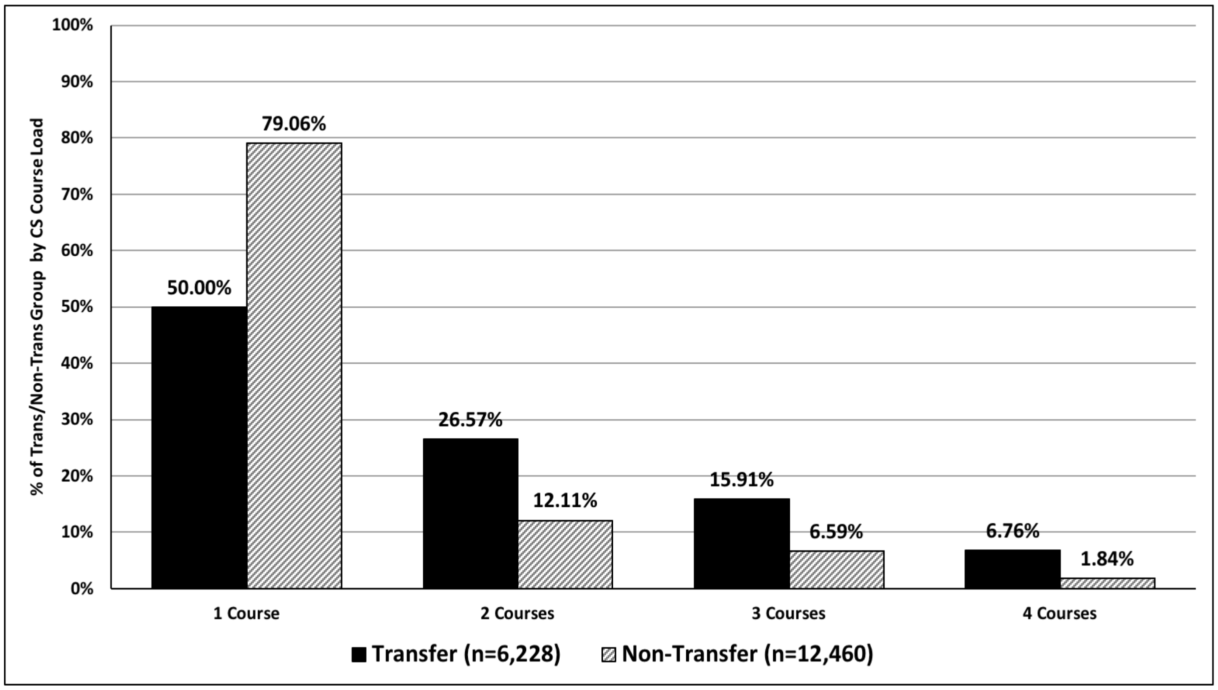


Figure 4.3: Percentages of Transfer/Non-Transfer Students in Various CS Course Loads.

Having established the CS course load variations, we moved to analyze the *pass/fail* rates of students based on transfer versus non-transfer status. From Figure 4.4, the data shows insignificant difference except when the transfer student registers with three CS courses. Figure 4.4 shows a failure for the CS student in <u>at least one CS class</u> indicating some setback.

Table 4.3: Student Populations by Semester CS Course Load.

CS Cohort	One	Two	Three	Four	Five	Six	Total Cohort
Transfer	3,114	1,655	991	421	45	2	6,228
Non-Transfer	9,851	1,509	821	229	36	2	12,460

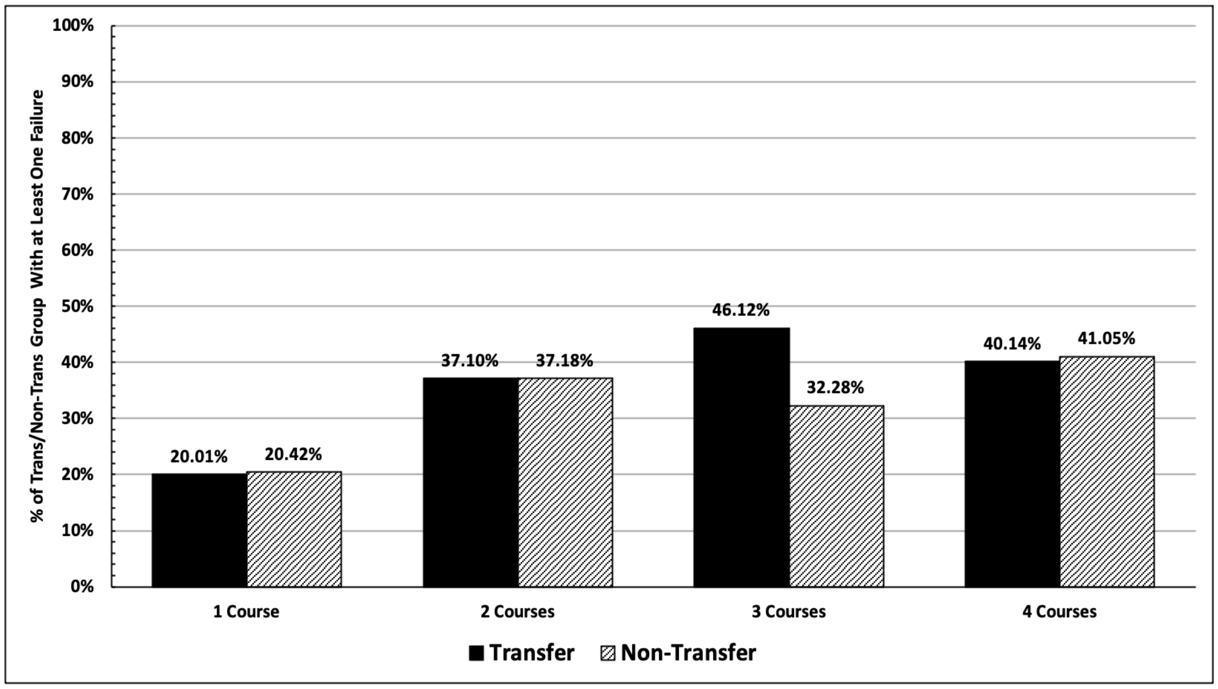


Figure 4.4: Transfer vs Non-Transfer Students With at Least One Failure Based on Their CS Course Load.

Another indicator of student success is the CS GPA calculations shown in Figure 4.5. We observed all course loads indicate a lower CS GPA for transfer students and at three and four courses the GPA for transfer students being 0.35 and 0.13 points lower, respectively. These two numbers show the largest delta in the CS GPA values. Regardless of the CS course loads, transfer students score lower and at three CS courses, the data indicates the largest CS GPA gap.

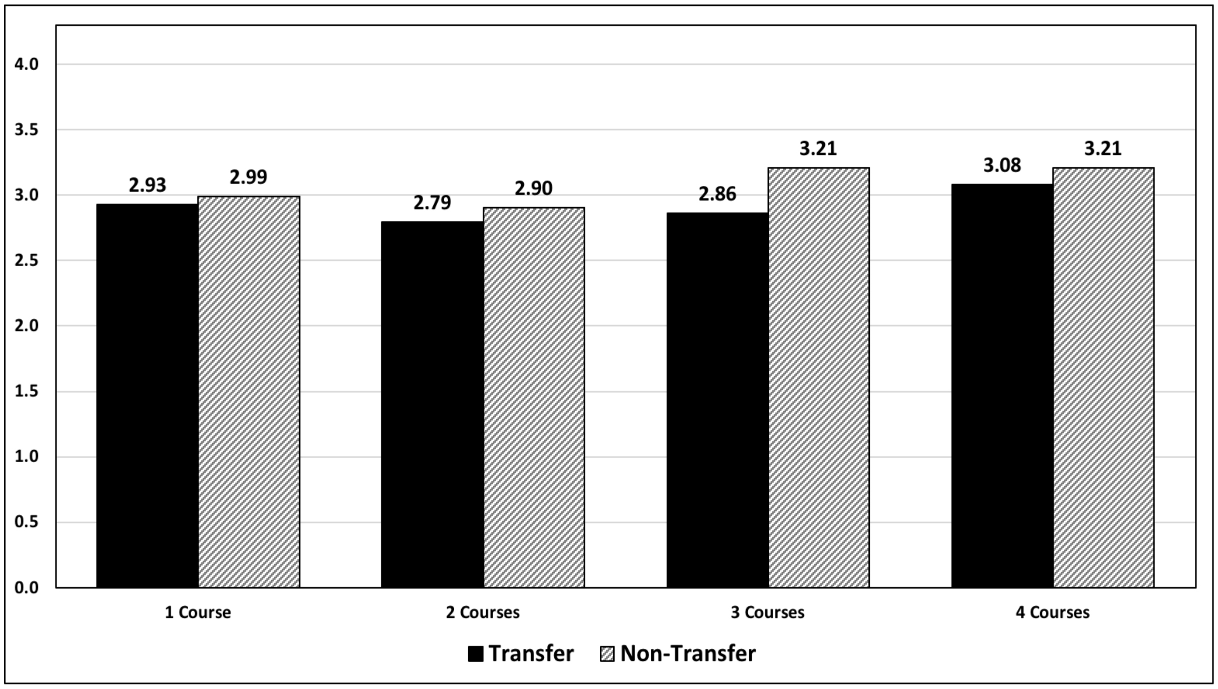


Figure 4.5: μ CS GPA in Various CS Course Loads.

We continue our analysis of *fail/pass* differences by analyzing the *fail* rates of the cohorts based upon their CS course load relative to how many **D**, **F**, **W** and **WF**, were obtained in each group over a given CS course load. When we analyzed complete CS course fail rates of the student, we see data shown in Figure 4.6. This indicates a cohort is demonstrating a consistent failure related to their status. For course loads of two and higher, we consistently see that transfer students have a higher rate of failing all their CS course load in a semester.

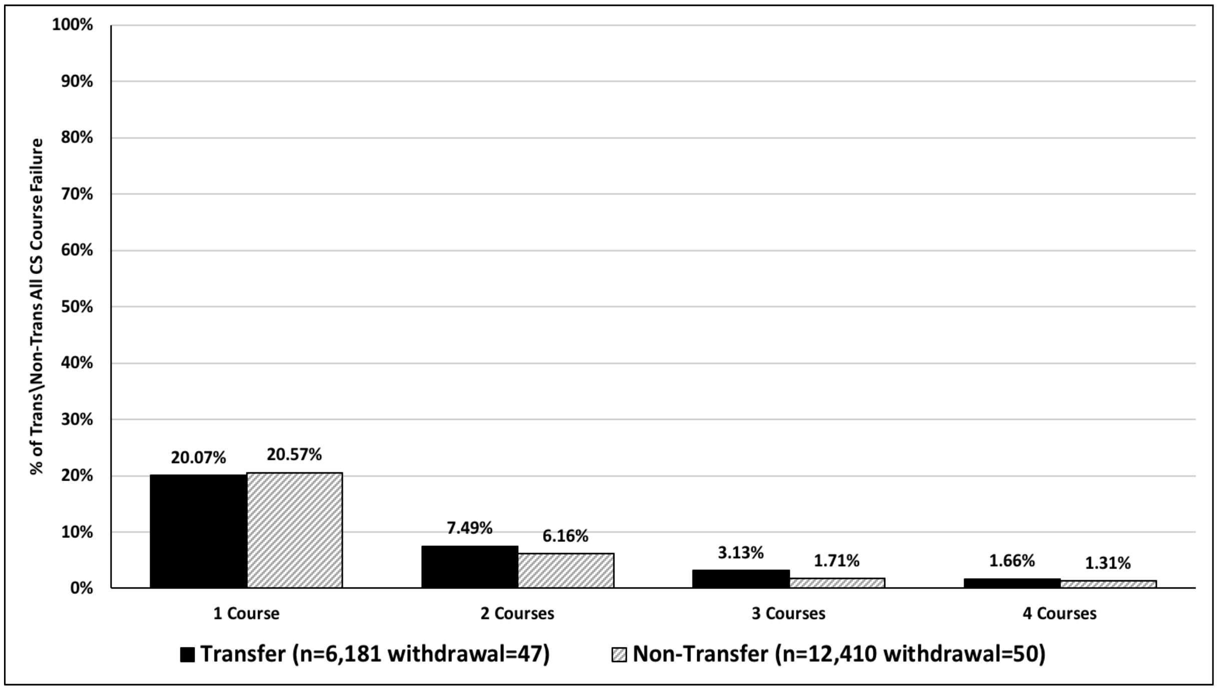


Figure 4.6: Various Course Loads Where Failed All CS Courses in a Semester.

We see the need to analyze our dataset deeper in the actual course level and the requirements as shown in Figure 4.7, to be included in the model. Doing so examines the difficulty of a CS course being taken by the cohorts to indicate why one performs poorly and one does not. Currently our data includes performance in classes ranging from the introductory level all the way to senior level classes. We expect to see even more striking results when focusing on a common subgroup of students and/or courses.

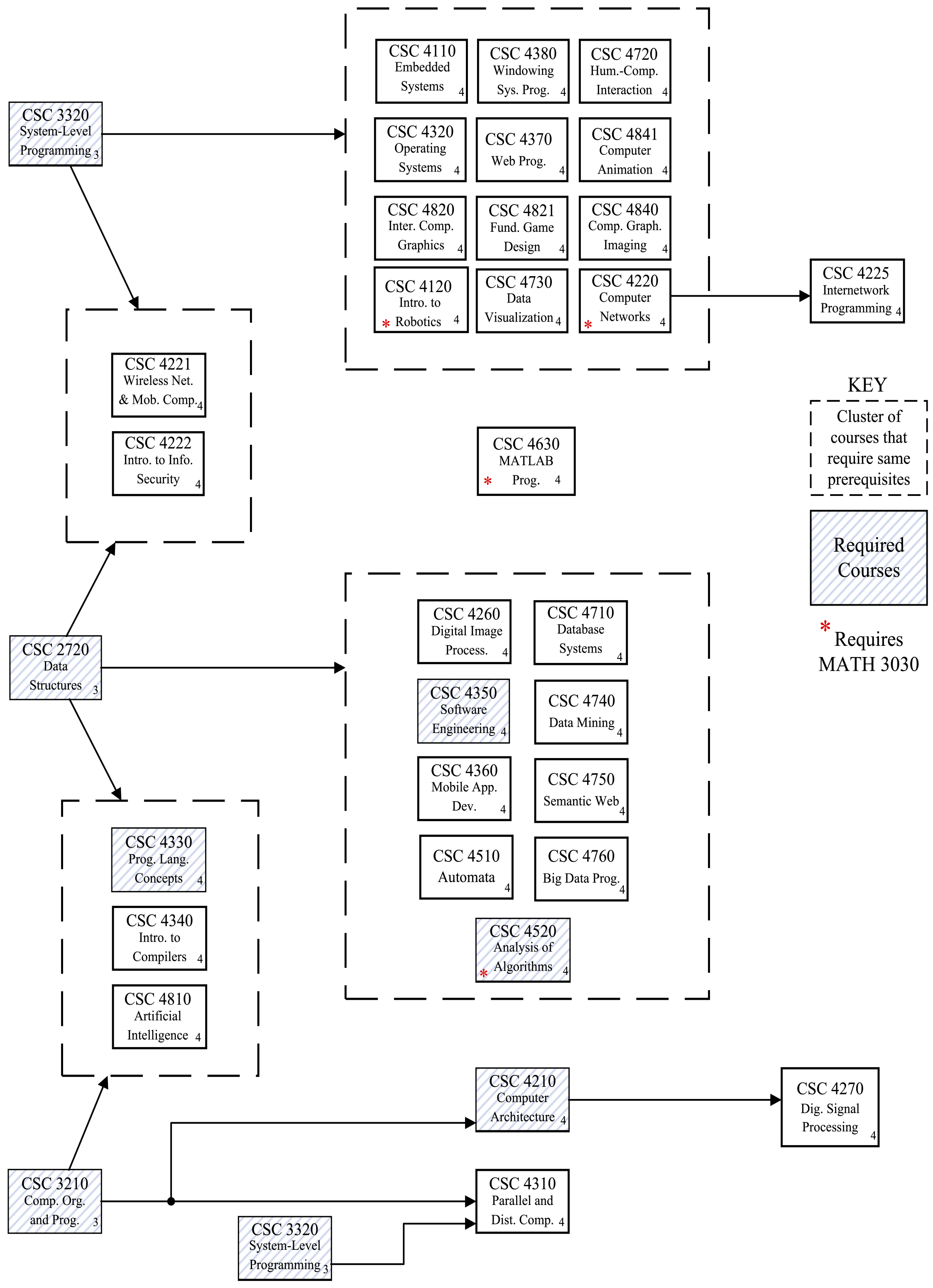

CSC 3320 System-Level Programming 3
CSC 4110 Embedded Systems 4
CSC 4380 Windowing Sys. Prog. 4
CSC 4720 Hum.-Comp. Interaction 4
CSC 4320 Operating Systems 4
CSC 4370 Web Prog. 4
CSC 4841 Computer Animation 4
CSC 4820 Inter. Comp. Graphics 4
CSC 4821 Fund. Game Design 4
CSC 4840 Comp. Graph. Imaging 4
CSC 4120 Intro. to * Robotics 4
CSC 4730 Data Visualization 4
CSC 4220 Computer * Networks 4
CSC 4225 Internetwork Programming 4
CSC 4221 Wireless Net. & Mob. Comp. 4
CSC 4222 Intro. to Info. Security 4
CSC 4630 MATLAB * Prog. 4
KEY
Cluster of courses that require same prerequisites
Required Courses
* Requires MATH 3030
CSC 2720 Data Structures 3
CSC 4260 Digital Image Process. 4
CSC 4710 Database Systems 4
CSC 4350 Software Engineering 4
CSC 4740 Data Mining 4
CSC 4360 Mobile App. Dev. 4
CSC 4750 Semantic Web 4
CSC 4510 Automata 4
CSC 4760 Big Data Prog. 4
CSC 4520 Analysis of * Algorithms 4
CSC 4330 Prog. Lang. Concepts 4
CSC 4340 Intro. to Compilers 4
CSC 4810 Artificial Intelligence 4
CSC 4210 Computer Architecture 4
CSC 4270 Dig. Signal Processing 4
CSC 3210 Comp. Org. and Prog. 3
CSC 4310 Parallel and Dist. Comp. 4
CSC 3320 System-Level Programming 3

Figure 4.7: Computer Science Course Requirement Sequence.

4.4 Conclusions

It is an undisputed fact that demand in society of K-12, CS learning, and four-year university CS graduates is increasing [25,26]. With dependence on computer technology and CS skills in higher demand from the workforce and the growing integration of autonomous computing in everyday living, the issues that influence success factors of CS graduates is ever more important. Much research has been done to analyze factors that influence this number and create solutions to indicate and mitigate them. Using ML, deep learning, and advanced data mining techniques are new areas in CS education data mining research. This chapter examines two naive CS student features to indicate a serious problem:

Transfer students taking more than two CS courses:

A. Have a lower overall μ CS GPA.

B. Have a higher complete fail rate for CS course load.

We demonstrate this clearly with analysis of a large dataset of CS student data and simple regression computations. We identify that transfer CS students do take more CS courses than their non-transfer cohort resulting in negative impact to CS GPA. The driving mechanisms of this trend is where we can expand our analysis and identify areas relating to these issues with investigation of mitigating interventions focused on transfer students, thus contributing to higher effectiveness and success factors for CS students and CS education.

The features used in this research were chosen based on protecting the student identity and observations of academic advising to the undergraduate CS student. The detrimental effects and the realization in this study is reason for further investigation. Working together with our four-year university, more features will be gathered to be used in our continued analysis. Within the CS department, we advise CS students against registering for more than two CS courses in a semester. However, our university advising system has a goal for students to graduate within four to six years and can be in conflict with the department at times. By demonstrating the impact of indicators and importance of increasing the number of CS graduates, our work can be used in advanced intervention systems and targeted advisement resources for undergraduate CS students to do well and finish within a four-year time period. Discovery of knowledge with student success related to a **D**, **F**, **W**, or **WF** in one of their CS courses is demonstrated with transfer and non-transfer status. This is inspiration to discover other knowledge from dominate features in a larger dataset.

5 PREDICTORS OF PERFORMANCE IN COMPUTER SCIENCE ALGORITHMS

In this study, we identify predictors of student performance (both good and poor) in an algorithms course. We focus on the algorithms course due to its integral nature within the CS discipline. Typically, features of race and/or gender are considered in such studies. However, we explore these combined with other features that can serve as predictors, specifically:

1. transfer status,
2. GPA of a prerequisite discrete math course, and
3. the semester span between discrete math and algorithms.

The goal of the study is to analyze these additional features in statistical regression models to identify if there are predictors of academic performance that exist for the algorithms course. The dataset we use includes approximately 49,000 CS students from GSU, a US based, R1, diverse university with over 55% of undergraduate students being non–White, over 58% being female, and 30% first generation students receiving Pell grants. Through our investigation, we achieve our goal by finding that additional predictors of performance in our algorithms course do exist. We show that all features considered have impact on performance, however, we see that grades in the prerequisite course as the most dominant predictor, even more so than race and gender.

Using these findings, one can design and deliver qualitative measurements that support mixed methods research through CS course assessments and offer interventions targeting the subgroups as defined with combined co–features. Our approach can be used to analyze a combination of predictors for other CS courses, or it can be generalized for use in other disciplines of study.

5.1 Introduction

The study of algorithm design and analysis is an integral component of most CS undergraduate programs. A good foundation in this area bridges the early portion of the program to advanced areas in CS. Thus, it is imperative to determine the best approaches to overcome possible learning differences among student subgroups, especially for underrepresented groups in the discipline. By testing several regression models and identifying significant predictors of performance in algorithms, this approach can be generalized across many of the CS program's courses. Prior to the algorithms course, a pathway in most CS programs leads the student through a knowledge area (KA) of Discrete Structures, as described in the guidelines of the ACM and IEEE Computer Society curriculum report [27]. This path via discrete structures content can include programming classes, a discrete math course delivered from the math department or a logic-focused course delivered through the CS department and possibly a separate data structures course. The pathway we use is shown in Figure 5.1.

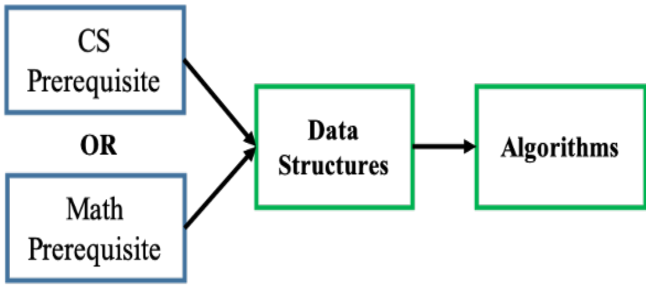


Figure 5.1: Pathway for Algorithms Course.

Our study tries to answer the following general research question:

RQ1: Do other features exist, beyond diversity and gender, that are predictors of performance in the algorithms course?

Once we answer this **RQ1** question, CS course assessments that involve related content can be designed and implemented as early interventions targeting our identified subgroups. Additionally, we discuss what quantitative and qualitative data we can gather and analyze that support our efforts to understand why some indicators work together better. Additionally, this work can show why some CS students finish their degree with overall higher performance compared to other CS students. Much like the work from Danielsiek et al., to determine CS students' misconceptions in an algorithms course using content topics [28], an example of similar work can be repeated and combined with our findings in subgroups as determined by methods we use to find performance predicting features from our algorithms course.

5.2 Background

Numerous recent works analyze influence upon retention from diversity population data in STEM. From their qualitative analysis, they present programs or interventions to address low populations of diversity groups that finish a STEM related Baccalaureate Degree. Hernandez et al. [29], found underrepresented students that participated in an undergraduate research experience along with faculty mentoring over several semesters increased their retention and graduation of a STEM degree. Research from Rheingans et al. [30], compares two groups, with one being in a Scholars program for STEM women and the other being their peers not participating in the program. Their intervention involved development of the Scholars students with leadership, mentoring, and community-building activities within the group. Their analysis covered ten years of data and shows the Scholars program yielding improvement in areas of grades, time-to-graduation, and retention. Townsend and Sloan [31], developed a similar Scholars program at DePauw University to increase participation of low-income, first-generation students interested in CS. Their in-depth work includes programs similar to Rheingans et al., but their focus is CS specific. They use a bridge program for the cohorts incorporating a Summer research project before their Fall matriculation.

Other research and intervention programs specifically focus on women in CS. These programs address gender differences and deeper beliefs and understandings of the causes of the low percentage of women in CS [32–35]. West et al. [36], research and address the lack of diverse employees, particularly women, in the artificial intelligence (AI) areas across numerous technology companies. Their report defines and examines diversity in more dimensions beyond race and gender. Their definitions include non-binary gender and power or position in the workplace, elucidating the harsh realities that AI is being designed, developed, and deployed in our society with very little input from diversity. The report solidifies that diversity issues in CS and technology need a very early intervention to reduce this widening gap.

One similar work from Babes-Vroman et al. [32], shows a comparative study of gender bias effect when taking a sequence of four CS classes that required CS prerequisite and subsequent courses. Their study involved binary gender and the effects within their sequencing pathway upon four specific CS courses: 1) Introduction to Programming, 2) Data Structures, 3) Computer Architecture, and 4) Algorithms. Their findings reveal that females with prior experience in computing, either self-taught or instructor led, as well as making a B or higher in Introduction to Programming, were more likely to continue with the Data Structures course. Their contribution shows a need for more recruitment of women interested in CS and to offer an introductory CS bridge course over the Summer before a new CS student's Fall matriculation [32].

As of late, diversity is a more complex issue in definition and biases in the technology field and many realize this at an early time in the CS person's education and profession as found in research from [33, 34, 37, 38]. We aim to support this research beyond race and gender with investigating the influence of additional student features. Lastly, a large study by Hellas et al. [39], from 2018, gives a systematic literature review of predicting academic performance, defining core study factors, and trends for values used in prediction research. The composition of observed disciplines in the study show 35% of the 357 reviewed papers are specific to CS from the authors' high-level synthesis of categorization factors. We identify our work to be aligned with their consensus of thematic analysis and preferred choice of data features found in our **RQ1**.

5.3 Methods

Our population source is GSU, a public US based, R1, diverse university with over 32% of undergraduate students being Black or African American, 8.5% Hispanic, over 58% being female, and over 30% being first generation. Additionally, our ranking recognizes us as one of the most ethnically diverse universities in the United States [11]. This is a close match to the population of underrepresented minorities in undergraduate CS programs, making it a prime set of students to analyze. The data originates from an extraction of our university's student data system. It is comprised of undergraduate students with a declared major of CS. The time span for our dataset begins with the Spring semester of 2018, and includes all other semesters going back to Fall of 2008. The attributes collected include demographics: gender, race, and transfer status, and student data: course identifier, the semester and year of each course, and their achieved letter grade in completed courses. All student classifications are combined in our dataset (freshman, sophomore, junior, senior) and we exclude any graduate student data. We create a computed attribute from the semester span between completing a prerequisite course from either the CS or math department, and successfully completing the algorithms course. We call the attribute 'Time-To-Algorithms' (TTA) with integer values in the range of: $[1, +\infty)$.

This large dataset of 236,266 records represents the varying course completion records for all CS declared majors over the time span. With this corpus, it was de-identified as one complete process as defined through our Institutional Review Board (IRB) lab protocol[1]. Amalgamation of the varying course records represents an entire corpus of 48,915 unique CS students. The subset of this as extracted and transformed gives us a total of 1,432 records representing CS students that successfully completed a prerequisite course from the CS (Cohort_CS) or math (Cohort_MA) department, a CS data structures course, and

[1]IRB: H19494

subsequently, the CS algorithms course. As shown in Table 5.1, the race groups are Asian, Black, other, and White. From Table 5.1, the study population is consistently distributed for the 'other' and 'White' categories, and has a difference in the 'Black' category, and significant difference in the 'Asian' category. We discover both female and male populations are consistently distributed among our cohorts. Interestingly, the transfer status between the two cohorts is a higher ratio for the Cohort_MA students.

Table 5.1: Population Comparisons.

Feature	Corpus	Cohort_CS	Cohort_MA
Asian	**16%** (7,635)	**30%** (245)	**39%** (136)
Black	**39%** (19,251)	**27%** (221)	**19%** (68)
Other	**13%** (6,286)	**10%** (84)	**11%** (37)
White	**32%** (15,743)	**33%** (263)	**31%** (109)
Female	**44%** (21,482)	**14%** (116)	**15%** (54)
Male	**56%** (27,422)	**86%** (697)	**85%** (296)
Transfer	**25%** (12,315)	**30%** (245)	**41%** (145)
Native	**75%** (36,600)	**70%** (586)	**59%** (205)

Our investigation methods use multiple regression models in Stata15 [40]. We analyze the μ *GPA* of the prerequisite course, diversity (race, gender, transfer status), and sequencing (TTA) that lead to the algorithms course. At our institution, we offer two options for the discrete math course, one taught in the CS department and the other in the math department (hereafter referred to as **Cohort_CS** and **Cohort_MA**, respectively). Both have the KA discrete structures content topics and lead to the algorithms course, however, in our institution, we observe the Cohort_CS have more foci of combinatorics, algorithmic analysis with *Big O*, and logic. Within the Cohort_MA, we observe heavy foci on mathematical proofs, less combinatorics and logic, and no algorithmic analysis. We introduce a control variable of race as Asian, Black, other, White, coded as ordinal values. A threat to validity of our data is being structured in its current form to not allow for race to be represented as dichotomous variables, e.g., Asian and non-Asian, Black and non-Black, etc.

The multiple regression models test the predictors, X_i, and illustrate the most statistically relevant one(s) for the outcome performance: (GPA_O) for each observation in the algorithms course. Our predictors are:

X_1 : race (Ordinal: Asian, Black, other, White)

X_2 : gender (Dichotomous: female or male)

X_3 : transfer status (Dichotomous: transfer or native)

X_4 : GPA_P of Cohort_CS or Cohort_MA (Ratio)

X_5 : Time-To-Algorithms (TTA) (Interval: *Low* (1-3), *Med* (4-5), *High* (6+) is the number of semesters between passing a prerequisite and the algorithms course.)

The various factors we observe and test derive from the model in Figure 5.2. The formulation is:

$$Y = \sum \beta_i X_i + \beta_0$$

with $Y = GPA_O$ and β_0 is the predicted starting value that minimizes the squared deviations between the predictor, X_i and outcome, GPA_O value for each observation.

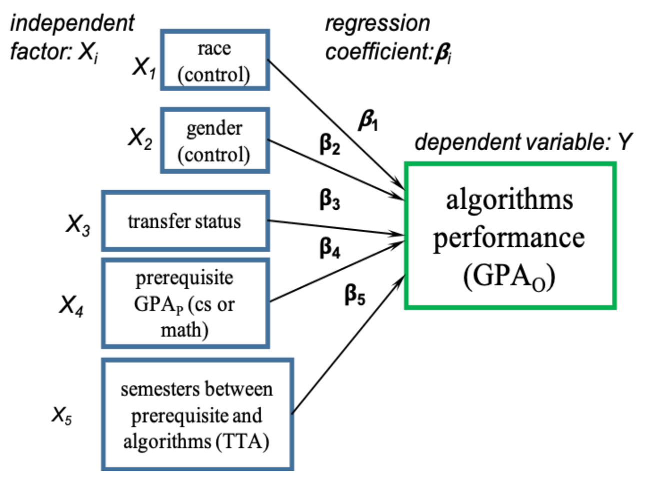

independent factor: X_i
regression coefficient: β_i
X_1
race (control)
X_2
gender (control)
X_3
transfer status
X_4
prerequisite GPA_P (cs or math)
X_5
semesters between prerequisite and algorithms (TTA)
β_1
β_2
β_3
β_4
β_5
dependent variable: Y
algorithms performance (GPA_O)

Figure 5.2: Multiple Regression Model.

Our analysis disproves the null hypothesis, H_0 and supports the alternative hypothesis, H_a. Our H_0 is: -there are no predictors (X_i), that influence the outcome GPA_O from the algorithms course. We express these as:

$$H_0 : h_1(t) = h_0(t), \qquad \text{for all } t \in [0, \rho]$$

$$H_a : h_1(t) \neq h_0(t), \qquad \text{for some } t \in [0, \rho]$$

High statistical power is determined with three independent factors, two controls, and initial sample sizes - Cohort_CS is 1,312 and Cohort_MA is 805. We perform numerous verification steps to validate the data sets for hetroskedasticity using Cameron and Trivedi's decomposition of IM-test and Breusch-Pagan/Cook-Weisberg test for constant variance [41, 42]. The IM-test reveals existence of hetroskedasticity in H_0 for each respective cohort. We also check the independence of observations using the Durbin-Watson statistic, and validate linearity using two-way scatter plots on all X_i predictors [43].

We determine the critical F value[2] is 4.3650, with Df_1=5, Df_2¿120 and α=0.05 (95.0% confidence level). Wald F-test values to validate significance are shown in Table 5.2. We identify and remove outliers using added-variable plots, and check the residuals are approximately normally distributed with a Normal P-P plot and Normal Q-Q plot. With these tests complete, our sample sizes become - Cohort_CS is 796 and Cohort_MA is 344 giving $N =$ 1,140. In Table 5.2, we observe the significance statistic value for each X_i and illustrate for the Cohort_CS, X_1 and X_4 do not support H_0, thus supporting H_a. The Cohort_MA shows only X_4 as supporting H_a, thus requiring deeper analysis in the model comparisons.

We analyze four regression models among the five X_i predictors (independent variables and control variables) with the first model as X_4 and subsequent models adding another X_i in each round of testing. Once each X_i becomes part of a test, all regression models with different values are cumulatively observed. From Table 5.2, we observe the β_0 values for Cohort_CS and Cohort_MA are not significantly different indicating the two cohort samples are similar. This indicates testing with the same multiple regression models with each X_i can be done in both cohorts to reveal their significant predictors.

[2] www.statisticshowto.datasciencecentral.com/tables/f-table

Table 5.2: F-test Values for Predictor Influence.

X_i	Data	$P > \|t\|$	Prerequisite Cohort
$\boldsymbol{X_1}$	**race**	**0.003**	**Cohort_CS**:
X_2	gender	0.168	F (5,790) = 25.59
X_3	trans	0.325	IM-test: ($P > 0.4528$)
$\boldsymbol{X_4}$	$\boldsymbol{GPA_P}$	**0.000**	$P > \chi^2 = 0.9592$
$\boldsymbol{X_5}$	**TTA**	**0.035**	$\beta_0 = 2.179133$
X_1	race	0.135	**Cohort_MA**:
X_2	gender	0.494	F (5,338) = 16.48
$\boldsymbol{X_3}$	**trans**	**0.046**	IM-test: ($P > 0.7079$)
$\boldsymbol{X_4}$	$\boldsymbol{GPA_P}$	**0.000**	$P > \chi^2 = 0.5628$
X_5	TTA	0.063	$\beta_0 = 2.204454$

Bold $\boldsymbol{X_i}$ indicate significant predictor.

5.4 Results

Analysis of regression models for Cohort_CS and Cohort_MA are shown in Table 5.3. In this table, we show the final regression model as it answers **RQ1**. We omit the prior multiple models for brevity. This final model for each cohort indicates the statistical influence of each X_i independent variable upon our dependent variable, GPA_O, the algorithms course final grade.

Table 5.3: Error Reporting and β_i Coefficients for Multiple Regression Model Testing (Robust Standard Errors in Parenthesis).

$\boldsymbol{X_i}$	Data	Final CS Model	Final Math Model
$\boldsymbol{X_1}$	**race**	0.0612**	0.040
		(0.021)	(0.028)
		$\boldsymbol{\beta}_1$=0.131023	$\boldsymbol{\beta}_1$=0.0731431
X_2	gender (female=1)	-0.098	-0.066
		(0.071)	(0.096)
		$\boldsymbol{\beta}_2$=-0.0470125	$\boldsymbol{\beta}_2$=0.1025506
$\boldsymbol{X_3}$	**transfer status** (transfer=1)	-0.032	-0.146*
		(0.053)	(0.073)
		$\boldsymbol{\beta}_3$=-0.0332792	$\boldsymbol{\beta}_3$=-0.0986949
$\boldsymbol{X_4}$	$\boldsymbol{GPA_P}$	0.299***	0.362***
		(0.033)	(0.045)
		$\boldsymbol{\beta}_4$=0.3063027	$\boldsymbol{\beta}_4$=0.396663
$\boldsymbol{X_5}$	**TTA**	-0.024*	-0.034
		(0.012)	(0.018)
		$\boldsymbol{\beta}_5$=-0.072960	$\boldsymbol{\beta}_5$=-0.0905834
	N	796	344
	R^2	0.127	0.187
	adj. R^2	0.122	0.175
	RMSE	0.681	0.659

Associated Table 5.2 - P > $|t|$: *$p < 0.05$, **$p < 0.01$, ***$p < 0.001$.

Cohort_CS: We observe in Table 5.3 for this cohort, the race, GPA_P (prerequisite course GPA), and TTA have influence on the performance in the algorithms course grade, GPA_O. Race and the GPA_P show positive influence, indicated where p and β_1 are positive. We interpret the higher the GPA_P value, a good performance is expected in GPA_O, whereas, the race attribute is positive and influential. We need deeper analysis to determine what specific race subgroups have higher influence on a good performance in GPA_O. Note the TTA has significance and both its p and β_5 are negative. Since we coded an interval variable, TTA, where '1' is *Low*, '2' is *Med*, and '3' is *High*, this means if a CS student has a '3' value for TTA, a poor performance (bigger negative influence) is expected in GPA_O. At this point, we need to mention one threat to our results is that observations of TTA *High* can include CS students that have failed and repeated the prerequisite course, the algorithms course, or are part-time CS students. This is a threat in most universities where students are given a 'repeat and replace' option or do not have capacities to be full-time students. If we observe Table 5.4, the μGPA_O is lower for all race and gender with the TTA *High*, except for the Asian and White females. Again, we need deeper analysis in the race and gender subgroups to determine the bigger influences upon μGPA_O where co-features of these two work better together as a predictor of performance in the algorithms course. Here is where the regression model data and subgroup population dissections indicate a need for personalized instruction targeting these subgroups (Black/other female with TTA *High*) with interventions and qualitative measurements to study any influences upon their GPA_O values.

Cohort_MA: Similarly for this cohort, we observe in Table 5.3, the GPA_P and transfer status have influence on the performance in GPA_O, however, race and TTA do not. This interprets as the higher a GPA_P value, a good performance is expected in GPA_O. The transfer status is coded as '1' where a student is transfer. We see a negative value for p and β_3 implying when a student is transfer, a poor performance impact in GPA_O is expected. If we observe Table 5.4, the GPA_O behavior is similar to the Cohort_CS where TTA *High* lowers the μGPA_O. Shown in Table 5.1, we see group percentages across the two cohorts are similar demographically, and exhibit similar characteristics with the μGPA_O, but our regression model does not place significance on the TTA in the Cohort_MA. This area indicates further a need for inferential analysis with testing new regression models and separating different race groups as dichotomous, gender subgroups with co-features, and TTA to determine this behavior.

Table 5.4: TTA μGPA_O for Cohorts (CS and MA).

Race	CS	CS	CS	MA	MA	MA
Gender	Low	Med	High	Low	Med	High
Asian M	3.28	3.22	2.89	3.25	3.24	2.97
Asian F	3.21	3.10	3.33	3.44	2.93	3.00
Black M	3.07	3.00	2.79	3.07	3.07	2.57
Black F	2.97	3.13	2.56	3.83	3.13	3.35
Other M	3.11	3.08	2.90	3.29	3.10	2.94
Other F	3.27	3.00	2.50	4.30	3.39	3.35
White M	3.34	3.55	3.25	3.41	3.37	2.95
White F	3.07	3.32	3.68	3.46	3.18	3.23

In support of our alternative hypothesis, H_a, we see there are alternate X_i predictors that influence GPA_O of the algorithms course in the observations and thus reject the null hypothesis, H_0. This gives us validation in answering **RQ1**.

5.5 Conclusions

In this study we have analyzed and discussed CS specific students and identified the most influencing features affecting the performance in subgroups taking an algorithms course. We have answered **RQ1** with statistical regression models showing features of diversity, gender, prerequisite GPA, transfer status, and TTA are predictors of subgroup's μGPA_O in an algorithms course.

Within our population, we identified two cohorts based upon a prerequisite course of discrete math taught in the CS and math departments. With these two cohorts, we created subgroups through testing of multiple regression models using the Stata15 statistical software [40]. Taking our evidence of the most influential predictors, β_i, future work can develop and evaluate targeted interventions to address deficit areas in the subgroups. Also, removing the part-time CS students can add strength to our dataset for these subgroups in showing a truer representation of TTA impact. Taking proven predictors in race and gender subgroups and giving them extra interventions to promote higher performance in a prerequisite course can increase performance in the subsequent algorithms course. New research in areas of undergraduate student prediction models for retention, course planning, and time-to-degree as found in [44, 45] use data analytics of student data across all disciplines in large R1 universities. Combining existing general student prediction models with our CS focused research can be specifically applied towards CS students to predict retention, the time-to-graduation, and course performance. Considering the big data set that we used, it would be beneficial to design and study personalized course plans, and determine if there are possible longitudinal trends among the CS student subgroups.

Unquestionably, there is a need for more qualitative data research at the beginning of a CS student's journey, especially in our identified subgroups. More recent reports and articles from the ACM Retention Committee [46–48] show even smaller numbers of Black and Latinx students at an average of 7.45% graduation with a CS degree from 2015. This ACM report explains an amorphous problem in understanding the undergraduate CS student. Our work reveals features of these students which can aid in propagating the pipeline of CS graduates for a more diverse computing workforce.

6 TEACHING STRATEGIES IN SOFTWARE ENGINEERING TOWARDS INDUSTRY INTERVIEW PREPAREDNESS

The Software Engineering (SE) curriculum in undergraduate computer science (CS) education is designed to train students in the process of software and systems development. Traditionally, topics such as software development methodologies, industry nomenclatures, and solution analysis are delivered through lectures and group projects. We propose a novel approach in teaching SE that we call MACROVR: MAchine learning to select project team members; Cloud technologies required for project control, code versioning, and team communications; ROtational schedules in Agile/Scrum roles; an individual Video of the team project story board; and Rubrics for all presentations. Our teaching strategy with this approach utilizes the latest technologies currently employed in industry and corresponds to soft skills commonly assessed in interviews.

The goal of our study is to measure if using the MACROVR approach contributes to preparedness for a computing job interview. Most often, this course is taken towards the end of a four-year CS degree program while students are job hunting or seeking an internship in the computing industry. We use an anonymous, fifteen question survey instrument sent to volunteers that indicated they are seeking a computing job and have successfully completed the SE course. The sample is comprised of three sections of the SE course using the MACROVR approach (135 students) and four sections that did not use all of the required strategies and technologies, which we call MACROVR-lite (184 students). Our two cohorts, MACROVR and MACROVR-lite, are each given the same survey questions. We analyze their Likert scale data responses using non-parametric methods. Our findings indicate the

MACROVR approach better prepares students with the skills and highly valued qualities for success in computing industry interviews.

6.1 Introduction

According to a Collabnet/VersionOne[1] 2019 report, when developing software systems, 97% of industry uses the Agile methodology and of those, 54% employ the Scrum technique. This has been a shift from the previous waterfall model methodology as shown in Figure 6.1. Recognizing this change, we see that many Software Engineering (SE) courses have adapted their course content focus on a more industry-relevant Agile methodology.

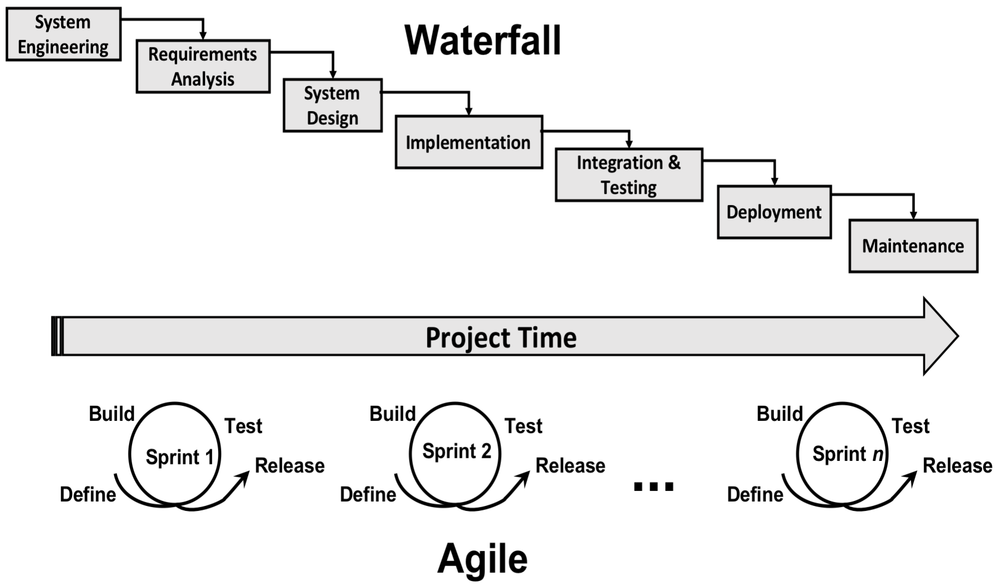


Figure 6.1: Two Models of Software Engineering Methodology.

We posit a CS student is better positioned to enter the workforce with a career in computing when the latest technologies are utilized in the classroom, in conjunction with training students to efficiently work in a team environment. In the case of a SE course, this would be Agile with Scrum, along with project based learning, using a version control system for team

[1]https://explore.versionone.com/state-of-agile/13th-annual-state-of-agile-report

code sharing, managing the system through a cloud-based project management system, and incorporating key areas of team experiences in collaboration and communication. Searching through several career opportunity resources seeking SE positions, such as GlassDoor[2], Indeed[3], and ACM Career and Job Center[4], we find key areas like Git, Agile/Scrum, team problem solving, and web services being advertised as desirable skill sets and knowledge for new hires in industry. In 2017, Ford et al., concluded that technical interviewers care greatly if candidates can demonstrate CS technical skills with verbal and written clarity [49]. Our MACROVR approach teaches beyond technical skills to include team presentations among peers, team collaboration and communication, and working through group consensus to deliver a software system, thus preparing them for the soft skills interview areas. Much research has looked at incorporating one or two of these components in SE courses, but do not analyze any impact upon a student's industry readiness [50–55]. We choose to integrate Agile/Scrum components in combination with the complementary skills mentioned above and then evaluate if this approach helps prepare students for a computing job interview.

Undeniably, knowledge and experiences CS students obtain from courses in their degree program is critical to articulate their skill and value during a computing job interview. Our teaching strategy incorporates additional techniques in the SE course to aid in SE career readiness. In our course, we employ machine learning (ML) to form teams in project-based learning (PBL), require cloud technologies for project management, rotational Scrum roles, detailed rubrics, and individual and student team presentations of their project. By integrating these teaching strategies and technology requirements into our SE course, students are exposed to a wide view of current practices of the computing industry.

Components in the MACROVR approach are chosen based upon the author's past twenty-three year industry experience as a software engineer in a major Fortune 500 US corporation and several US federal government agencies. They used many Agile/Scrum techniques in their job, managed software development teams, and conducted numerous

[2]https://www.glassdoor.com/Job/jobs.htm?sc.keyword=software+engineer
[3]https://www.indeed.com/q-software-engineer-jobs.html
[4]https://jobs.acm.org/jobs/results/keyword/Entry+Level+Software+Engineer

technical and soft skill interviews for new CS graduates seeking a computing position. We concur and include researched areas and attributes as being important to computing job interviews and industry skills as found in [52,56]. Recent, seminal work from Oguz et al., find the perspectives of industry identify recent graduates with a lack of soft skills, limited use of current tools, and little exposure to real life projects as important gaps between SE education and workforce [57]. Again, we use countermeasures in these areas with our approach through team presentations among peers, required technology, and instructor vetted projects. The CS technical skills students acquire from CS program courses like algorithmic computations, object-oriented coding principles, and system architectures are considered to be utilized by all students in any SE course, so they are not studied specifically in the MACROVR approach.

Our motivation is to evaluate if teaching real-world SE experiences through MACROVR contributes to a CS student's career readiness. We measure the effect of our strategy through a fifteen question survey, approved by GSU's IRB[5], using a five-point Likert scale [58]. To guide our research, we develop the following research question:

RQ2: Does teaching SE using the MACROVR approach better prepare students for a computing job interview?

To answer our research question, our survey asks CS students that finish the SE course if they feel our teaching strategies help with career readiness by preparing them for computing job interviews. This voluntary, anonymous online survey contains questions relating to specific MACROVR course components and some that are common in all SE courses taught at our university. We form two cohorts where the first is students that successfully complete the SE course requiring all components of the MACROVR approach be used (three sections, 135 students). The second cohort is students that successfully complete the SE course where the instructors may or may not use all the MACROVR approach components (four sections, 184 students). We call these cohorts MACROVR and MACROVR-lite, respectively. All survey responses result from semesters of Fall 2017, through Fall 2019, from the SE courses.

[5]IRB: H19266

We do not include the Summer semesters due to the shorter time for delivery of the course content, only one section being offered, and typically being taught by a graduate teaching assistant. The semesters being surveyed in this work are taught, face-to-face by faculty instructors, usually three each Fall and Spring semester. All volunteers taking the survey either graduated with a four-year CS degree and were actively looking for a full-time computing job through interviews or remained enrolled while actively interviewing to secure an internship in computing. The same survey is given to both cohorts to preserve our experimental group (MACROVR) and the control group (MACROVR-lite). Since the control group is taught without requirements of using all the MACROVR components, the data results indicate which ones may have been used when comparing to our experimental groups' results. Our results show when the experimental group is taught with all required components of the MACROVR approach, students are better prepared for a computing job interview.

6.2 Background

Much research in SE education relates to learning strategies, active teaching techniques [59], investigation on gaps between academia and the software industry [57], and authenticity of the SE course content [56]. Many report research on the effectiveness when using Project-based learning (PBL) in SE education [56,60,61]. Garcìa-Peñalvo et al. in 2019, research the broader area of active learning that includes PBL and concludes SE courses using PBL give students a better understanding of the concepts involved when used in the classroom [62]. Additionally in 2018, Garcìa-Holgado et al., found using similar teaching strategies increased student's performance in SE courses by 20% [63]. We adopt a similar pedagogical strategy in our research and introduce several new techniques within PBL and then measure their effect related to our goal - CS student preparedness for computing job interviews.

One strategy researched is collaboration in SE teams [64]. Chowdhury et al., use the IBM Watson^{TM} Personality Insights service to analyze their student teams use of a collaboration platform from Slack[6]. They find the participation of a teams' communications highly correlate to their self and team evaluations, both good and bad, as well as their team grade for each Agile/Sprint cycle in the SE course. We require the use of Slack's team collaboration platform for our SE course in the MACROVR approach.

Teaching SE while using a software version control system such as GitHub[7] has been shown to benefit students [52]. Feliciano et al., find there are numerous challenges with using GitHub as a learning platform and its wide use in industry makes it an important part of the CS student's SE education. They also expand its use beyond the team project work, utilizing it to disseminate course assignment materials and lab content with related discussions. They conclude the students benefit from using this platform and incorporate their team content as part of their personal, online presence. Eraslan et al., find many problems using Git based version control systems in teaching SE [53]. They identify seven categories of errors and poor practices from students using Git. Their research is the most recent (2020) and comprehensive list of Git usage artifacts in the literature and gives us a new basis to consider Git as an integral part of SE education along with measurable instruments. By contrast in the MACROVR approach, we require Git usage in all project teams and require the instructor be a member of each to monitor submissions of a team's code and documentation, indicating levels of progress.

We find formation of student teams a significant factor for SE project success as reported by [50, 65–67]. Dzvonyar et al., develop and use a hierarchical criteria for team composition and through a manual selection process by instructors, form the student teams [50]. Bosnic et al., conduct a ten-year study of matching student teams to projects and find the 'first come, first serve' method or teaching staff directed were not effective [68]. They realize a system of student pre-course questionnaires and student's proposals of projects based on platform groupings (standalone, web, or mobile) as a better solution. Conversely, we form teams using several ML algorithms basing on student skills, CS aptitudes, and semester course load. We

[6]https://slack.com/features
[7]https://github.com/features

design projects requiring specific component technologies that all students enrolled in the course should possess, and teams choose a project using a lottery system. We offer details of these component technologies in Section 6.3.2, and sample project descriptions in Section 6.3.4.

Soft skills in SE education are considered important for student preparation and readiness for industry. Abad et al., analyze data gathered from SE students over three semesters and investigate the amount of authenticity that can be achieved in PBL courses in SE education [56]. Their work investigates seven 'authentic' activities with *Why?*, *What?*, and *How?* explored in each activity. Interestingly, their work indicates 'soft-skills' like *planning*, *problem understanding*, *negotiation*, and *organizational* as the most statistically significant for CS students to be prepared for industry.

Hogan et al., find CS students presenting their work in front of peers improve soft skills like communications [69]. We posit our MACROVR approach supports soft skills and amplifies them through teaching elements requiring a rotation of Scrum roles: -Project Manager, Scrum Master, Senior Programmer, Junior Programmer, QA/Testing, in each Sprint cycle. This encourages students to work on different aspects of their project for a period of time and broaden their understanding of software engineering principles.

Through our literature review, we are not able to identify any other research studies that consider a holistic approach in delivering the SE course, to better prepare students for the workforce. We find most research focuses on just one or a few strategies. We incorporate into our MACROVR approach numerous teaching strategies, technical requirements, and communication skills to ready a CS student for their first real-world experience, namely, a computing job interview.

6.3 Course Structure

The SE course using our MACROVR approach has several foundations to enable the project team formation and introduce project technology requirements. Using a variety of strategies, we organize the course to simulate a real-world environment where software engineers face many abstract ideas, must use critical thinking to understand project team goals, and must integrate/communicate with a group of unfamiliar individuals. Technology requirements in our approach include a cloud-based version control system with bug reporting, an Agile/Scrum management system, and integrating a communication system through using an instant messaging system.

In the first two weeks using MACROVR, we introduce concepts of Agile/Scrum in lectures and give three short assessments to build the ML models for project teams formation. We discuss the technical requirements for projects like web services, a database element, and using an object oriented programming (OOP) language. The second week has team formation and team meeting time given at the end of each class session (usually fifteen or twenty minutes) for questions with the instructor, to establish meeting times for each group to interact face-to-face, and give the instructor interactions with the teams to check their project's current sprint progress. At the beginning of each sprint cycle, the instructor gives a rubric for appropriate preparation time and provides a partial structure to complete their tasks as shown in Figure 6.2. At the end of the semester, each student prepares an individual video of the project story board according to the rubric as shown in Figure 6.3. The rubrics serve as an important component of the MACROVR approach and is similar to the project requirements or specifications one would see in industry.

Points	Timing	Content	Demonstrate Product Knowledge	Architecture and Interfaces	Skill and Look of Presentation
Level 4 20 points	Effectively use between 6 and 8 minutes. PM and SM are equally presenting content.	**Discuss These:** • Burn down chart. • Velocity chart. • Completed Scrum stories. (Not necessary for support tasks.) • Backlog stories and tasks (If none, why?).	PM can freely describe the overall purpose of the system and SM can do the same for technology and/or integration of the system.	**Architecture Pattern:** • Declare why you choose a particular pattern. • Show a clear and detail drawing (by hand) of your pattern. **Interface Model(s):** • Clear and readable interface objects (drawn by hand). • Brief explanation of where used in the system.	• Clear, concise with an appropriate amount of text. • Ease of answering questions. • Honesty of answering questions or saying you do not know the answer. (If you do not know, be honest and say so.) • You do not look as the screen when talking. (Talk to audience.) • No use of animation. • No blue or green text. • Readable photo / screen shots (Big enough and clear if hand drawn.)
Level 3 17 points	Use between 6 and 8 minutes. PM and SM are **NOT** equally presenting content.	**Discuss These:** • Burn down chart. (If none, why?) • Velocity chart. (If none, why?) • Completed Scrum stories. (Not necessary for support tasks.)	You can describe the overall product and challenges without reading the screen text verbatim.	**Architecture Pattern:** • Show a clear and detail drawing (by hand) of your pattern. **Interface Model(s):** • Clear and readable interface objects (drawn by hand).	• Clear, concise with an appropriate amount of text. • Ease of answering questions. • Honesty of answering questions or saying you do not know the answer. (If you do not know, then be honest and say so.) • You read text verbatim from the slides. (Talking to audience.) • No use of animation. • No blue or green text. • Unreadable photo / screen shots.
Level 2 10 points	Use between 4 and 6 minutes. PM and SM **ARE** equally presenting content.	**Discuss These:** • What is the product? • What does it do? • Is it mobile app or other? • Completed Scrum stories. (Not necessary for support tasks.)	You are reading the screen text while facing away from the audience.	**Architecture Pattern:** • Declare why you choose a particular pattern. • Show a clear and detailed drawing (by hand) of your pattern.	• Slides have too much text. • Ease of answering questions. • Looking at the screen when talking. (Does not apply to writing on the white board.) • You read text verbatim from the slides. (Talking to audience.) • Use of animation. • Use of blue or green text. • Unreadable photo / screen shots.
Level 1 6 points	Use between 4 and 6 minutes. PM and SM **ARE NOT** equally presenting content.	**Discuss These:** • What is the product? • Completed Scrum stories. (Not necessary for support tasks.)	You appear lost and not knowing what is going on in your project.	**InterfaceModel(s):** • Clear and readable interface objects (by hand). • Brief explanation of where used in the system.	• Slides have too much text. • Looking at the screen when talking and reading the slides. (Does not apply to writing on the white board.) • Use of animation. • Using many different colors or hard to read like blue or green. • Unreadable photo / screen shots.

Figure 6.2: End of Sprint Rubric for Grading Team Project Presentations.

Points	Timing	Content	Demonstrate Product Clearly and Coherently	Skill and Look of Demonstration Video
Level 4 25 points	Effectively use between 10 and 15 minutes presenting your project and product.	• What is the product? • What is the audience for the product? (Targeted users.) • What does it do? (Or supposed to do?) • What are the technologies you used to build the product? • What makes it worthy of being built? (Market importance.)	• You can freely show all the proposed and completed features of your product. (Think test scenarios of all the stories that are completed.) • The demonstration is uniquely yours. (Not a voice over with showing someone else doing the demonstration.) • The demonstration matches (actions, reactions) what you are saying as it is recorded. • The quality is such that your picture is always shown and sized to not block the screen actions or slides as it is recorded. • The demonstration is well done where there are no technical problems with the product, you don't REPEATEDLY say filler words (umh, like you know, as you see here).	• Clear speaking with showing high comfort levels using the software. • No pauses and no bugs recorded in the demonstration. • One continuous recording, not segments put together. • Sound is understandable and lighting is good. • Your skill of demonstration is well rehearsed and you show confidence in the video.
Level 3 18 points	Effectively use between 8 and 10 minutes presenting your project and product.	• What is the product? • What is the audience for the product? (Targeted users.) • What does it do? (Or supposed to do?) • What are the technologies you used to build the product?	• You can freely show all the proposed and completed features of your product. (Think test scenarios of all the stories that are completed.) • The demonstration is uniquely yours. (Not a voice over with showing someone else doing the demonstration.) • The demonstration matches (actions, reactions) what you are saying as it is recorded. • The quality is such that your picture is always shown and sized to not block the screen actions or slides as it is recorded.	• Clear speaking with showing high comfort levels using the software. • No pauses and no bugs recorded in the demonstration. • One continuous recording, not segments put together. • Sound is understandable and lighting is good.
Level 2 10 points	Effectively use between 6 and 8 minutes presenting your project and product.	• What is the product? • What is the audience for the product? (Targeted users.) • What does it do? (Or supposed to do?)	• You can freely show all the proposed and completed features of your product. (Think test scenarios of all the stories that are completed.) • The demonstration is uniquely yours. (Not a voice over with showing someone else doing the demonstration.) • The demonstration matches (actions, reactions) what you are saying as it is recorded.	• Clear speaking with showing high comfort levels using the software. • No pauses and no bugs recorded in the demonstration. • Sound is understandable and lighting is good.
Level 1 6 points	Effectively use between 6.5 and 4.5 minutes presenting your project and product.	• What is the product? • What is the audience for the product? (Targeted users.)	• You can freely show all the proposed and completed features of your product. (Think test scenarios of all the stories that are completed.) • The demonstration is uniquely yours. (Not a voice over with showing someone else doing the demonstration.)	• Clear speaking with showing high comfort levels using the software. • No pauses and no bugs recorded in the demonstration.

Figure 6.3: End of Semester Rubric for Grading Individual Video Presentations.

6.3.1 Project Team Formation

With the MACROVR approach, we form project teams by ranking the students using several ML models. Based upon a student's current course workload and three individual assessment scores we develop a heuristic for choosing students for a project team as shown in Figure 6.4. Most class populations are close to a multiple of five, allowing five or six students per team. Typically, a MACROVR class size is fifty and we compute the total number of combinations (*n=50, r=5*), to form five-person teams as 2,118,760 choices. This computation formula is:

$$C(n, r) = \frac{n!}{r!(n-r)!}$$

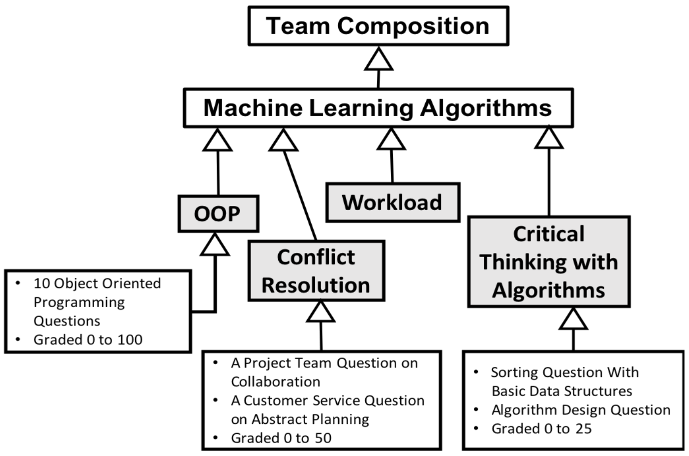

Team Composition
Machine Learning Algorithms
OOP
Workload
Conflict Resolution
Critical Thinking with Algorithms
• 10 Object Oriented Programming Questions
• Graded 0 to 100
• A Project Team Question on Collaboration
• A Customer Service Question on Abstract Planning
• Graded 0 to 50
• Sorting Question With Basic Data Structures
• Algorithm Design Question
• Graded 0 to 25

Figure 6.4: Assessment Structure for Team Composition.

Using ML with these four attributes, we create supervised training data through ranking some students as one (novice) to five (very knowledgeable) based on their scores and course workload in contrast to [50, 68]. Using five ML classifiers: 1) Gaussian Naive Bayes, 2) K-Nearest Neighbor, 3) Decision Tree, 4) Nearest Centroid, and 5) Linear Discriminant Analysis, we take the arithmetic mean of these five rankings for grouping into classifications, one to five. We cross-validate our ML algorithms using the holdout method with a 20 / 80 split. After teams selection, we make a schedule of rotations with respect to all Scrum roles (project manager, Scrum master, senior programmer, junior programmer, QA/tester) and give teams the rubric for their first project presentation. We show a sample of one team's rotation schedule in Table 6.1. We did not include race and gender attributes in the team formation algorithms or as independent variables in our regression analysis because this research intent is examining teaching strategies and using technology components, then measuring their effect for CS student readiness with computing job interviews in the workforce.

Table 6.1: Sample Team Rotation Schedule with Scrum Roles.

Team*X*	**Sprint1**	**Sprint2**	**Sprint3**	**Sprint4**	**Sprint5**
Person1	Project Mgr*	Scrum Master*	Senior Prog.	Junior Prog.	Testing and QA
Person2	Scrum Master*	Senior Prog.	Junior Prog.	Testing and QA	Project Mgr*
Person3	Senior Prog.	Junior Prog.	Testing and QA	Project Mgr*	Scrum Master*
Person4	Junior Prog.	Testing and QA	Project Mgr*	Scrum Master*	Senior Prog.
Person5	Testing and QA	Project Mgr*	Scrum Master*	Senior Prog.	Junior Prog.

* indicates a team presenter (active role) at end of sprint

6.3.2 Team Project Requirements

In the second week, the class instructor presents a pool of team projects with descriptions and recommendations of platforms for implementation (samples shown in Section 6.3.4). Each team randomly selects a sealed, envelop with a number forming a lottery selection from the list of projects. Most are web, but several are desktop or can be mobile app centric. Each must incorporate at least one web service API, and a database (SQL or No SQL) to maintain application and/or user state. The instructor gives a brief project description, possible associated web service API, and other requirements like GUI based interfaces, and recommended OOP languages. Some script languages like JavaScript are allowed, but cannot not comprise the majority of the code base.

Technology requirements for projects in our MACROVR approach include cloud-based version control management (GitHub[7]), a group communication systems (Slack[6]), and the project's progress and activity management using Scrum (ZenHub[8]). End of sprint cycle's team presentation incorporates SE course content timed with teaching progress like: conceptual models, test case scenarios with test cases, UML activity diagrams, object class diagrams, and UML state diagrams specific to their project as well as metric reporting (burn down and velocity) of their project progress from the ZenHub system.

The first sprint involves building a Scrum backlog with user stories mostly comprised of investigation and research for the project's technology requirements and setup of the team members in GitHub, ZenHub, and Slack. Additionally, the first sprint requires a global system concept model review by the instructor before their first presentation (end of sprint one) to allow constructive feedback. This is important to start each team with a realistic and correct direction for their system development and implementation in the lifetime of the course.

Grading from the rubric requires each project team to upload their presentation files into our learning management system (LMS) for feedback and suggestions to improve soft skills areas for the next sprint presentation. A rubric for the team presentation is shown in Figure 6.2. Some research indicates that collaboration, fair cooperation, and communication are the most important student skills and is a problem in teams of PBL, especially in SE education [64]. In order to mitigate some of these negative effects, the MACROVR strategy uses a sliding scale of grade impact from each end of sprint presentation. The more active

[8]https://www.zenhub.com/product

roles (project manager and Scrum master) carry more responsibility, thus have the majority of earned points for the grade of a presentation. We weight the project manager at 45%, the Scrum master at 25% and other members equally at 10% each. Grading with this style of scaled points puts emphasis on active and passive roles during a sprint. It encourages team leadership and strong communications from the active roles. Referring back to Table 6.1, we ensure each team member will have a chance to experience all team roles in the Scrum methodology over the semester time period. The total course grading for a student is weighted as 50% for presentations with the rotating Agile roles and 50% equally split among exams.

6.3.3 Survey Questions

Each Likert scale question gives the participant a choice: *Strongly Agree*, *Agree*, *Neutral*, *Disagree*, *Strongly Disagree*, or *N/A*. We separate the responses for analysis in two cohorts from students in MACROVR or MACROVR-lite SE courses taught in Fall 2017, through Fall 2019. Both cohorts receive the same set of questions to allow us a statistical comparison in their opinions, interests, or perceived efficacy as it relates to the **RQ2**.

From the fifteen survey questions, we show in Table 6.3, there is a threat of Type 2 errors in our analysis to use a parametric testing model with the distribution of student responses being unevenly distributed. We observe several histogram graphs to verify this non-normal distribution. We choose a two non-parametric test, Kruskal-Wallis[9] to analyze our data. The general descriptive analysis of our n=319 is shown in Table 6.2, and in Section 6.4 we interpret the results of our tests.

[9]https://www.statisticshowto.datasciencecentral.com/kruskal-wallis/

Table 6.2: Survey Data Composition.

Event	**MACROVR**	**MACROVR-lite**
Sent	135	184
Complete	87	29
Incomplete	18	1
Refusal	5	3
Participation	64.4%	15.8%

The percentage of participation from our cohorts, shown in Table 6.2, indicate instructors of MACROVR-lite sections of the course may not have encouraged their students to respond to the survey with similar enthusiasm as instructors of the MACROVR sections. We present the data separately as percentages based upon each respective cohort's participation shown in Figures 6.5 and 6.6.

Table 6.3: Survey Questions.

No.	Survey Question
(1)	The use of team projects better prepared me for interviews.
(2)	The use of Agile and Scrum methodologies in class better prepared me for interviews.
(3)	The single and final project presentation in class better prepared me for interviews.
(4)	The topics covered in class better prepared me for interviews.
(5)	The single and final project document better prepared me for interviews.
(6) †	The Agile SCRUM team roles being rotated better prepared me for interviews.
(7) †	Having end-of-sprint presentations better prepared me for interviews.
(8) †	Using cloud based project and control management tools better prepared me for interviews.
(9) †	The in-class team activities related to Agile and Scrum better prepared me for interviews.
(10)	The in-class team activities related to course topics (UML, test cases, state models) better prepared me for interviews.
(11) †	Using a web API (remote) better prepared me for interviews.
(12) †	Incorporating a database system into my project better prepared me for interviews.
(13) †	Incorporating a cloud based system for project team communications better prepared me for interviews.
(14) †	Incorporating a GIT / repository system better prepared me for interviews.
(15) †	Creating a video of the project storyboard better prepared me for interviews.

† MACROVR specific survey question.

6.3.4 Sample Projects

Several successfully completed applications by student project teams are given from the instructor list below. We use these and other projects in the MACROVR approach. Note that some require multiple teams communicating and coordinating project resources among two or three teams. This requires a small amount of product management from the instructor to help with setup and configuration as a 'monitoring' member of their Slack, GitHub, and ZenHub systems. Additionally, the instructor helps the multiple team projects with the abstract concept of a 'test harness interface' to initially connect different applications before full implementation.

Our sample list of team projects include:

- **Attend-In:** Mobile (Android) application to use the geo-location from a student's mobile device's IP address for time and attendance in college classrooms. It is designed so an instructor can setup their classes from a web browser or desktop application, class times, and campus buildings. The application will convert building address to geo-location. The web services recommended is IPStack[10]. This project has two teams requiring combined work. One for the Android application built using Android Studio, one for an instructor application with back end database in a web hosted system like Google Firebase[11].

- **Client Care:** A desktop or web browser application to help a non-profit organization with client intake and management for providing services. One team to build a system to register a client's information and show if already registered for an available service. A second team for back end database and GUI to manage the services. This non-profit might offer services to include a food pantry, educational class for: 1) personal finance, 2)resume writing, 3) job interviewing and 4) MS Office training, and a clothing closet for professional and business clothing. Recommended web service would be screen

[10]https://www.gps-coordinates.net/gps-coordinates-converter
[11]https://firebase.google.com/docs/hosting

scraping for food banks data[12] or interview skills on Udemy[13] to give case workers a resource in the application.

- **Baby Buddy:** Mobile (Android) application to use near field communications, (NFC) with a RFID tag to log your baby or pet as an occupant of your vehicle. It detects movement with accelerometers once baby registers as 'checked in' and when stopped. If Baby-Buddy is not checked out through the NFC tag and vehicle is stopped after a time limit, alarms are sent to registered contacts and 911 at the central system. This is all potentially hosted from Amazon Web Services. The same database will be used for check-in and check-out of baby's NFC tag. Recommended web services are Google GPS on the mobile application and Amazon web services for texting and database. This project has two teams requiring combined work. One for the Android application using Android Studio and a second for the back end database and communication to the web services.

- **Food Oasis:** Mobile (Android) application to find small popup or unknown food sources offering quality items in food deserts and swamps. A good example is at Georgia Food Oasis in FaceBook, but it requires a FaceBook account and uses your personal data and computer GPS location data in undisclosed ways. The design is a small footprint of user information for finding good, fresh food sources with web services of Google GPS and Firebase. This project has two teams requiring combined work. One for mobile app and a second for the back end database and communication to the food sources. The back end system will need a GUI to register and maintain food sources (vendor data).

- **Food Saver:** A complimentary project to Food Oasis. This is a desktop or web browser application to allow grocery stores or popup food providers a mechanism to register themselves as offering fresh, quality food. Recommend web services include

[12]https://www.feedingamerica.org
[13]https://www.udemy.com

Google GPS and Google Firebase. One team can build the desktop or web browser application that will connect to a back end system built by a team in Food Oasis.

6.4 Results

In this section, we present our statistical analysis, show our descriptive and inferential data, and discuss threats to validity. The composition of data from our cohorts' responses are previously shown in Table 6.2. We present the descriptive analysis in Table 6.4, and inferential analysis in Table 6.5, where we indicate the survey question's correlation to our **RQ2** and show the most influential indicators. Lastly, we discuss threats to the validity of our findings, both internal and external.

6.4.1 Interpretation of Results

The highest mean score, and closest to *Strongly Agree* response, are questions 2, 12, and 14; -two requirements, 12 and 14 are in the MACROVR approach. Question 2 indicates a high mean score and if we look at its Likert scale scores in Figure 6.5, the MACROVR cohort shows there are 0% *NA* responses and the third highest, 55.4% *Strongly Agree* in the entire corpus. The highest *Strongly Agree* response, 59.3% is in the MACROVR cohort indicating use of a database component in the MACROVR approach realizes a benefit in computing job interviews. This high mean value for question 12, could result from students taking a database fundamentals course prior to the SE course being studied. Inferential analysis of question 11, using a Web API, in both cohorts indicate highly significant (Table 6.5), but descriptively, the MACROVR approach shows as a much higher correlation; -*Strongly Agree*, 36.4% and *Agree,* 31.8% in contrast to MACROVR-lite *Strongly Agree*, 14.8% and *Agree*, 22.2% respectively. Simple descriptive analysis supports our **RQ2** showing the MACROVR approach responses (x=87) is three times the amount of the MACROVR-lite approach responses (x=29).

Table 6.4: Survey Question Descriptive Statistics.

No.	Obs	Mean	Std.Dev	Min	Max
(1)	111	3.982	1.206	1	5
(2)	107	4.093	1.178	1	5
(3)	109	3.817	1.270	1	5
(4)	111	3.901	1.206	1	5
(5)	108	3.593	1.215	1	5
(6) †	108	3.796	1.372	1	5
(7) †	108	3.935	1.162	1	5
(8) †	99	3.929	1.197	1	5
(9) †	98	3.755	1.293	1	5
(10)	108	3.954	1.171	1	5
(11) †	98	3.878	1.204	1	5
(12) †	109	4.312	0.959	1	5
(13) †	100	3.890	1.180	1	5
(14) †	109	4.156	1.148	1	5
(15) †	93	3.419	1.362	1	5

† MACROVR specific survey question.

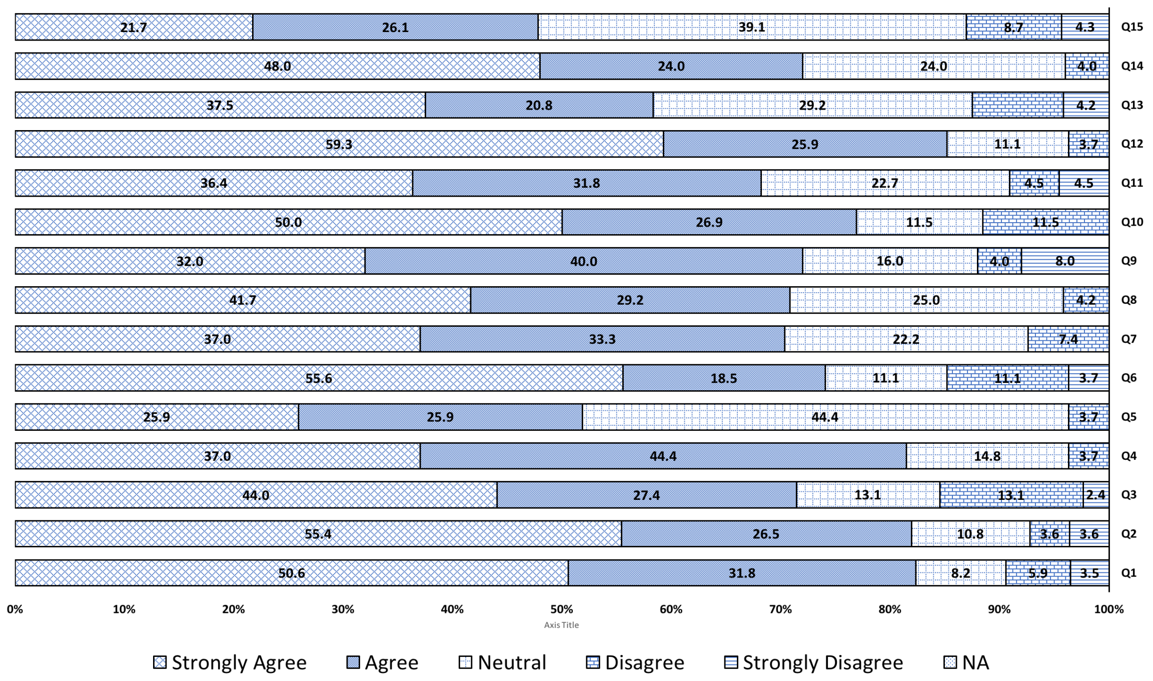


Figure 6.5: Percentages of MACROVR Survey Responses by Likert Value.

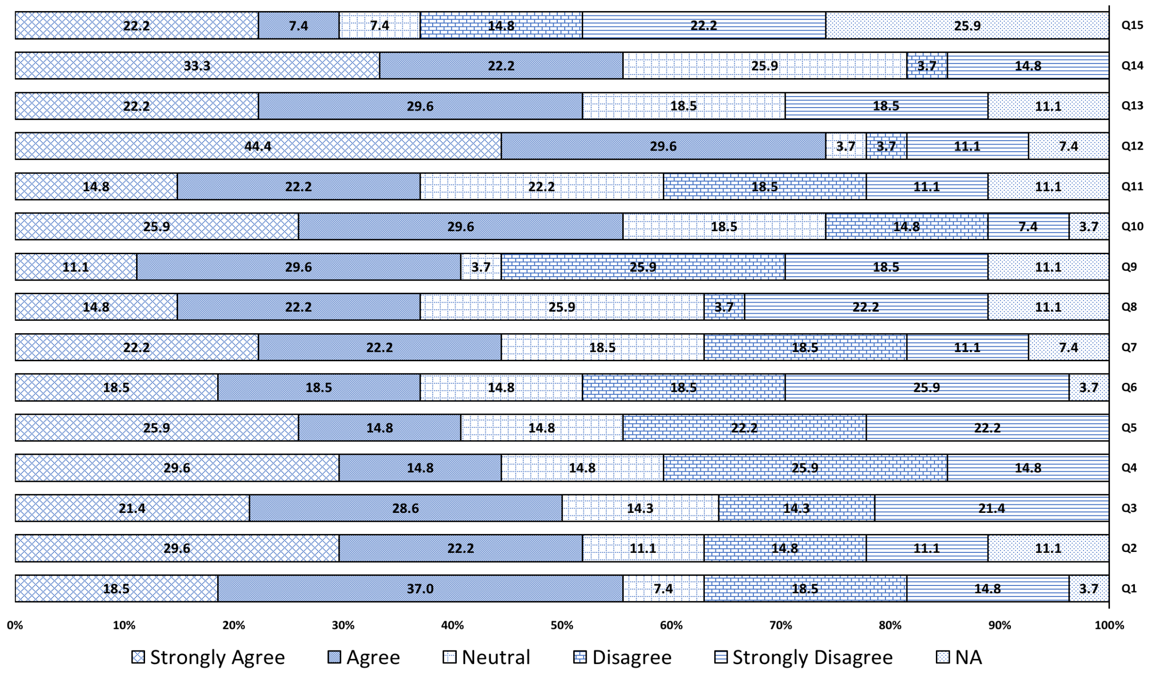


Figure 6.6: Percentages of MACROVR-lite Survey Responses by Likert Value.

An indicator supporting the use of the video component might simply be an outlier in the MACROVR-lite cohort. Looking at the results of survey question 15, as shown in Figure 6.6, we see this cohort gives 25.9% of *NA* choice, the highest over all other questions. We also see they give 14.8% *Disagree* and 22.2% *Strongly Disagree* for the same question. Comparing to the MACROVR cohort for the video component question, the *NA* is 0%, *Disagree* is 8.7%, and *Strongly Disagree* is 4.3%. These responses indicate the individual video component has a stronger connection to students in the MACROVR approach. Another indicator supporting the MACROVR approach versus MACROVR-lite is question 14. The support for GIT/repository in both cohorts show the MACROVR cohort has a combined support (*Strongly Agree* with *Agree*) of 72% versus 55.5% of similar support in the MACROVR-lite cohort. This indicates both cohorts use GIT/repository, but our experimental group, the MACROVR cohort verify it better prepares them for computing job interviews. Our holistic approach, combining all MACROVR components, shows advantage over using only some of them. We normalize all percentages shown in Figures 6.5 and 6.6 relative to the respective cohort's response sizes.

The Wilcoxon rank-sum (Mann-Whitney) test indicates several of our survey questions reject the H_0 hypothesis. In Table 6.5, we see the same questions: 1, 3, 4, 5 ,6, 7, 8, 9, 10, 11, and 14, reject the H_0 hypothesis. This indicates the findings in our survey responses for these questions show significant acceptance of the alternative hypothesis, H_a, and their Likert data substantiate our **RQ2** that the MACROVR approach is reporting as more effective in preparing students for computing job interviews.

Table 6.5: Survey Question Inferential Statistics (Mann-Whitney).

No.	MACROVR Obs	MACROVR-lite Obs	p-value	z score
(1)	85	26	0.0009***	-3.309
(2)	83	24	0.0133*	-2.476
(3)	82	27	0.0070**	-2.695
(4)	84	27	0.0039**	-2.885
(5)	81	27	0.0156*	-2.418
(6) †	82	26	0.0001***	-3.824
(7) †	83	25	0.0038**	-2.891
(8) †	75	24	0.0002***	-3.782
(9) †	74	24	0.0004***	-3.565
(10)	82	26	0.0356*	-2.102
(11) †	74	24	0.0006***	-3.451
(12) †	84	25	0.3305	-0.973
(13) †	76	24	0.0627	-1.862
(14) †	82	27	0.0032**	-2.948
(15) †	73	20	0.1090	-1.603

$*p < 0.05$, $**p < 0.01$, $***p < 0.001$

† MACROVR specific survey question.

6.4.2 Threats to Validity

Although our data collection is sourced from multiple semesters, our research treats them as one group. We acknowledge the validity of new discovery in the MACROVR approach requires maturation and a longitudinal study as is the case in most research. Revisiting this survey in future CS student cohorts, refining the components of MACROVR, without losing the intention and relevance in the computing industry can mitigate this threat. However, our study is to determine if using the MACROVR approach better prepares students for interviews, as compared to a MACROVR-lite approach. Establishing benefits of the MACROVR approach opens investigation into new methods to optimize the method and delivery. Another threat is pre-test and post-test analysis. This research conducts only post-test, as students require exposure to the MACROVR approach in order to form opinion of its efficacy in preparedness in a computing job interview. One mitigation 'pre-test' would involve mock interview questions with specific relations to components of the MACROVR approach, collection of performance data in the course, and inferential analysis.

Extraneous variables such as the details of the computing job focus as well as the practical industry conducting the interview is not in this research. We do not ask specifically in the survey what are the details and duties of the SE job like building cybersecurity components versus web UI components. Also we do not ask from what industry the interview originates like healthcare versus communications versus gaming. Our assumptions that all computing job interviews relate to SE positions in similar ways threaten the external validity or our findings. Mitigation of this area would involve future surveys to include groupings of computing job details and industry as demographic data to be collected and thus, creating an internal threat to longitudinal research with this original data.

Students are known to try and 'game' a system to gain an advantage over others. The team formation algorithms do not have mitigation mechanisms for such adversarial data when classifying the students as 'novice' or 'expert' levels. The ML algorithms assume honesty in answering questions, not deception. A student indicating they do not know an answer, when they in-fact do, can be incorrectly classified as 'novice' and paired with true 'expert' students, giving that team more 'expert' members than others. This can be mitigated with a different set of survey questions with more free-form answering instead of specific answer(s) from multiple choice questions and additional attributes related to off-campus employment or part-time enrollment status.

The study of SE education effectiveness in industry is a vast challenge, but analysis of teaching specifics in the application of the MACROVR approach shows help in minimizing the scope in this challenge. This case study supports preparing CS students for industry by using several techniques and technologies from the MACROVR approach when teaching SE. We see promise in our measures to strengthen the validity of our findings by continuing to refine and compare SE education practice and computing industry readiness. The computing industry and workforce readiness is a moving target, but with identifying threats to validity, we increase awareness of this change and the need to constantly examine, collect, and research SE education.

6.5 Conclusions

In this study we analyze a novel approach to teaching Software Engineering (SE) that integrates numerous researched techniques for industry readiness in SE education. Specifically, we incorporate Agile/Scrum, team formation in project-based learning (PBL), Git version control, and team communication and collaboration. Combining these researched and proven strategies as effective in the computing workforce with our techniques of AI team formation, rotation of team roles, rubrics, and video reporting, we present the MACROVR approach.

We report upon Likert scale data responses from a fifteen question survey and with descriptive and inferential analysis, positively answer our research question in that the MACROVR approach better prepares a CS student for a computing job interview. Our analysis validates this research in eleven out of fifteen survey questions supporting Agile/Scrum, using machine learning to form student teams for PBL, project technologies integrated in the cloud, rotational Scrum roles, rubrics, and student video recordings; -the MACROVR approach. Within the CS student population, we identified two cohorts: 1) students being taught SE topics and concepts requiring use of all components in our MACROVR approach and, 2) students being taught SE utilizing some, but not all components, we call MACROVR-lite. From the total population of n=316, the responses give us an analysis sample size of x=116. From this sample size, we investigate the effect of using real-world SE experiences through CS technical skills and soft skills. Using teaching strategies from the MACROVR approach is improving preparedness in students for computing job interviews.

7 CLASSIFICATION USING DECISION TREES FROM COMPUTER SCIENCE COURSE PERFORMANCE AND SEMESTER SPAN

Using decision trees, we analyze the GSU dataset of CS student records (approximately 49K) with the goal to classify success or failure based upon several CS core curriculum course grades and their semester spans. Decision trees are a non-cyclic connected graph that forms a tree. The tree has internal nodes that correspond to a logical test on some attribute and connecting branches that represent an outcome of the test, usually success or failure. Using this type of structure, the decision tree becomes a classifier for the dataset and selected attributes under investigation.

Our choice of the selected attributes is guided by the CS prerequisite course chart shown in Figure 7.1. We queried our 49K dataset to determine prerequisite courses (cs2720, cs3210, and cs3320) with all *success*, passing grade, as shown in Figure 7.2 and selected CS courses matched as dependent courses. We analyze our extracted student data in a pathway constructed from these prerequisite courses and the respective dependent courses including the semester sequence of all courses. The tree builds with the corresponding dependent course grade and the semester span between the prerequisite course semester sequence and their respective dependent course sequence. We discuss these actions in detail in Section 7.2.

Most decision trees produce a classification of success or failure using Quinlan's Iterative Dichotomiser 3 (ID3) [70]. With ID3, it is implied all values exist for all attributes, i.e., no missing values. Quinlan later created the C4.5 algorithm to compensate for missing values [71]. We are using the R language and its implementation for decision tree classification algorithm is Classification and Regression Trees (CART) [72].

7.1 Background

The main goal of the research in this chapter is to examine GSU's Computer Science Core Curriculum (CSCC) courses, shown in Figure 7.1, in a novel manner and determine factors of success or failure. We determine this based upon demographic data, course performance, course sequencing (prerequisite related to dependency), and semester span derived from the course sequencing. Many studies have looked at prediction of student performance. In 2019, Zhang and Wu analyze student behavior of course achievement in chapter testing [73]. They combine chapter test scores with lab scores in a CS 'C' programming course. Their work compares ID3, C4.5, and CART to form decision trees for prediction of students passing or failing the course. They found the CART algorithm gave the best accuracy in predicting if the student passed or failed the course. Their accuracy is reported as 90% using the CART algorithm.

In another study, Liao et al., uses clicker data gathered in three CS courses applied with Peer Instruction pedagogy. Their goal involves predicting low-performing students in beginning CS courses [74]. Similar to our study using lower level CS prerequisite courses, they find using support vector machine algorithms to mine the clicker data gives a 62% accuracy in predicting a failing student. Others like Dhanpal et al., apply graph clustering algorithms to predict student performance based upon recent GPA values over a three-year period [75].

Our work in this area is to determine accuracy of our model using a CS student's GPA values from CS courses along with their behavior in course sequencing with predicting success of failure of these CS courses. The student behavior related to selecting a CS course at a specific time can be influenced by things outside of their control such as availability of a CS course when needed, personal events that require reducing course loads, and the effectiveness of the learning they receive in a classroom. These attributes are difficult if not impossible to capture in a dataset for analysis. However, by using the semester span that is influenced by these and depends upon sequencing of a student taking a CS course, we see potential to realize new knowledge related to student behavior aiming to improve classification accuracy when predicting if they finish a program successfully or not.

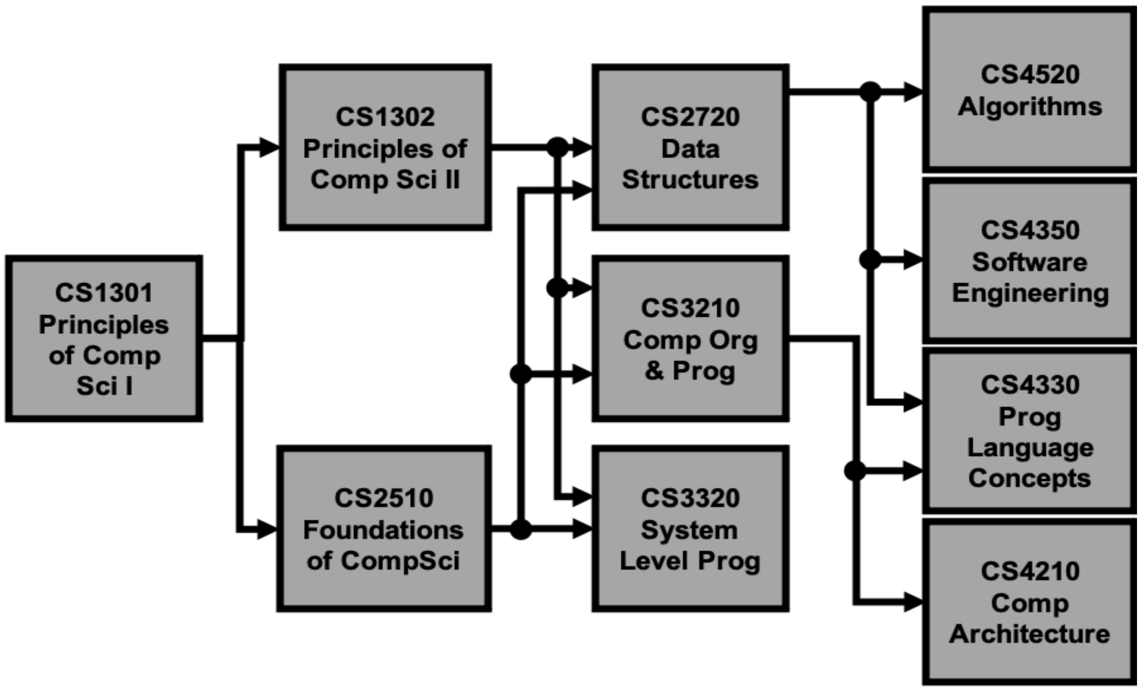

CS1301 Principles of Comp Sci I
CS1302 Principles of Comp Sci II
CS2510 Foundations of CompSci
CS2720 Data Structures
CS3210 Comp Org & Prog
CS3320 System Level Prog
CS4520 Algorithms
CS4350 Software Engineering
CS4330 Prog Language Concepts
CS4210 Comp Architecture

Figure 7.1: Computer Science Core Curriculum (fall 2019).

7.2 Methods

Our applied heuristic to determine *success* or *failure* selected from attributes of the dataset is done using supervised learning where we consider the intersection of two criteria. The first is a selection status of *success* when a student passes with a grade of ‘C’ or higher in three prerequisites CS courses: -data structures (cs2720), computer organization and programming (cs3210), and system-level programming (cs3320). The second selection criteria is the student must complete (without withdrawal) all four dependent CSCC required courses that includes algorithms (cs4520), software engineering (cs4350), programming language concepts (cs4330), and computer architecture (cs4210). To summarize, if a student passes the three prerequisites (cs2720, cs3210, cs3320) and complete the four dependent required courses, they are extracted. We establish the ground truth data from this extracted set by classifying as *success* if the student passes the three prerequisites and all four dependents and classify as *failure* if the student passes the three prerequisites and fails one of the four dependents.

We compute the semester span by using Figure 7.1 as a guide. This heuristic utilizes the prerequisite course of cs2720 and its dependent courses cs4520, cs4350, and cs4330. We select the prerequisite course cs3210 because of the dependent course cs4210. We chose to keep cs3320 as a selection criteria because the course does have content related to other four-thousand level CS courses in our dataset and is a required course. We feel the topics in cs3320 influence courses beyond the four, four-thousand level courses.

Computing the semester span is different for cs4210 from cs4520, cs4350, and cs4330. The cs4210 course computes the semester span with the prerequisite semester sequence from cs3210. The cs4520, cs4350, and cs4330 all use the prerequisite semester sequence from cs2720 in their computation for semester span. The computation is how many semesters transpire from completing the prerequisite course and completing the dependent course. Again, we capture the 'completed' semester sequence regardless if the student passed or failed the course because this value indicates student behavior and dependency in our decision tree. GSU offers a 'repeat and replace' for courses and this can influence the semester sequence values. We capture an attribute in our dataset we call the 'DFWFsequence' that indicates the semester sequence the failure for a CS course occurs. Additionally, we capture the number of failures for all CS courses taken, the withdrawal semester sequence, and frequency of a CS course withdrawal. Because there are a very small number of observations with these attributes, we do not consider them in computing our decision tree. In our future work, we discuss generative modeling using many attributes of ground truth data and these are considered descriptively valuable enough to be included in the modeling.

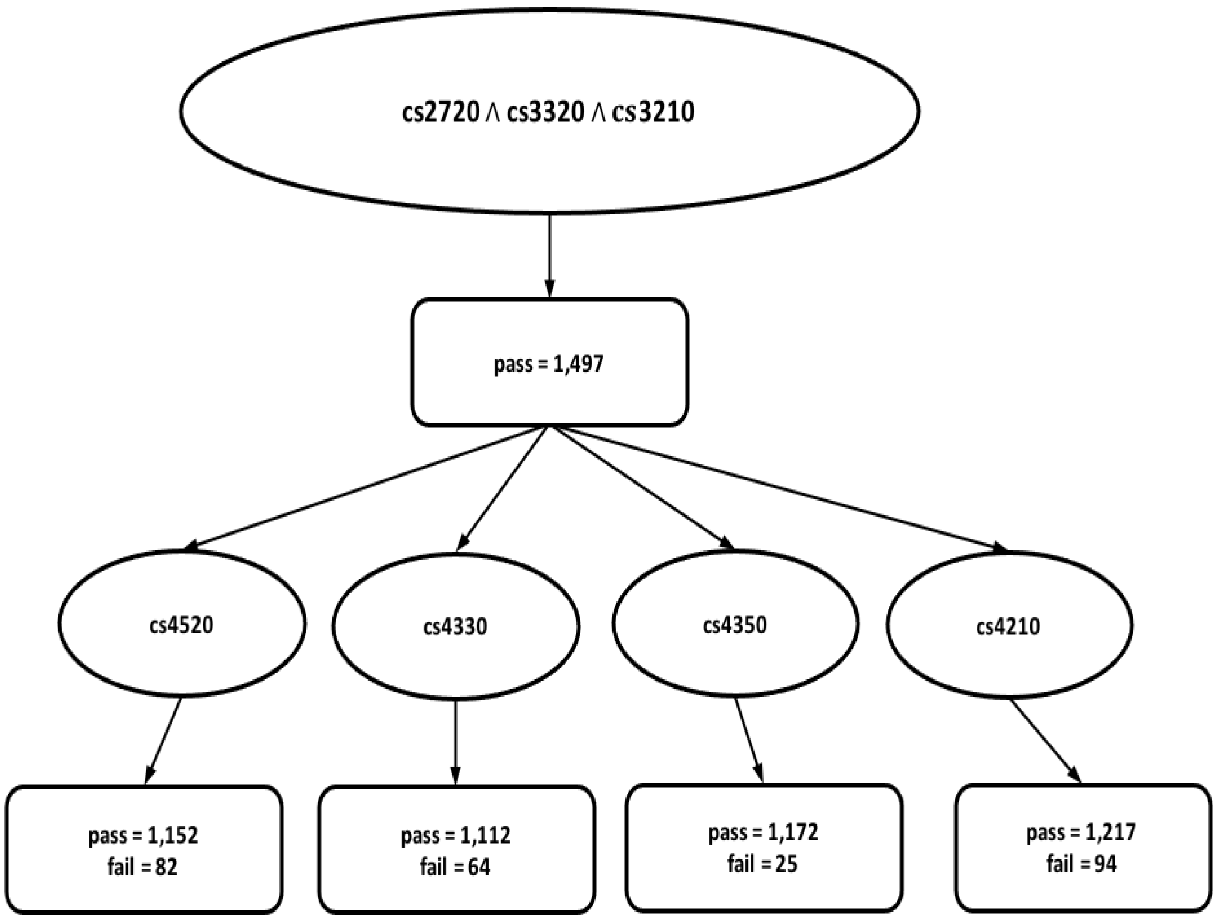


Figure 7.2: Sample Sizes in Attribute Selection from Dataset for Decision Tree Computations.

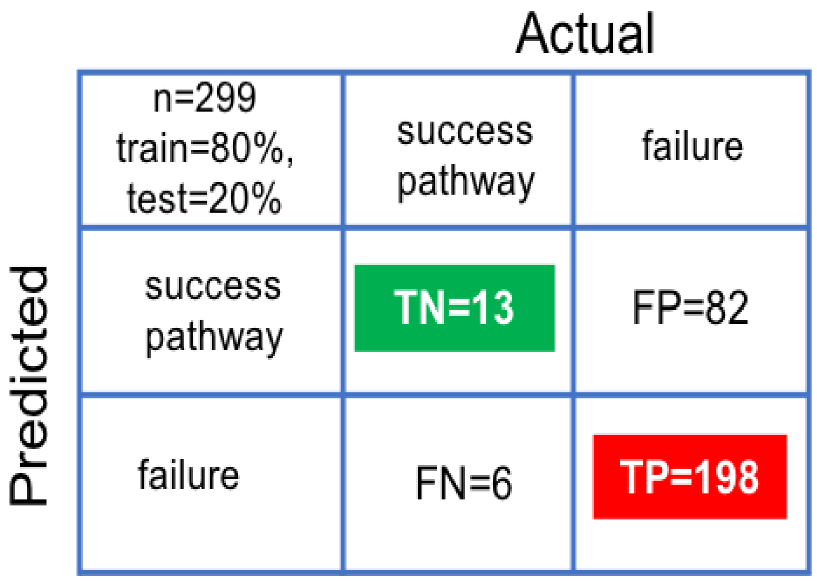
Several criterion are used to refine the calculations for attribute selection. These calculations are based on information gain for the ID3 algorithm, gain ratio for the C4.5 algorithm, and the Gini Index if we use the CART algorithm. Also, gain ratio is used to overcome some of the bias introduced when partitioning the data [76]. The R decision tree algorithm uses CART and relies on the Gini Index value. With our sample size $\boldsymbol{S}$ having $\boldsymbol{k}$ observations, each has $\boldsymbol{n}$ decision attributes. Each p_j represents a relative probability of the decision attribute's value $\boldsymbol{j}$ in sample $\boldsymbol{S}$. Where $\boldsymbol{C}$ is the number of classes in the tree branch indicated by our binomial classifier (success or failure). If we divide the sample $\boldsymbol{S}$ into two datasets, S_1 and S_2, we would have our Gini Index formula:

$$Gini_{split}(S) = \frac{k_1}{k} * Gini(S_1) + \frac{k_2}{k} * Gini(S_2)$$

The more generalized formulation is:

$$Gini = 1 - \sum_{j=1}^{C} (p_j)^2$$

Predictions in our model to classify the *success* or *failure* outcome can be visualized with a confusion matrix. Figure 7.3 shows the result or our model's classification. A *true positive* is where the model correctly predicted *Failure*=198. The *false positive* is where the model incorrectly predicted *Failure*=82. The *false negative* is where the model incorrectly predicted *Success*=6. The *true negative* is where the model correctly predicted *Success*=13.



		Actual	
	n=299 train=80%, test=20%	success pathway	failure
Predicted	success pathway	TN=13	FP=82
	failure	FN=6	TP=198

Figure 7.3: Our Confusion Matrix with Split Ratio of 80%.

The accuracy formula derived from a confusion matrix is:

$$Accuracy = \frac{TP + TN}{n}$$

The accuracy in our confusion matrix is computed as:

$$\frac{198 + 13}{299} = 70.57\%$$

We discuss the output of the confusion matrix related to our model in section 7.3. We cross-validate our classification by using the holdout method with a 20/80 split. We experimented with varying holdout percentages where training data was modeled with 70%, 75%, and 80%, but the prediction accuracy is optimal at the 25/75 split. Our algorithm for determining an optimal tree for knowledge discovery is shown in Algorithm 1.

Algorithm 1 Decision Tree Pruning

```
Input: Decision trees $D = \langle T_1, T_2, T_3 \rangle$
Output: Pruned decision tree
// $T_1$ has SR 70%, $T_2$ has SR 75%, $T_3$ has SR 80%
1  Build and observe all tree's accuracy and Kappa, select best
2  Plot the best tree from set $D$
3  Examine root node logical test
   if $Step3 = sensible\ AND\ root\_node\_frequency \leq 2$ then
4      Select tree from $Step3$
5      go to $Step11$
6  else
7      Pre-prune root node
8      go to $Step2$
9  end
11 Perform Complexity Parameter (CP) operation, obtain best cross-validation value
12 Rebuild classification tree selected in Step3 using the CP value (post-pruning)
13 Plot the new tree
14 Compare analysis between two trees (Step2, Step13)
15 Select best based on size, number of binomial paths, sensibility
```

7.3 Results

We begin with showing a subset of output results generated from the R confusion matrix:

```
Confusion Matrix and Statistics
          Reference
Prediction failure success
   failure     198      82
   success       6      13
               Accuracy : 0.7057
                  Kappa : 0.1366
       'Positive' Class : failure
```

Interpretations for the Kappa[1] value is the indication of *rating agreement* between the supervised learning classification and prediction model's classification. The closer the Kappa value is to 100%, indicates a higher agreement between the prediction and reference data. Even though our model produces a 70.57% accuracy, the Kappa value of 13.66% is low. One can explain this because of the heuristic to determine supervised learning observations in our dataset resulted in success=981, and failure=516, shown in Table 7.1. The confusion matrix in our prediction model is success=13, and failure=198, showing a large variation from the supervised.

Perhaps our model would be better to use a 'quality' indicator based upon another heuristic such as class size ratio of students to faculty instructor and if taught by a faculty or graduate student. Intuition tells us using these attributes could impact prediction models for a student's success or failure. Currently, it is not possible to capture these attributes, preprocess them, and analyze in a decision tree model.

The dataset's descriptive statistics are shown in Table 7.1. It indicates that each of our chosen attributes related to student performance results in a 'C+' grade or higher. The *span*

[1]http://john-uebersax.com/stat/kappa.htm (Accessed in October 2020).

columns indicate values representing the semester from completing a prerequisite CS course and the semester of completion in the respective dependent CS course. It is interesting to see that even though cs4520, cs4350, and cs4330 share the same prerequisite course, the software engineering course (cs4350) appears from the *Mean* value to take the least time for students to complete. This dataset is from years 2008 through 2018, and the newly activated requirement of taking cs4520 prior to cs4350 might explain why their span's *Mean* values are similar compared to the cs4330 span value. The negative values for *span* could be indicating students were granted permission from the university to take the prerequisite course simultaneously with the dependent course, but failed the prerequisite course and repeated, thus replacing the semester sequence with a later occurrence or a repeat to replace a 'C' grade.

Table 7.1: Descriptive Statistics.

Stat.	**cs4520 grade**	**cs4350 grade**	**cs4330 grade**	**cs4210 grade**	**cs4520 span**	**cs4350 span**	**cs4330 span**	**cs4210 span**
Min.	0.0	0.0	0.0	0.0	-2.0	-2.0	-11.0	-6.0
1st Q	2.7	3.0	2.3	2.3	1.0	1.0	2.0	1.0
Med.	3.3	3.0	2.7	3.0	3.0	3.0	3.0	2.0
Mean	3.149	3.305	2.745	3.003	2.997	2.764	3.244	2.34
3rd Q	4.0	4.0	3.3	4.0	4.0	3.0	4.0	3.0
Max.	4.3	4.3	4.3	4.3	20.0	20.0	23.0	15
NA's	323	315	321	254	323	315	321	280

Success=981, Failure=516 (Supervised Learning).

We present three models of our decision trees visualized in Figures 7.4, 7.5, and 7.6. Using Algorithm 1, each decision tree, incorporating all attributes, derives the same tree size of four. Observing the terminal nodes marked as *success*, we see Model 2, in Figure 7.5, offers the highest success rate of 30% and the highest Kappa value of 23.95%. Next, we take Model 2 and apply the R algorithm to compute Complexity Parameter (CP) to find an optimal value closest to the cross-validation level as shown in Figure 7.7. Applying this to our selected decision tree, it produces a very similar tree with little significant difference as shown in Figure 7.8. We observe the Kappa value decreases and the accuracy decreases. Referring back to our Algorithm 1, we perform a pre-pruning of the cs4330 grade and regenerate the decision tree.

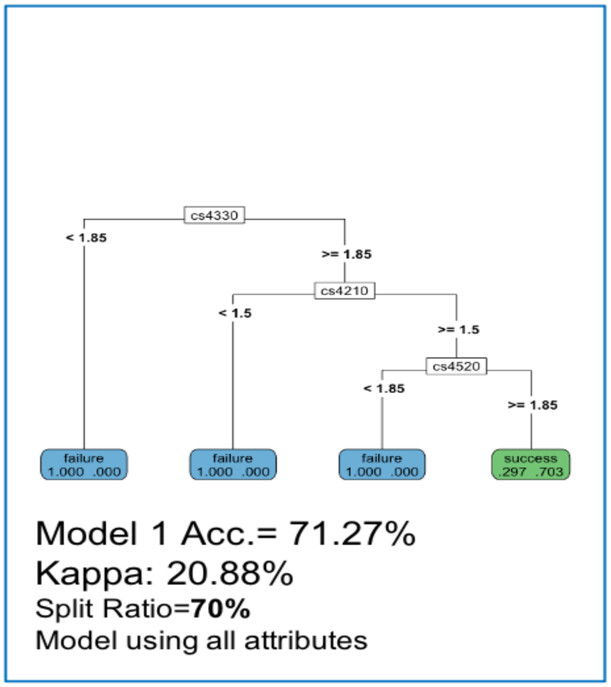


Figure 7.4: Decision Tree All Attributes Split Ratio = 70%.

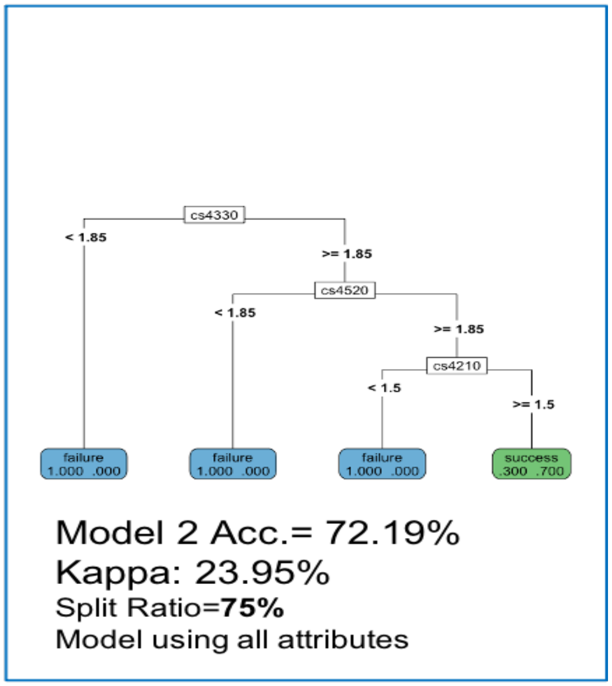


Figure 7.5: Decision Tree All Attributes Split Ratio = 75%.

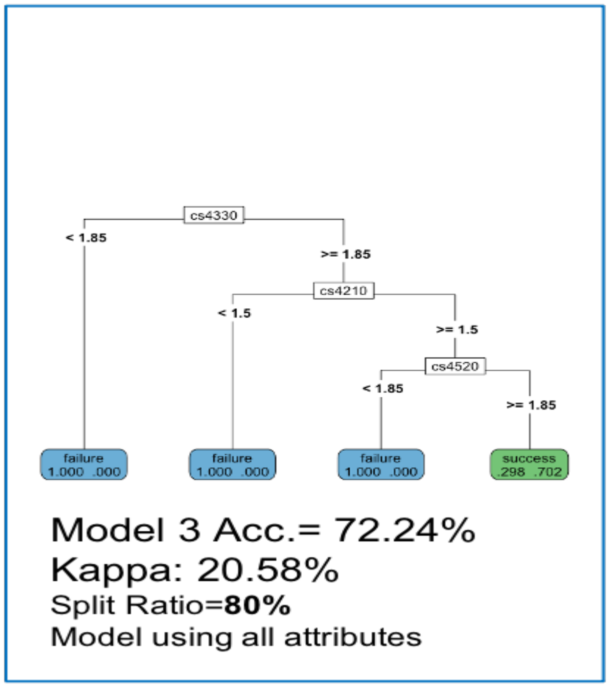


Figure 7.6: Decision Tree All Attributes Split Ratio = 80%.

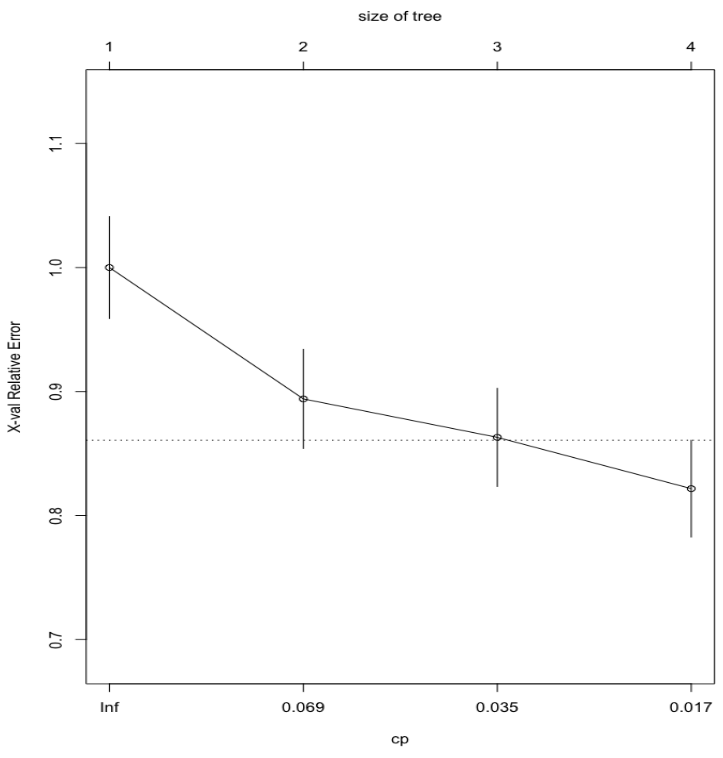


Figure 7.7: Complexity Parameter Computation with Model 2, Shown in Figure 7.5.

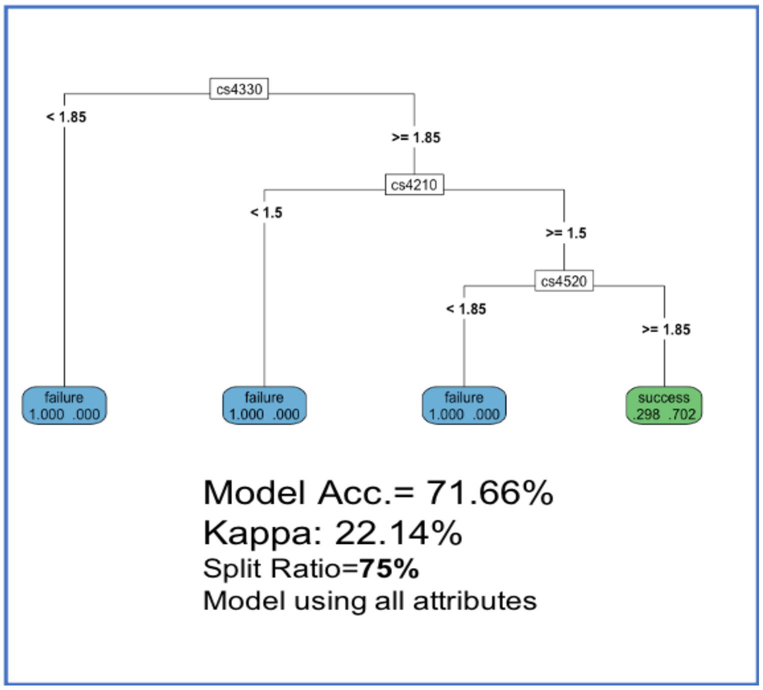


Figure 7.8: New Decision Tree of Model 2, after CP Algorithm Applied.

Our new decision tree, shown in Figure 7.9, displays the larger detailed tree after we perform a pre-pruning of the cs4330 grade. We remove this attribute because it appears in all three models and the newly generated model after a CP is applied to the Model 2. Our Algorithm 1 states if it is 'sensible' or has 'sensibility' to make a decision to keep a node logical test in the decision tree or re-build without that attribute. We delete the cs4330 grade because of its relatively small value all the binomial splits at the root node in each model we build and present.

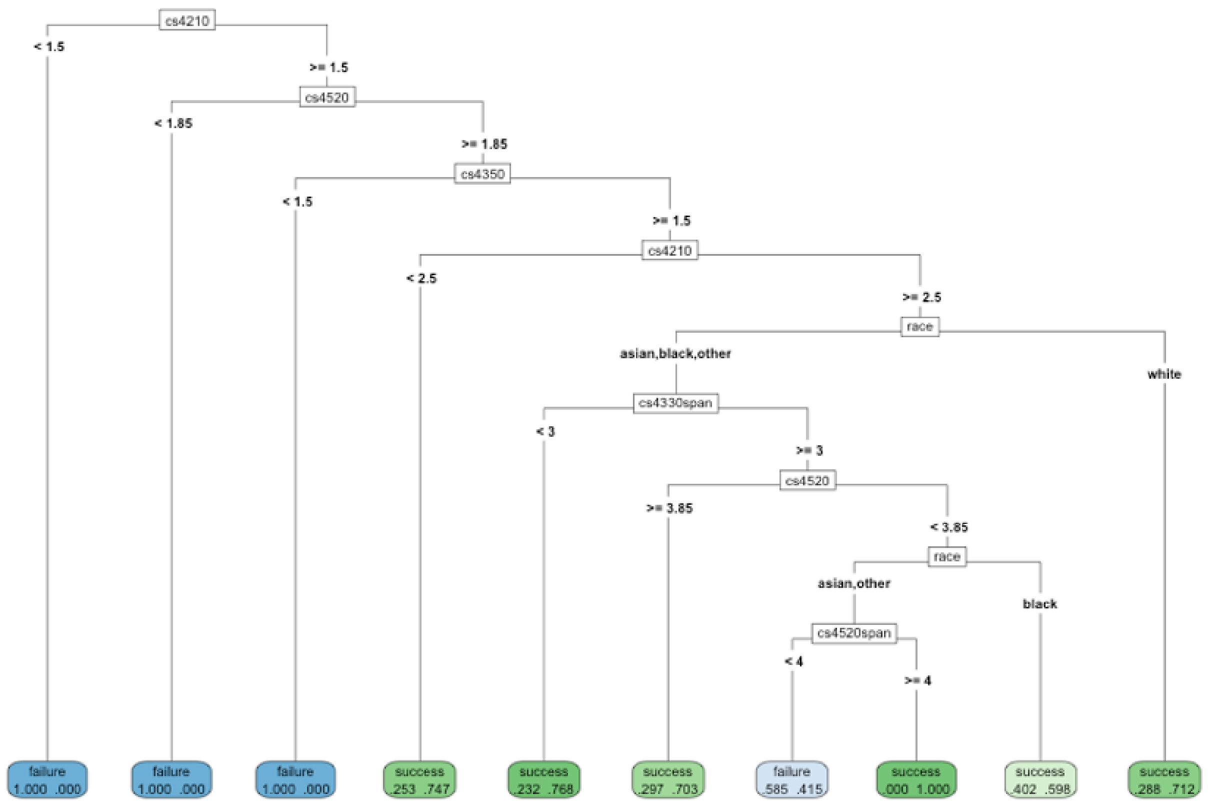


Figure 7.9: New Decision Tree of Model 2, with Pre-pruning 'cs4330' Attribute.

The new decision tree in Figure 7.9 shows a new high success prediction probability of 40.2% in the second from the right terminal node. Observing this discovery, we can formulate a more readable flowchart of the entire tree shown in Figure 7.10. Interpretation of our decision tree with a flowchart representation gives us new knowledge for describing *success* pathways for the CS student. The root node and others that show a very low GPA (1.5, 1.85) as their binomial choice indicate this attribute's information gain is the highest when split on this low GPA value. One can interpret as a large number of different grade values with the best bifurcation of occurrence being at this low value. Perhaps a better grade value is to create an ordinal range to normalize the large variety of grade values. This new understanding showing pathways can be described in the following narrative:

1. Computer Architecture grade is greater than or equal to 1.5

2. Algorithms grade is greater than or equal to 1.85
3. Software Engineering grade is greater than or equal to 1.5
4. Computer Architecture grade is greater than or equal to 2.5
5. Race is Black, Asian , or other
6. Programming Language Concepts span is greater than or equal to 3
7. Algorithms grade is greater than or equal to 3.85
8. Race is Black (*Success* classification with prediction of 40.2%)

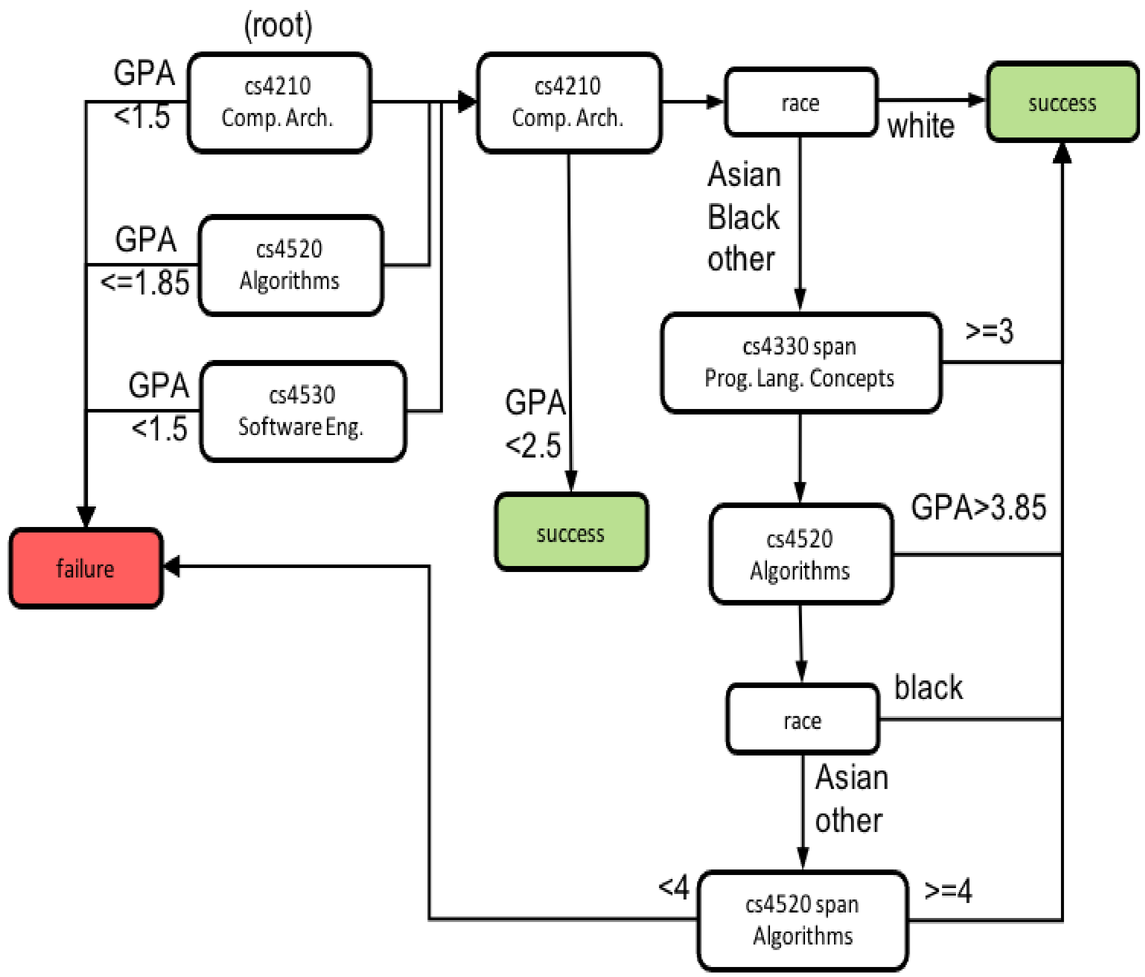


Figure 7.10: Flowchart of New Decision Tree from Figure 7.9.

Using the aforementioned heuristic, we can use our new understanding and create actionable recommendations by making changes within our decision tree and reanalyzing. Performing a repeat of our Algorithm 1, and heuristic, we find more ways to increase the pathways of *success* for our CS students.

7.4 Conclusions

In this study, we have analyzed CS student data to build a decision tree classification model. We looked at numerous attributes in our dataset and determined to focus on the Computer Science Core Curriculum (CSCC) courses in our program. The dataset was analyzed and we were able to extract a representative sample of 1,497 student records that contained demographic attributes, GPA of CS courses, and a semester sequence that occurred in our time span of 10 years.

Analyzing these attributes, we use them in the R programming language with statistical libraries to analyze and compute classifications using the CART algorithm for decision trees. This analysis gives us varying degrees of prediction accuracy and our final best accuracy achievement is 72.24%. The supervised learning data reveals interesting pathways for semester sequencing relating to the demographic subgroups of students. Additionally, producing knowledge from these decision trees reveals optimal pathways for *success* in completing the CSCC courses. With this understanding of our CS student population, we can design, implement, and test intervention activities. These can relate to reducing the semester span between a CS prerequisite course and its dependent course as well as supporting programs for new CS students that are transferring from other universities and colleges. Building on this work, additional student attributes can be gathered, analyzed, and interpreted for new understanding and realization of successful pathways for the CS student population and applied in other academic populations.

8 CONCLUSIONS

In this dissertation, we study the challenges facing the undergraduate CS student's pathway to graduation at Georgia State University (GSU). These express themselves through data attributes in our the university's data system and surveys that measure teaching strategies in course delivery and outcomes. Investigating these planes of data realization with data mining, statistical regression, and classification models, we discover new knowledge where known inequities exist, intuitively, but are not always quantifiable. Multiple research works comprise this dissertation with foci on a global problem of meeting a growing workforce demand for computer science Baccalaureates.

Firstly, this dissertation addresses realistic problems of inequities with transfer students and prerequisite courses being taught in departments other than CS. We show transfer students at GSU take heavier loads of CS courses and suffer with lower GPA and more frequent failures. Additionally, we find a difference in the outcome performance of a CS algorithms course that is dependent upon a prerequisite course that is accepted from the mathematics department or CS department at GSU. Looking into the analysis, we find the students taking the prerequisite course not in the CS department have lower performance in the algorithms course and a longer semester span before successfully completing the CS course.

Secondly, this dissertation studies the effect a novel teaching strategy has upon CS student's seeking a computing industry job. Developing teaching strategies, utilizing technology platforms in the classroom, and employing machine learning algorithms gives us the ability to measure effect and outcome. We create the MACROVR approach that shows students taking software engineering under this teaching strategy and approach have greater success in computing industry interviews. Our research supports the GSU initiative[1] of embedding a 'college to career' mechanism into courses for preparing CS students entering the practice reality in the workforce.

Thirdly, this dissertation considers a study involving an early CS1 course at GSU related to a student's interpretation of a graphical chart. Our study's purpose is to test the hypothesis that if a beginning CS student can successfully answer questions relating to our prerequisite chart found in Figure 8.1, their performance in the CS1 course will be higher. Our study, approved by IRB[2] delivered a paper survey to students in the first two weeks of the CS1 course. We collected their course performance at the end of the semester and ran several regression models with the demographics, survey score, and CS1 course GPA. The analysis results did not give correlations related to our hypothesis. Since no correlations could be found, the research was stopped.

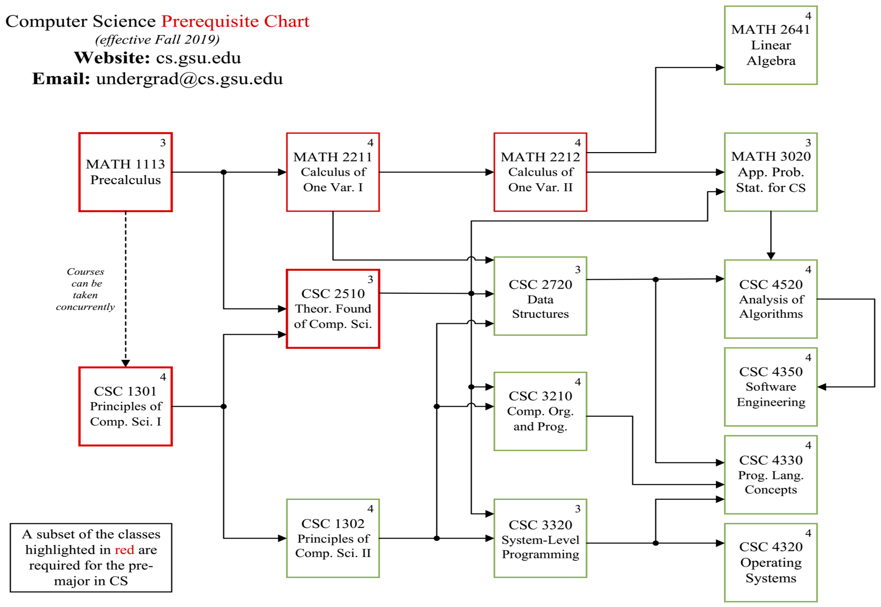

Computer Science Prerequisite Chart
(effective Fall 2019)
Website: cs.gsu.edu
Email: undergrad@cs.gsu.edu
3 MATH 1113 Precalculus
4 MATH 2211 Calculus of One Var. I
4 MATH 2212 Calculus of One Var. II
4 MATH 2641 Linear Algebra
3 MATH 3020 App. Prob. Stat. for CS
3 CSC 2510 Theor. Found of Comp. Sci.
3 CSC 2720 Data Structures
4 CSC 4520 Analysis of Algorithms
4 CSC 4350 Software Engineering
Courses can be taken concurrently
4 CSC 1301 Principles of Comp. Sci. I
4 CSC 3210 Comp. Org. and Prog.
4 CSC 4330 Prog. Lang. Concepts
4 CSC 1302 Principles of Comp. Sci. II
3 CSC 3320 System-Level Programming
4 CSC 4320 Operating Systems
A subset of the classes highlighted in red are required for the pre-major in CS

Figure 8.1: CS Prerequisite Chart for CS1 Course Study.

[1] https://success.gsu.edu/initiatives/college-to-careers (Accessed in November 2020).
[2] IRB: H20340

This study can be revisited with a different survey instrument and perform pretest and post-test surveys. Using a more qualitative approach, may yield valuable knowledge about the first year CS student. Also tracking their progress with follow-up surveys or course assessments in additional lower level CS courses would contribute to a predictive model for optimal pathway and decrease time-to-graduation.

Lastly, this dissertation shines the light of classification into our CS dataset. We wish to understand the bottlenecks of our CS student's pathway to success by analyzing CS course performance and sequencing of the required courses. Pathway analysis using data mining and machine learning in CS datasets is not that easily found in research. Our first step with building a classification decision trees based upon this data has established the beginning structure related to analysis in our dataset. In order to understand the CS student journey at GSU and find areas where improvements can be given and tested for effect, we need a much larger dataset representing undergraduate CS students.

Garnering a very large dataset of CS student data is a daunting task and most student research relies on small, specific datasets or synthetically created datasets. In order to use big data mining techniques, one must posses a very large dataset that is robust and representative of the characteristics in the population being studied. A new area of established research known as a Generative Adversarial Network (GAN) emerges to use generative models and neural networks combined that can create a very large plausible, synthetic dataset based upon a ground-truth dataset. Recent works in research have been found and prove to deliver very large generated datasets of in numerous formats like tabular and relational [77, 78]. Other research has addressed the data security and privacy concerns with synthetic data from generative modeling [79, 80], thus opening new research that builds upon the work in this dissertation.

It is our hope this new area of research will open avenues in data analysis of CS students and give us the ability to create and establish predictive models to be used for finding optimized pathways for our CS students and students in other academic programs. The possibilities are hopeful to use these new techniques, strategies, and algorithms to build a very large, publicly available dataset for continued research to improve pathways of success for our students.

REFERENCES

[1] C. R. Glass and C. M. Westmont, "Comparative effects of belongingness on the academic success and cross-cultural interactions of domestic and international students," *International Journal of Intercultural Relations*, vol. 38, no. 1, pp. 106–119, 2014.

[2] C. c. Yang and B. B. Brown, "Motives for Using Facebook, Patterns of Facebook Activities, and Late Adolescents' Social Adjustment to College," *Journal of Youth and Adolescence*, vol. 42, no. 3, pp. 403–416, 2013.

[3] A. D. Tiwari and A. Misal, "College Student Work Habits, Interuptions, and Stress," vol. 2, no. 2, pp. 23–29, 2008.

[4] Flores, I. del Arco, P. Silva, L. Arthur, E. Cox, and G. Siemens, "The flipped classroom model at the university: analysis based on professors' and students' assessment in the educational field," *International Journal of Educational Technology in Higher Education*, vol. 13, no. 1, p. 21, 2016.

[5] P. G. de Barba, G. E. Kennedy, and M. D. Ainley, "The role of students' motivation and participation in predicting performance in a MOOC," *Journal of Computer Assisted Learning*, vol. 32, no. 3, pp. 218–231, 2016.

[6] W. Young, L. Allen, and K. Warfield, "Developing Online / Hybrid Learning Models for Higher Education Programs," *Alabama Journal of Educational Leadershi*, vol. 3, pp. 47–56, 2016.

[7] H. Danielsiek, "Stay on These Roads : Potential Factors Indicating Students ' Performance in a CS2 Course," *Proceedings of the 47th ACM Technical Symposium on Computer Science Education (SIGCSE '16)*, pp. 12–17, 2016.

[8] A. Elbadrawy, R. S. Studham, and G. Karypis, "Collaborative Multi-Regression Models for Predicting Students' Performance in Course Activities," *Proceedings of the Fifth*

International Conference on Learning Analytics And Knowledge - LAK '15, pp. 103–107, 2015.

[9] S. Shehata and K. E. Arnold, "Measuring Student Success Using Predictive Engine," *Proceedings of the Fifth International Conference on Learning Analytics And Knowledge*, pp. 416–417, 2015.

[10] B. McMurtrie, "The future of learning: How colleges can transform the educational experience.," *Chronicle of Higher Education*, 2018.

[11] Best College Reviews, "Top 50 Ethnically Diverse Colleges in America," 2018.

[12] C. Romero and S. Ventura, "Educational data mining: a review of the state of the art," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 6, pp. 601–618, 2010.

[13] R. Asif, A. Merceron, S. A. Ali, and N. G. Haider, "Analyzing undergraduate students' performance using educational data mining," *Computers & Education*, vol. 113, pp. 177–194, 2017.

[14] N. Ketui, W. Wisomka, and K. Homjun, "Using Classification Data Mining Techniques for Students Performance Prediction," in *2019 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT-NCON)*, pp. 359–363, IEEE.

[15] A. P. Patil, K. Ganesan, and A. Kanavalli, "Effective deep learning model to predict student grade point averages," in *2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, pp. 1–6, IEEE, 2017.

[16] I. Burman and S. Som, "Predicting students academic performance using support vector machine," in *2019 Amity International Conference on Artificial Intelligence (AICAI)*, pp. 756–759, IEEE, 2019.

[17] Z. Iqbal, A. Qayyum, S. Latif, and J. Qadir, “Early Student Grade Prediction: An Empirical Study,” in *2019 2nd International Conference on Advancements in Computational Sciences (ICACS)*, pp. 1–7, IEEE, 2019.

[18] J.-L. Hung, B. E. Shelton, J. Yang, and X. Du, “Improving predictive modeling for at-Risk student identification: a multistage approach,” *IEEE Transactions on Learning Technologies*, vol. 12, no. 2, pp. 148–157, 2019.

[19] W. W. T. Fok, Y. s. He, H. H. A. Yeung, K. Y. Law, K. H. Cheung, Y. Y. Ai, and P. Ho, “Prediction model for students’ future development by deep learning and tensorflow artificial intelligence engine,” in *2018 4th International Conference on Information Management (ICIM)*, pp. 103–106, IEEE, 2018.

[20] various, “Campus Ethnic Diversity,” tech. rep., U.S. News and World Report, Washington D.C., 2018.

[21] S. S. Ditchkoff, D. N. Laband, and K. Hanby, “Academic performance of transfer versus ”native” students in a wildlife Bachelor of Science program,” *Wildlife Society Bulletin*, vol. 31, no. 4, pp. 1021–1026, 2003.

[22] M. E. Vito, “American Journal of Educational Studies, Vol.6, No.1, 2013,” vol. 6, no. 1, pp. 47–65, 2013.

[23] M. D. Johnson, “Academic Performance of Transfer Versus ’Native’ Students in Natural Resources & Sciences,” *College Student Journal*, vol. 39, pp. 570–579, 9 2005.

[24] J. R. Colley and A. G. Volkan, “Evaluating the quality of transfer versus nontransfer accounting principles grades.,” *Journal of Education for Business*, vol. 71, p. 359, 7 1996.

[25] E. Bacon and L. MacKinnon, “Computer science graduates: why do they top unemployment tables?,” *The Guardian (online)*, p. 1, 7 2017.

[26] D. Franklin, “Putting the computer science in computing education research,” *Communications of the ACM*, vol. 58, no. 2, pp. 34–36, 2015.

[27] ACM/IEEE-CS Joint Task Force on Computing Curricula, “Computer Science Curricula 2013,” tech. rep., ACM Press and IEEE Computer Society Press, 12 2013.

[28] H. Danielsiek, W. Paul, and J. Vahrenhold, “Detecting and understanding students’ misconceptions related to algorithms and data structures,” in *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, pp. 21–26, ACM, 2012.

[29] P. R. Hernandez, A. Woodcock, M. Estrada, and P. W. Schultz, “Undergraduate Research Experiences Broaden Diversity in the Scientific Workforce,” *BioScience*, 2018.

[30] P. Rheingans, E. D’Eramo, C. Diaz-Espinoza, and D. Ireland, “A Model for Increasing Gender Diversity in Technology,” 2018.

[31] G. C. Townsend and K. Sloan, “Julian Scholars: Broadening Participation of Low-Income, First-Generation Computer Science Majors,” *Computing in Science and Engineering*, 2016.

[32] M. Babes-Vroman, I. Juniewicz, B. Lucarelli, N. Fox, T. Nguyen, A. Tjang, G. Haldeman, A. Mehta, and R. Chokshi, “Exploring Gender Diversity in CS at a Large Public R1 Research University,” 2017.

[33] M. Kordaki and I. Berdousis, “Course Selection in Computer Science: Gender Differences,” *Procedia - Social and Behavioral Sciences*, 2014.

[34] L. J. Sax, K. J. Lehman, J. A. Jacobs, M. A. Kanny, G. Lim, L. Monje-Paulson, and H. B. Zimmerman, “Anatomy of an Enduring Gender Gap: The Evolution of Women’s Participation in Computer Science,” *Journal of Higher Education*, 2017.

[35] D. Michell, A. Szorenyi, K. Falkner, and C. Szabo, “Broadening participation not border protection: how universities can support women in computer science,” *Journal of Higher Education Policy and Management*, 2017.

[36] S. West, M. Whittaker, and K. Crawford, "Discriminating Systems: Gender, Race, and Powerin AI," tech. rep., AI Now Institute, 2019.

[37] S. L. Rodriguez and K. Lehman, "Developing the next generation of diverse computer scientists: the need for enhanced, intersectional computing identity theory," *Computer Science Education*, 2017.

[38] G. Y. Lin, "Self-efficacy beliefs and their sources in undergraduate computing disciplines: An examination of gender and persistence," 2016.

[39] A. Hellas, P. Ihantola, A. Petersen, V. V. Ajanovski, M. Gutica, T. Hynninen, A. Knutas, J. Leinonen, C. Messom, S. N. Liao, K. D. Kolo, S. A. Adepoju, and J. K. Alhassan, "Predicting academic performance: a systematic literature review," *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, vol. 5, no. 5, pp. 175–199, 2015.

[40] N. Newton, H; Cox, "Stata Journal," *Stata Journal*, vol. 19, no. 2, 2019.

[41] A. C. Cameron and P. K. Trivedi, *Regression Analysis of Count Data.* No. no. 30 in Econometric Society Monographs, Cambridge University Press, 1998.

[42] A. F. Hayes and L. Cai, "Using heteroskedasticity-consistent standard error estimators in OLS regression: An introduction and software implementation," *Behavior research methods*, vol. 39, no. 4, pp. 709–722, 2007.

[43] P. Kabaila, S. Alhelli, D. Farchione, and N. Bragg, "The effect of a Durbin-Watson pretest on confidence intervals in regression," *arXiv preprint arXiv:1804.04306*, 2018.

[44] S. Morsy and G. Karypis, "A Study on Curriculum Planning and Its Relationship with Graduation GPA and Time To Degree," in *Proceedings of the 9th International Conference on Learning Analytics & Knowledge*, LAK19, (New York, NY, USA), pp. 26–35, ACM, 2019.

[45] A. Elbadrawy, A. Polyzou, Z. Ren, M. Sweeney, G. Karypis, and H. Rangwala, "Predicting Student Performance Using Personalized Analytics.," *Computer*, no. 4, p. 61, 2016.

[46] C. Stephenson, A. Derbenwick-Miller, C. Alvarado, L. Barker, V. Barr, T. Camp, C. Frieze, C. Lewis, E. Cannon-Mindell, L. Limbird, D. Richardson, M. Sahami, E. Villa, H. Walker, and S. Zweben, "Retention in Computer Science Undergraduate Programs in the U.S.: Data Challenges and Promising Interventions," tech. rep., ACM, New York, NY, USA, 2018.

[47] A. Derbenwick Miller, C. Alvarado, M. Sahami, E. Villa, and S. Zweben, "Wrestling with Retention in the CS Major: Report from the ACM Retention Committee," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, SIGCSE '19, (New York, NY, USA), pp. 807–808, ACM, 2019.

[48] H. M. Walker, "Retention of Students in Introductory Computing Courses: Preliminary plans—ACM Retention Committee," *ACM Inroads*, vol. 8, p. 12, 10 2017.

[49] D. Ford, T. Barik, L. Rand-Pickett, and C. Parnin, "The tech-talk balance: what technical interviewers expect from technical candidates," in *2017 IEEE/ACM 10th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pp. 43–48, IEEE, 2017.

[50] D. Dzvonyar, L. Alperowitz, D. Henze, and B. Bruegge, "Team composition in software engineering project courses," in *2018 IEEE/ACM International Workshop on Software Engineering Education for Millennials (SEEM)*, pp. 16–23, IEEE, 2018.

[51] C. Anslow and F. Maurer, "An experience report at teaching a group based agile software development project course," in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pp. 500–505, 2015.

[52] J. Feliciano, M.-A. Storey, and A. Zagalsky, "Student experiences using GitHub in software engineering courses: a case study," in *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*, pp. 422–431, IEEE, 2016.

[53] S. Eraslan, J. C. C. Rios, K. Kopec-Harding, S. M. Embury, C. Jay, C. Page, and R. Haines, "Errors and Poor Practices of Software Engineering Students in Using Git," in *Proceedings of the 4th Conference on Computing Education Practice 2020*, pp. 1–4, 2020.

[54] J. Börstler and T. B. Hilburn, "Team projects in computing education," *ACM Transactions on Computing Education (TOCE)*, vol. 15, no. 4, pp. 1–5, 2015.

[55] R. Chanin, J. Melegati, A. Sales, M. Detoni, X. Wang, and R. Prikladnicki, "Incorporating real projects into a software engineering undergraduate curriculum," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pp. 250–251, IEEE, 2019.

[56] Z. S. H. Abad, M. Bano, and D. Zowghi, "How much authenticity can be achieved in software engineering project based courses?," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, pp. 208–219, IEEE, 2019.

[57] D. Oguz and K. Oguz, "Perspectives on the Gap Between the Software Industry and the Software Engineering Education," *IEEE Access*, vol. 7, pp. 117527–117543, 2019.

[58] S. McLeod, "Likert Scale," 2008.

[59] W. Silva, I. Steinmacher, and T. Conte, "Students' and instructors' perceptions of five different active learning strategies used to teach software modeling," *IEEE Access*, vol. 7, pp. 184063–184077, 2019.

[60] M. L. Fioravanti, B. Sena, L. N. Paschoal, L. R. Silva, A. P. Allian, E. Y. Nakagawa, S. R. S. Souza, S. Isotani, and E. F. Barbosa, "Integrating project based learning

and project management for software engineering teaching: An experience report," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pp. 806–811, 2018.

[61] M. Souza, R. Moreira, and E. Figueiredo, "Students Perception on the use of Project-Based Learning in Software Engineering Education," in *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, pp. 537–546, 2019.

[62] A. Dominguez, H. Alarcón, and F. J. Garcia-Peñalvo, "Active learning experiences in Engineering Education," 2019.

[63] A. Garcia-Holgado, F. J. Garcia-Peñalvo, and M. J. Rodriguez-Conde, "Pilot experience applying an active learning methodology in a Software Engineering classroom," in *2018 IEEE Global Engineering Education Conference (EDUCON)*, pp. 940–947, IEEE, 2018.

[64] S. Chowdhury, C. Walter, and R. Gamble, "Toward Increasing Collaboration Awareness in Software Engineering Teams," in *2018 IEEE Frontiers in Education Conference (FIE)*, pp. 1–9, IEEE, 2018.

[65] G. Beranek, W. Zuser, and T. Grechenig, "Functional group roles in software engineering teams," in *Proceedings of the 2005 workshop on Human and social factors of software engineering*, pp. 1–7, 2005.

[66] R. van Cann, "Optimal Team Composition in Distributed Software Development," in *Collaboration in Outsourcing*, pp. 160–182, Springer, 2012.

[67] S. Akbar, E. Gehringer, and Z. Hu, "Poster: Improving Formation of Student Teams: A Clustering Approach," in *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)*, pp. 147–148, IEEE, 2018.

[68] I. Bosnić, I. Čavrak, M. Orlić, and M. Žagar, "Picking the right project: Assigning student teams in a GSD course," in *2013 26th International Conference on Software Engineering Education and Training (CSEE&T)*, pp. 149–158, IEEE, 2013.

[69] J. M. Hogan and R. Thomas, “Developing the software engineering team,” in *Proceedings of the 7th Australasian conference on Computing education-Volume 42*, pp. 203–210, Australian Computer Society, Inc., 2005.

[70] J. R. Quinlan, “Induction of decision trees,” *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[71] J. R. Quinlan, “C4. 5: Programming for machine learning,” *Morgan Kauffmann*, vol. 38, p. 48, 1993.

[72] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[73] Y. Zhang and B. Wu, “Research and application of grade prediction model based on decision tree algorithm,” in *Proceedings of the ACM Turing Celebration Conference-China*, pp. 1–6, 2019.

[74] S. N. Liao, D. Zingaro, K. Thai, C. Alvarado, W. G. Griswold, and L. Porter, “A robust machine learning technique to predict low-performing students,” *ACM Transactions on Computing Education (TOCE)*, vol. 19, no. 3, pp. 1–19, 2019.

[75] J. Dhanpal and T. Perumal, “Efficient graph clustering algorithm and its use in prediction of students performance,” in *Proceedings of the International Conference on Informatics and Analytics*, pp. 1–5, 2016.

[76] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[77] L. Xu and K. Veeramachaneni, “Synthesizing tabular data using generative adversarial networks,” *arXiv preprint arXiv:1811.11264*, 2018.

[78] J. Fan, T. Liu, G. Li, J. Chen, Y. Shen, and X. Du, “Relational data synthesis using generative adversarial networks: a design space exploration,” *arXiv preprint arXiv:2008.12763*, 2020.

[79] G. Douzas and F. Bacao, “Effective data generation for imbalanced learning using conditional generative adversarial networks,” *Expert Systems with applications*, vol. 91, pp. 464–471, 2018.

[80] N. Park, M. Mohammadi, K. Gorde, S. Jajodia, H. Park, and Y. Kim, “Data synthesis based on generative adversarial networks,” *arXiv preprint arXiv:1806.03384*, 2018.