Machine Learning based Recommendations to aid Educational Planning and Academic Advising through the Virtual Academic Advisor System

Sri Venkata Bhavani Likitha Vijjapu

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Computer Science & Software Engineering

University of Washington

2019

Committee:

Erika Parsons

Marc Dupuis

Wooyoung Kim

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Machine Learning based Recommendations to aid Educational Planning and Academic Advising through the Virtual Academic Advisor System

Sri Venkata Bhavani Likitha Vijjapu

Chair of the Supervisory Committee:
Erika Parsons

Computing and Software Systems

The process of educational planning and academic advising are critical for supporting student retention rate and on-time graduation. Faculty-based advising has been widely adopted by several community colleges because it results in a better advising experience for a student. However, due to lack of resources, outdated technologies, and growing diverse student population, the process increases the workload on already overwhelmed faculty. The role of technology in different areas of education sector is being slowly adopted, for instance, to provide various services such as online classes, registration services, and classwork maintenance. Despite these advancements, it is not being used for the purpose of academic advising. In this thesis, we discuss how machine learning (ML) and other technologies can be used to assist with faculty-based academic advising in higher education planning. The Virtual Academic Advisor (VAA) is an initial software solution to address this problem. Various ML-based techniques such as supervised learning, natural text processing, collaborative filtering, and sequence classification are explored to provide new functionality for the VAA. Collaborative filtering is used to give a recommendation of study plans based on a set of input parameters. Sequence Classification is used to predict possible suitable college majors, based on the course work designated for a student. In addition, we propose a strategy to generate synthetic data, necessary because it is nearly impossible to collect copious amount of real

observations necessary for properly training ML-based modules. These various approaches will be integrated into the VAA system, to help faculty with the advising process, saving time for more meaningful conversations with students, and providing students with the ability to explore different educational paths.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to thank my chair, Professor Erika Parsons, for her supervision and support during this Capstone thesis. Ever since taking a class on High-Performance Computing, to the thesis work, Professor Parsons has always been a great teacher and an advisor. I would also like to express my gratitude to my thesis committee members Professor Marc Dupuis and Professor Wooyung Kim for offering their valuable advice and for their time and encouragement. A special thanks to Everett Community College faculty member, Professor Matthew Fuentes for providing me guidance and helping out through difficult design decisions. This project would not have been possible without your constant support and feedback.

DEDICATION

To my parents and my dear husband.

Chapter 1

INTRODUCTION

1.1 Background

Educational or Academic planning is the process of determining a student's pathway to achieve their academic goals, while considering the different procedures of attaining them and target educational institutions [11]. Students may have a number of paths and universities to choose from, making academic planning a confusing process, particularly in the face of personal difficulties [12]. Planning has been found to be one of the most essential elements in helping students reach their goals while pursuing higher education.

Higher educational institutions are increasingly serving people from diverse backgrounds and needs, for instance, veterans, first-generation students, working adults, international students, etc. Many of these students plan to transfer to a university after their two years of starting at a community college [12]. The students face many challenges related to the lack of cultural, academic, financial, social support, etc. [13]. Understanding students' professional goals are fundamental to their success and to help improve the quality of American Higher Education [14]. This success has been observed to be critically influenced by student - advisor interactions [13].

Academic advisors bridge the gap between student goals and educational institutions, guiding them through different university's requirements and help them schedule the courses accordingly to complete their studies in a timely fashion. During an advising session, degree options, course requirements, course selection, transfer procedures, and career choices are discussed [13]. Advisors do their best to ensure that educational experience aligns with professional and life goals, while maintaining consistency of academic plans [15][16].

According to [17], students who received academic advising had a higher rate of persis-

tence and proceeded into their second year of education, compared to students who have not participated in advising sessions and have attended only the orientation program. Moreover, proper advising increases the chances of a student finishing math courses and transferring into a four-year college [18]. Finishing math and science courses successfully is a major challenge faced by students of STEM (Science, Technology, Engineering, and Mathematics). A student that does not undergo proper advising is less motivated and inspired to take these courses.

In most community colleges, faculty act as academic advisors. The practice of faculty-based advising came about in the early 2000s, based on the idea that the classroom is the most important point of contact between the student and the college, and faculty hold an important position in the student's educational development [19]. When a teacher works as an advisor, they serve more purpose than just advising, for instance:

- Helping students figure the rationale behind their education.
- Comprehending the seemingly disconnected aspects of the plan.
- Basing the educational choices on a sense of efficacy.
- Enhancing learning experiences by relating them to the knowledge gained.

Despite these advantages, advisors now serve more duties and are stretched thin due to the minimal budget in community colleges. During an advising session, advisors have to spend time gathering information from multiple resources to understand the student's background, limitations, and goals. The advisors should also collect logistics information such as course prerequisites, transfer requirements, course times and offerings, etc. Even when these parameters are similar across sets of students, the advisor has to work through the entire process each time to construct an academic plan. These faculty end up spending many extra hours every week, especially at the beginning of an academic calendar. For instance, in Everett Community College (EvCC), the student and faculty manually fill out a paper which records the course work to be considered in different quarters (see Figure 1.1). This additional workload limits the time faculty can be engaging with the students holistically

Associate of Science – Pre-Engineering

Computer and Electrical Engineering

This checklist is targeted at transfer students with an interest in one of the above engineering majors at the University of Washington or Washington State University. Students should meet with an advisor and maintain this checklist while at Everett Community College. The quarter before expected completion, this checklist should be submitted with a diploma application to the Enrollment Services Office. Note: Though courses in a foreign language are not required in the Associate of Science degree, some universities may require two or three quarters of foreign language for admission or for graduation.

Note: Prior to starting some or all of the following courses, students should:

- ☐ Complete ENGR 101 (formerly 109) recommended for all students considering an engineering major
- ☐ Complete ENGL 098 or earn a placement score into ENGL& 101
- ☐ Complete MATH& 144 or MATH&142 or place into MATH& 151
- ☐ Complete PHYS& 114 or physics placement test
- ☐ Complete PHYS 130 before PHYS& 233
- ☐ Complete CHEM& 140 or place into CHEM& 161
- ☐ Complete ENGR 121 and PHYS& 241/231 before ENGR& 214
- ☐ Complete ENGR 111 and MATH& 142 before ENGR 121

Student:_____

❑ **COMPLETION of Diversity Course** _____ (Where Completed/Course Title) _____ (Year Completed) _____ (Grade)

Course Number	Course Title	Credits	Quarter Completed	Grade
COMMUNICATIONS SKILLS (5 credits) [1]				
ENGL& 101	English Composition I	5		
MATHEMATICS (Pre-requisite Math courses may also be required.)				
MATH& 151	Calculus I	5		
MATH& 152	Calculus II	5		
MATH& 163	Calculus 3	5		
MATH 260	Linear Algebra	5		
MATH 261	Differential Equations	5		
HUMANITIES AND SOCIAL SCIENCE (15 credits, in three different disciplines. One course must be selected from Humanities, and the other from Social Sciences. The third course may be from Humanities or Social Sciences. For acceptable courses, see course list for the Associate of Science – see separate guide. See Notes 1 and 2.)				
SCIENCE AND ENGINEERING (37 credits. **CS& 141 is an acceptable substitute for CS& 131 for this degree)				
CHEM& 161	General Chemistry I	5.5		
CS& 131**	Computer Science	5		
ENGR 111 (see Note 3)	Intro to Engineering 1	5		
ENGR& 204	Electrical Circuits	5		
PHYS& 241/231	Engineering Physics I	5.5		
PHYS& 242/232	Engineering Physics II	5.5		
PHYS& 243/233	Engineering Physics III	5.5		
SPECIALIZATION COURSES (minimum 22 credits; select minimum **five** as appropriate for intended major and transfer institution. Please see the last page of this guide for course recommendations by intended transfer institution.)				
BIOL& 222	Majors Cell/Molecular	5		
CHEM& 162	General Chemistry II	5.5		
CS 143 or 132	Computer Science II	5		
CS 233	Advanced Data Structures	5		
ENGR 121	Intro to Engineering 2: Design	5		
ENGR 202	Logic Circuits	6		
ENGR 205	Electric Circuits Lab	1.5		
ENGR& 214	Statics	5		
ENGR& 215	Dynamics	5		
ENGR& 224	Thermodynamics	5		
ENGL& 230	Technical Writing	3		
ENGR 240	Applied Numerical Methods	5		
MATH& 264	Calculus 4	4		

Total: minimum 104 credits required, minimum 2.0 GPA. See Note 2.

Note 1: Use one of these courses to satisfy the diversity requirement.
Note 2: Students transferring to WSU should take ECON& 201 or 202 AND either HIST 103D, HIST 170D ANTH 116D, ANTH&206D or HUM 110D.
Note 3: ENGR 111 may be waived, at the Engineering faculty's discretion, for students transferring to EvCC with advanced standing in engineering.

Figure 1.1: The transfer guide from Everett Community College to University of Washington or Washington State University [1].

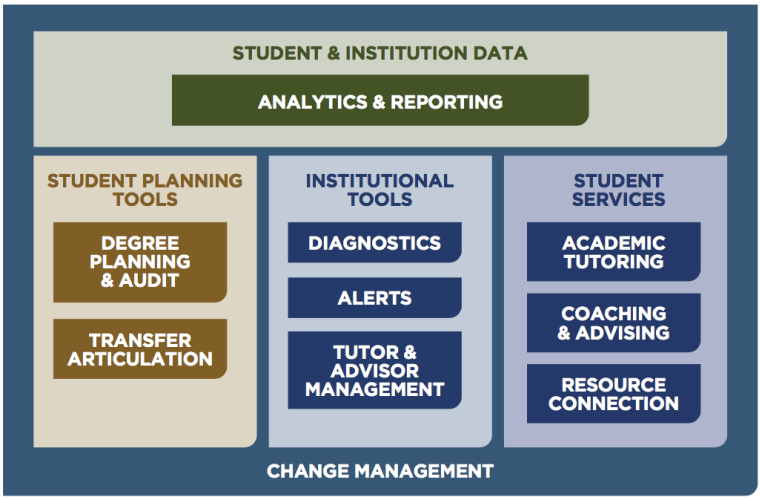


Figure 1.2: The 9 distinct categories where technology can be used to improve the efficiency of the services provided to the student [2].

during an advising session. Furthermore, students usually have questions that go beyond the creation of an academic plan, having to choose between constructing the plan or having their questions answered during their timed session, making it more challenging for them to reach their goals.

The use of modern software can help handle advising tasks that can be automated, e.g., academic plan creation (educational planning), hence improving the quality of the academic advising experience where the conversation can focus around personal challenges that burden students [20]. Various fields of the educational sector such as, degree planning, academic tutoring, etc. are using technology-based solutions. According to [12], the educational sector can be classified into 3 categories (see Figure 1.2) where technology can improve the efficiency of various services [2]. The number of educational administrators and advisors who are willing to incorporate technology to improve educational efficiency increased in 2017-2019 (see Figure 1.3). By using technology to provide support services and automate otherwise monotonous tasks, advisors can save time and better focus on relevant discussions that can help student retention and success [12].

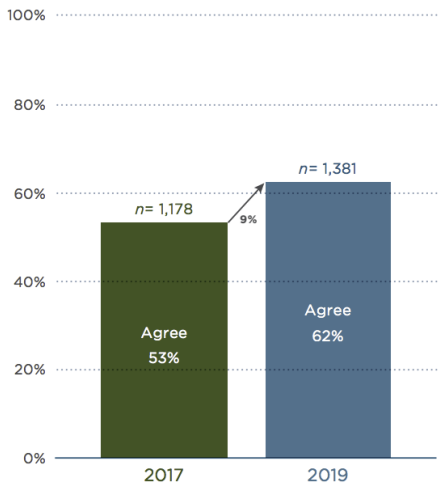


Figure 1.3: The number of advisors who agree on the importance of technology for advising and increased effectiveness [2].

The Virtual Academic Advisor (VAA) is an end-to-end software system envisioned by Dr. Erika Parsons, proposed as an initial solution [3] to automate the repetitive tasks associated with academic plan creation and hence facilitate effective educational planning. In particular, the goal of the VAA is to automate the process of making study plans, thereby improving the faculty-based advising experience. This system helps overcome problems such as the disproportionate ratio of students-to-faculty, advisor availability, lack of faculty experienced in advising, lack of formal advising training for new faculty, and the lack of a centralized source that captures advising "tribal knowledge." The VAA also aims to reduce the amount of redundant work, thereby saving time during advising sessions. This system will also try to prevent associated human-induced errors and, as a long-term goal, to motivate educational institutions to adopt new technologies in areas otherwise neglected [21][22][23] such as student services and institutional tools (see Figure 1.2).

1.2 Problem Statement

As discussed in Section 1.1, the increasing number of students attending community colleges (particularly in STEM disciplines) together with the lack of appropriate funding for these

schools has resulted in less than ideal academic advising situations, which adds unnecessary workload for the faculty. Community colleges host a large number of students from diverse backgrounds and underrepresented groups. These students generally have little to no understanding about the options and pathways they can follow to successfully transfer to a university. The absence of adequate and efficient software technology makes parts of the advising process tedious and repetitive for the already overwhelmed faculty.

Faculty should instead be able to focus on having a wholesome conversation with the students to help them understand the demands and challenges from their different options. Software used to aid in academic advising is generally outdated and inflexible, in some cases offering hard-coded options for creating educational plans that cannot be configured, changed or scaled. This software is costly as the advising functionality is usually just a small part of comprehensive software for student and school management. In other words, academic advising is an area that has been neglected by software development, and hence little understood. The use of machine learning models can improve the experience of academic advising and educational planning.

1.3 Existing System

The current VAA system consists of a minimal implementation to generate study plans based on a few student preferences as input parameters. In Figure 1.4, we can see a sample of study plan where courses are distributed across different quarters. The system currently consists of several modules such as basic UI for interaction, a deterministic recommendation engine, and modules for interaction with the database.

The current recommendation engine includes a prerequisite-network generator. The information provided by the community college is used to determine the prerequisites. It also implements a basic machine learning module which is used to classify plans into appropriate ranks that indicate how good or suitable the plan is, based on input parameters such as student preferences and other constraints. Ranked plans can be used by the recommendation engine to present at least one plan that best aligns with the input preferences. Figure 1.5

Figure 1.4: A sample education plan generated by the current VAA implementation through the deterministic recommendation engine.

offers the workflow of the existing system. It serves the fundamental purpose of reducing the workload on an advisor.

Limitations of the existing system. The current VAA software implementation has various limitations, both functional and theoretical, for instance:

- Considers only a limited number of features to describe a study plan, for instance, it does not capture the length of the course plan, the sequence of courses, break in a sequence of core courses. These features turn out to be very important in plan creation.

- Study plans are not validated, i.e., there are no checks for missing core requirements; this could result in a plan missing a *major* required course or a broken course sequence.

- Original design was targeting students with defined Major and transfer university goals; however, in many cases, students start college without knowing this.

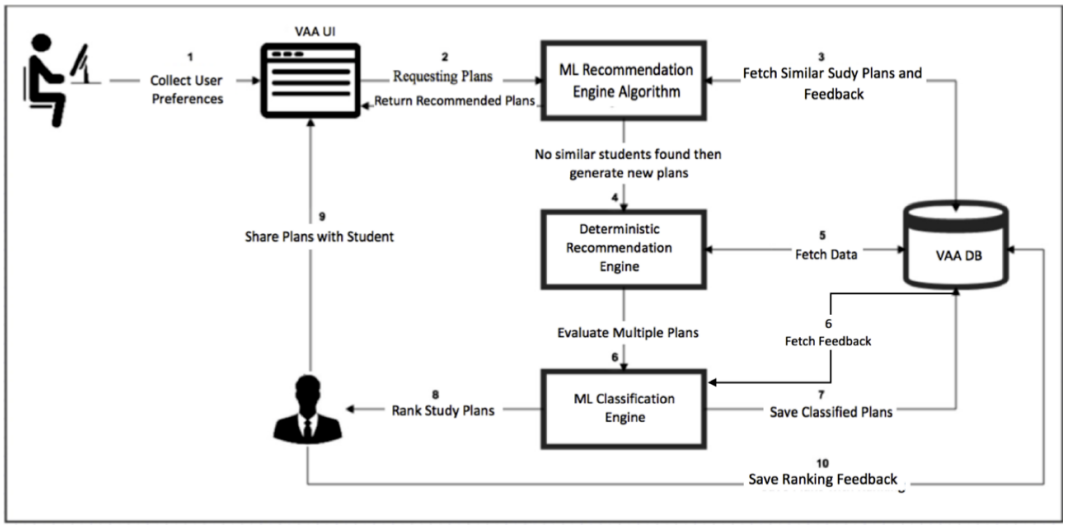


Figure 1.5: The workflow of existing VAA system [3].

1.4 Proposed Solution

The VAA system targets students at community colleges who aim to transfer to a university or plan on completing a 2-year Associate Degree at the current community college. The system automates academic plan creation based on student limitations, preferences, and background while trying to reduce the workload on the advisors. Advisors can then have more meaningful sessions with students instead of performing repetitive tasks. The proposed solution consists mainly of improving and adding new features and functionality to the existing VAA system. The work discussed in this paper can be split into four main goals illustrated in Figure 1.6. The functionality of each module can be seen in detail in Section 1.4.

Increasing the Dataset. A large amount of data is required for proper training and assessment of ML approaches. Collecting real-life examples of study plans is challenging due to the lack of consistency in human-generated plans, coupled with poor software tools available to advisors of EvCC to capture and maintain plans. During the course of the present research, a new batch of human-generated study plan examples has been collected

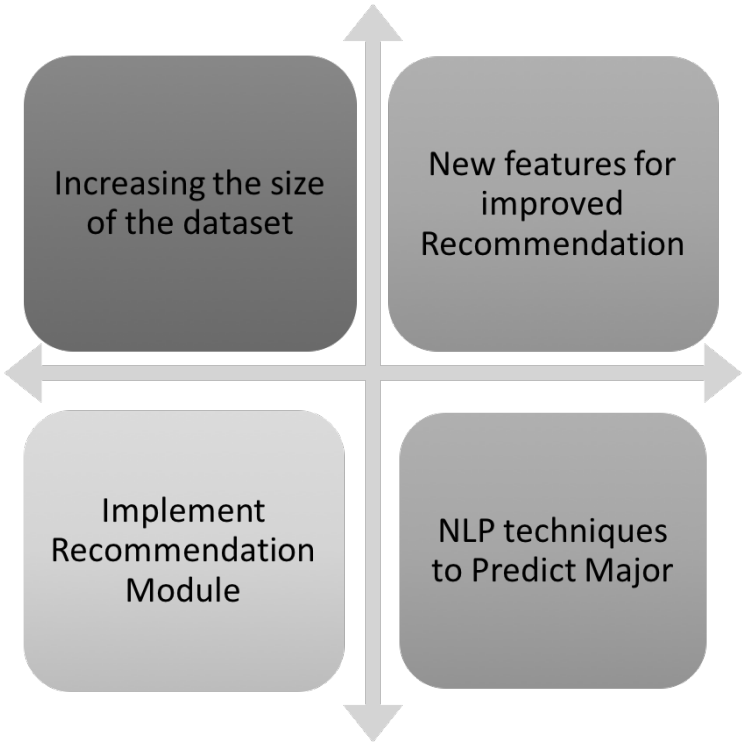


Figure 1.6: The 4 main goals of this paper. Detailed description of each module can be referred from Section 1.4.

from EvCC and has been added to the VAA database. However, only about a hundred useful examples can be extracted during advising season. For this reason, in addition to new plans, we have used existing examples to generate synthetic data. Synthetic generation makes sure that the data is statistically distributed while covering essential cases. This strategy will be discussed in detail in Section 4.3.

Aggregating Features for Improved Recommendation. In previous work done to explore the potential of an ML-based recommendation engine for the VAA [3], only a few features from the plan were considered. The limitation in the number of features considered was mainly due to the small number of inadequate study plan examples, and it wasn't possible to extract or represent other features. The proposed work makes use of an updated implementation that, along with previously existing feature, considers new features including

user preferences, and new study-plan metrics. New user preferences include Mathematics and English course-sequences starting points. These can be used to calculate features like distribution of core courses in different quarters/semesters, length of the plan, and a sequence breaks in the course sequence. This information can be used to build a more insightful representation of the study plan and improve the quality of the examples and data used for training an ML module. The rank of the plan is used to determine the quality of the plan. The rank is determined based on a predefined rubric used as a ranking standard, and this is vital in ensuring that the ranking is consistent for different advisors over time. This rank also captures the cognitive information regarding a study plan, which is central to the method of collaborative filtering.

Recommendation Module. Due to a large number of course options, time offerings, prerequisite paths, student's limitations, and goals, the different possible study plans is combinatorial. The proposed module will create a study plan recommendation for the students based on the student's goals and preferences. This module takes in student's preferences as input parameter and returns a list of recommended study plans from which the student can choose. The principle of Collaborative Filtering is used to generate recommendations. The details of the implementation are discussed in Section 4.3.

Major Prediction Module. The present research aims to explore ML and Natural Language Processing (NLP) techniques to analyze the course work to determine a major for the student. The NLP module is implemented as a new/improved module of the VAA system. This work will help to address the dilemma students face while selecting a major and college combination to make a study plan. This module will predict a major depending on the course work designated for a student. This module is discussed in detail in Section 4.4.

1.5 Scope

The problem of creating study plans can be applied to many educational levels and is common across many educational institutions. For the purpose of our research, we part from the following premise to narrow down the scope and to develop a feasible proof of concept:

An automated Machine Learning Recommendation System can reduce the workload for advisors and provide a better advising experience for the students.

The scope of the work described in this document targets Everett Community College (EvCC) focusing on Engineering Department courses and transfer data to Washington States universities. Based on this, the scope of our work can be outlined as follows:

- The dataset composed of study plans provided by Everett Community College (EvCC) will be used for identifying relevant features to determine the efficiency of a plan for a STEM major.
- Synthetic data will be generated and used in combination with the data collected from EvCC.
- Current implementations will be extended to take additional input preferences to make the recommendations more relevant and customized.
- Increase the number of features to identify the quality of a plan
- ML-based strategies suitable for building a recommendation module based on this type of data will be investigated.
- Study plans will be analyzed to find patterns relevant for transfer scenarios, i.e., to a target university and major from the community college.
- NLP techniques will be explored to understand the sequence of classes.
- Sequence Classification techniques will be explored to enable a student to explore different "Major" options.

1.6 Motivation

Improving the student retention rate and on time graduation rate are the major challenges faced by the US higher education system. These rates have not shown significant improvement in the last few years, and tend to be worse for people from underrepresented groups and diverse backgrounds [24]. Academic planning has been found to be one of the most essential things to help students reach their goals while pursuing higher education. Due to the lack of

proper planning and sufficient information, the process of completing degree requirements is protracted, fragmented, and unfulfilled.

Our work primarily concerns with requirements of community colleges. Faculty-advisors in these institutions faces many challenges while working with students to devise a successful educational path. These institutions have limited resources to improve the academic advising experience, among which is the lack of efficient, automated degree planning systems. The reason behind this problem is that several states follow performance-based budgeting, where the performance of the college [25] determines the funding for a college. This performance-based budgeting has led many colleges to allocate more resources for improving the condition of education in the institutions. This performance is evaluated based on a set of predefined metrics. These metrics measure quantitative growth but not qualitative development [26].

Many community colleges (like EvCC), have an archaic approach for academic advising. Study plans are not only prepared by hand, but they are also predetermined or have a"one size fits all" method. These study plans are not customized for each student's needs. In recent years, there have been some efforts to develop software to automate the process. However, these solutions have been rudimentary, relying on static or hard-coded strategies [12].

At the same time, ML-based approaches are becoming ubiquitous in many application areas – the education sector is not one of them. A wide range of new ML-specific software tools makes ML an attractive, feasible, and novel approach to tackle aspects of academic advising. The VAA project is a pioneer solution that aims to improve academic advising by providing functionality for customizing educational plans based on student's requirements and preferences.

1.7 Outline

The rest of this thesis is organized as follows. Chapter 2 discusses the related work and Chapter 3 introduces the basic concepts used in implementing the proposed goals. Chapter 4 talks about architecture of the proposed system and the methods followed to implement

them. Chapter 5 is followed with results of the methods tested. The last chapter, Chapter 6 concludes the work with ideas to further the research. The references are listed at the end of the paper.

Chapter 2

RELATED WORK

In this chapter, we discuss the existing work and technologies that support and/or relate to our research and the proposed goals. First, we present the current state of academic advising and the different systems following collaborative filtering. We then talk about the research in text classification and how it can be used to address our problem. Finally, we observe how the current work differs from the work previously done on VAA.

Current State of Academic Advising

As discussed in Section 1.1, academic advising, and educational planning are critical for a student to achieve their academic goals. In Section 1.6, we point out an increase in the number of tools and methods being developed to address the deficiencies in current advising scenarios [2].

Such tools address different areas of advising such as caseload management; alerts, signals and notifications; performance management and measurement [2]. For instance, Salesforce offers a cloud solution called Salesforce Advisor Link [27] for caseload management for advisors . This system tries to streamline advising operations and maintaining student records. Services like Aviso retention [28] analyses student's performance and identifies at-risk students and strives to improve the student retention rate . Tools like Signal Vine Blended [29] communication provides solutions for notifications and direct interactions with students. Student management tools like Edunav's optimize [30] aggregates degree plans to make predictions about demand for courses. Another example is Loud Cloud [2], which provides very user-centric solutions, mainly considering the student's financial ability.

Though many of these tools deliver some functionality, they fail to address the fundamen-

tal problem of academic advising: *easy, tailored development of study plans*. The students need to identify the required courses for them to graduate and satisfy all the prerequisites for different courses. The VAA system tries to address this problem, but initial deterministic approaches were slow and computationally heavy, and alternative ML-based approaches lacked amount and variety of data [3]. Our proposed approach is to upgrade the ML-based recommendation system using collaborative filtering (for details, see Section 3.2 and Section 3.2).

Recommendation Systems based on Collaborative Filtering

Collaborative filtering is widely used to generate recommendations due to its simplicity in making distance-based recommendations. Recommendation systems are widely used and deployed in various application areas, especially in e-commerce, entertainment, technology-enhanced learning, knowledge management, and many others [31].

For instance, in [31], recommendations of movies are generated for users based on local and global user similarity, this is a framework developed based on user-based collaborative filtering. Similarly, GroupLens [32] is a system that generates recommendations of news articles. In [33], personalization of tag-based searches [34] is done using collaborative filtering based on clustering .

Despite being widely used and researched, collaborative filtering based recommendation systems have not been used in the education sector, particularly in academic advising and educational planning.

Research in Sequence Classification

Sequence classification is a powerful approach that has a wide range of applications including bioinformatics, security, social networking, and, etc. [35], including specific uses in DNA classification, anomaly detection, and sentiment analysis. For example, DNA classification is performed to understand the biological details of an organism [36]. Successive letters are used to represent DNA sequences. These letters are transformed into words, and sequence

classification is then performed [36]. Similarly, in [37], sequence classification is done on movie reviews to classify as either positive or negative. The sequence classification of reviews (here, movie) is called sentiment analysis. In [37] Naive Bayes classifier and Support Vector Machine (SVM) classifiers were used to classify the reviews. Anomaly detection also uses sequence classification, where the sequence of user actions are analyzed to detect if the user's behavior is consistent with previous actions [38].

In this research, we use sequence classification techniques to predict potential college majors based on the courses that need to be taken by the student (refer to Section 4.4.3 for details).

Previous Work on the VAA

VAA is an automated ML-based Recommendation System that can improve the quality of faculty-based academic advising for educational planning. Over the past couple of years, various strategies and approaches have been used to come up with a solution to the problem of automating study plan creation. Initial efforts consisted of using a deterministic approach that yielded a factorial-order number of solutions due to the combinatorial nature of the problem. This approach was not only computationally expensive but also made the implementation extremely difficult to maintain and scale. An approach considered to replace such implementation, and currently under improvement, is an adaptation of the Job Shop Algorithm (JSS) [6], which uses a greedy approach to assign interdependent n jobs (required courses and prerequisites), to m machines (availability of the class).

The nature of the problem and the type and amount of data required are some of the biggest challenges in this project, added to the fact that expert assistance (EvCC faculty) will be needed to validate and test the system. In addition, there is little to no work or research done in this area, making every problem a completely new challenge. The nature of data as well as the need for large amounts of valid test data to test the efficiency of the system make the process challenging.

As an initial ML-based solution to the problem, a collaborative filtering based recommen-

2018	Current Work
Limited human-based dataset: few real-world observations	Augmented human-based dataset: two new batches of real-world academic plan incorporated
Limited synthetic dataset: restricted amount of artificially generated data, no specific strategy to generate this	Augmented synthetic dataset: large amount of synthetic data generated following strategy specifically designed for this
Limited number of features	Augmented features, both preferences and study-plan metrics
Numerical-based ML strategies	Researched ML strategies for categorical data as well as NLP approaches
No integration with UI	Better integration with UI to facilitate data exploration and ranking

Figure 2.1: Comparison of the current work with previous work done on VAA.

dation system has been discussed in [3] . Though the work has been proved to be functional, it did not address specific issues as discussed in Section 1.3. In this work, we address such limitations. Figure 2.1, elaborates the difference between the previous research and the current work.

Chapter 3
BACKGROUND CONCEPTS

In this chapter, we discuss some of the concepts required to implement the different parts as defined in methods (Chapter 4). We first talk about supervised learning in Section 3.1 which is used for ranking a feature set (for details see Section 4.3.2 and Section 4.3.5), and for predicting college-major for a student (see Section 4.4). In Section 3.2, the implementation principle behind the recommendation module is discussed. We finally introduce the concept of sequence classification in Section 3.3.

3.1 Supervised Learning

Machine Learning enables to identify relations between different features in data and makes the analysis of data simpler. Every instance of the dataset used by a single ML model represents the same features. If the instances are represented with labels, it is called supervised machine learning [39] in contrast to unsupervised learning where the labels are not present. In the process of training a classifier, a set of rules are learned to aid the process of assigning a label to a new instance. The first step for this is formatting or reformatting the data to maximize information gain to define (learn) the set of rules. The classifiers can be grouped into different subsets. In this paper, we mainly work with Logistic Regression, Logic-Based classifiers, Support Vector Machines, and Ensemble Classifiers.

Logistic Regression. The Logistic Regressor is a non-linear classifier that tries to identify boundaries in a given set of data points to differentiate between the classes [40]. These boundaries are identified by equations which are developed after training the existing data. In a logistic regression classification, regressor uses one-vs-rest approach while trying to

classify data with multi-class labels. The probability of the data belonging to one class vs. the rest of the classes is estimated. A simple logistic function plot looks like in Figure 3.1. The details of the implementation of Logistic function are discussed in Section 4.4.3.

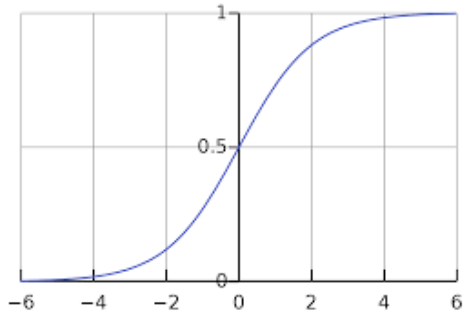


Figure 3.1: Representation of a simple sigmoid function [4].

Logic Based Classifiers. Logic-based classifiers are mainly of 2 kinds, Decision Trees, and rule-based classifiers [5] [39]. Both of these models work based on a set of rules. Decision Trees predict a label to the data based on the rules it learns while training [41]. A simple decision tree can be visualized in Figure 3.2 where the tree consists of a set of nodes and leaves.

In rule-based classifiers, the model is represented as a set of If-Then rules [5]. Knowledge can be easily represented as rules. A set of rules and actions are defined, and logical operations are performed on them [5][39].In this thesis Decision Trees have been used to label the study plans to help build the recommendations (see Section 4.3.2). Decision Trees are easy to implement and maintain. They work well with both continuous and categorical data [41].Implementation details of Decision Tree are discussed in Section 4.3.5.

Ensemble Classifiers. Ensemble classifiers are a combination of classifiers. Ensembles are used to improve the accuracy over a single classifier [5], [10]. The process can be visualized with the help of Figure 3.3. The predictions of several base estimators are combined to predict and improve the performance of the model in comparison with a single predictor.

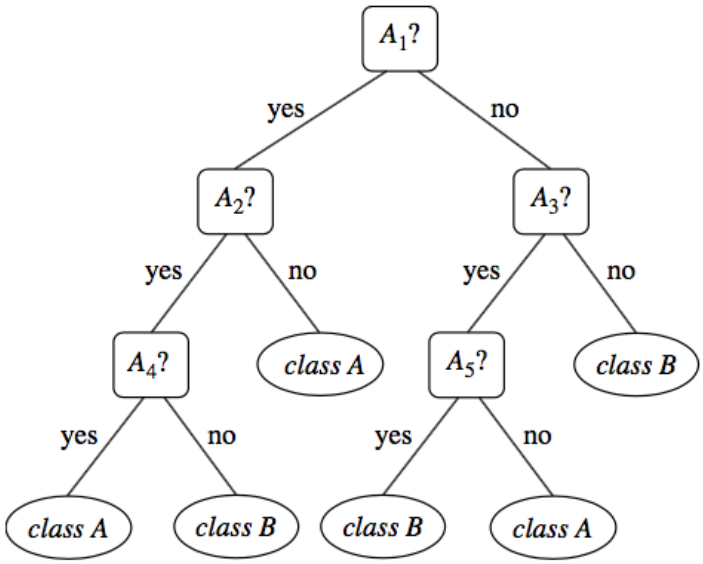


Figure 3.2: A representation of a simple Decision Tree [5].

According to [10], ensemble methods are of two kinds:

- Averaging ensemble methods: Several base classifiers are used to predict a label, and the average of the results of all the models is used to estimate the performance of the classifier. Random Forest Classifier, Extra Tree Classifier, and Bagged Classifier are averaging ensemble methods. In this thesis, we use the mentioned classifiers to test for improvement in performance w.r.t. Decision Trees classifier. In Section 4.3.5 we discuss about the individual classifiers in detail.
- Booster ensemble methods: In this method, the base classifiers are run sequentially, and the one model tries to reduce the variance of the other model [5], [10]. Booster classifiers work together so that several weak performing models can be combined to improve the overall prediction.

Support Vector Machines. Support vector machines can be used for both linear and non-linear data. The original training data is mapped into a higher dimension through a non-

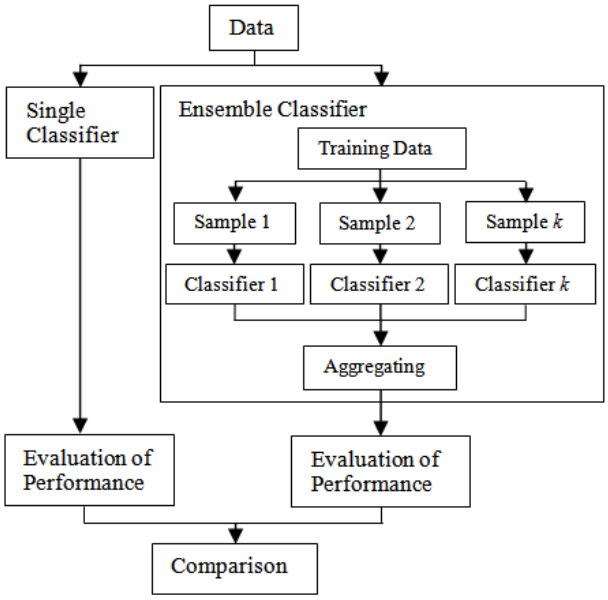


Figure 3.3: An illustration of how a simple classifier differs from an Ensemble Classifier [6].

linear mapping (kernels) [7] when the data cannot be separated by linear decision planes. Decision boundaries are calculated to divide the tuples belonging to one class from other class. Simply put, if we have an n-dimensional feature space and all the data points plotted in that plane, SVM tries to find a boundary in that plane which divides the data points.

In Figure 3.4 we see a 2D data plane. The dotted lines represent the different ways we can build decision boundaries for this data space. In this thesis, we use SVM for sequence classification. For implementation details, see Section 4.4.3.

3.2 Recommendation System

As mentioned in Section 2, recommender systems are widely used in different industries. A recommendation system is defined as a system that can predict or give suggestions to the user based on past preferences. A simple recommender system uses data mining and filtering techniques to provide a user with recommendations. There are two kinds of recommender systems: Content-Based Recommendation system and Collaborative Filtering based recom-

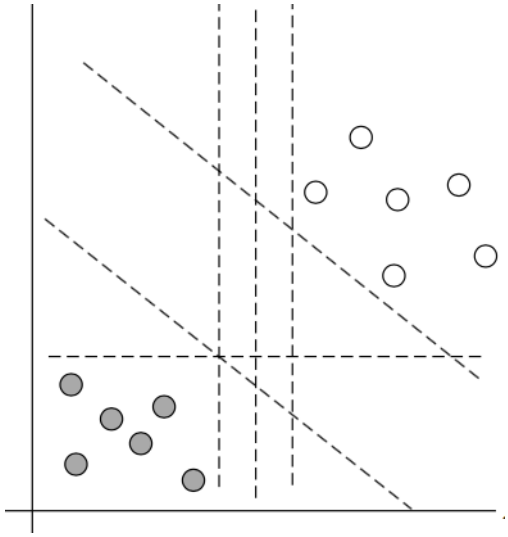


Figure 3.4: The figure shows 2D data and the different planes that can be developed as decision boundaries using SVM classifier [7].

mendation system. The difference between the two methods is discussed below and can be visualized through the Figure 3.5.

Content Based Filtering

In this method, the recommendations are generated based on the metadata about the users. For example, recommend users with poems based on the content of a poem. These kinds of systems require more data like information about features of the item, or details about the movie [42].

Collaborative Filtering

Collaborative filtering can be done to retrieve information efficiently to generate recommendations [31]. The measure of similarity of the user is critical in generating recommendations. The usage of user similarity is what makes collaborative filtering different from content-based filtering. The similarity of the users is measured by "rating". The rating can be tuned in a way to capture the context of the user [43]. For instance, in VAA while recommending study

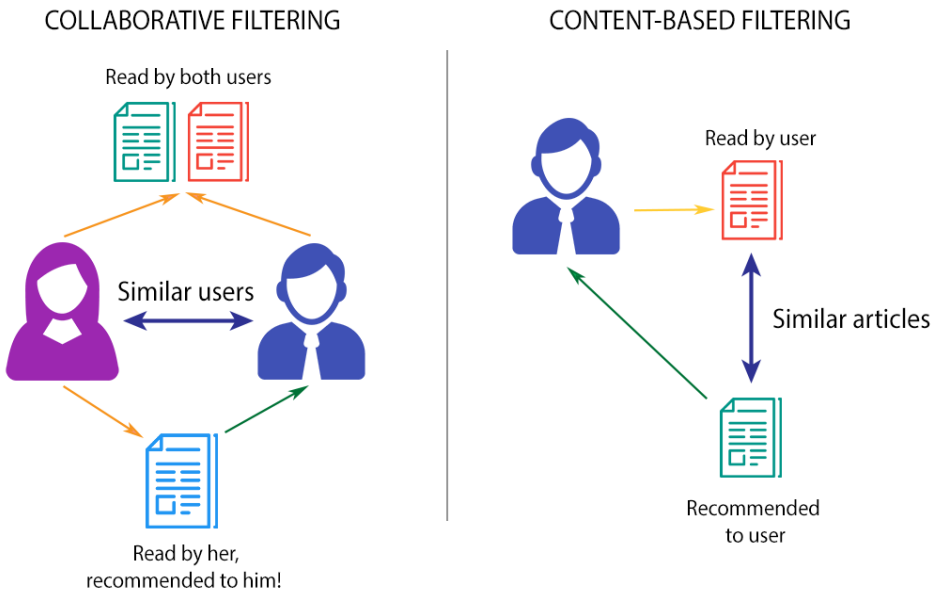


Figure 3.5: A simple illustration to bring the difference between collaborative filtering and content based filtering [8].

plans, we want to recommend study plans which are most relevant for the student in terms of intended major, college, or any other parameter. A student's preferences and the features of the plans are combined to assign a rating which captures the context of a user. The rating details are discussed in detail in Section 4.3.2. The similarity of the user is combined with the rating to make a prediction for another user. This method of implementation is called user based recommendation [31]. In this paper, we implement user-based collaborative filtering to generate recommendations of study plans for students. The details of implementation are discussed in Section 4.3.6.

3.3 Sequence Classification

In simple words, given an input sequence, the process of predicting the class label is known as sequence classification [44] (see Figure 3.6). The sequence has an order and has to be observed while training ML models. As mentioned in Section 2, sequence classification has

wide applications in real-world applications. Sequence classification follows the same rules of supervised classification. Sequence classification can be categorized into three main types [45]:

- Feature Based Classification: The sequence is transformed into a vector of features, and classification is done on the feature vector.
- Distance-Based Classification: Like distance based collaborative filtering, distance is used to measure the similarity between two sequences to determine the quality of classification [45].
- Model-Based Classification: Statistical models such as hidden Markov models are used to classify sequences [35].

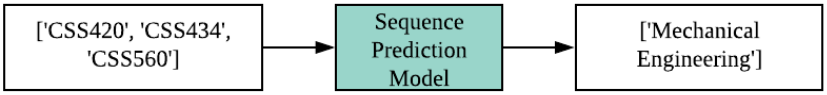


Figure 3.6: A sequence of courses are taken as input by the classifier which predicts a Major after doing sequence classification.

In this paper, we use feature-based classification to classify course plans to predict a college major. The details of implementation are discussed in Section 4.4.3.

Chapter 4

METHODS

4.1 Architecture

This section describes the architecture of the system and the research methods used to implement the proposed goals. The VAA system is designed on the principles of service-oriented architecture for a web application. A service-oriented architecture is a widely accepted standard for efficient utilization of web resources, and for development and integration of enterprise applications [46]. In this section, we further elucidate the architecture details and function of each component in the system.

3-Layered Service Oriented Architecture

Service-oriented architecture (SOA) is a method for interaction between user and the internet [47]. A service-oriented architecture is expanded on the client/server paradigm. The main premise of SOA is to allow applications to communicate through the platform-independent standard HTTP protocol [48] making the Application Programming Interface (API) calls platform and language agnostic. Every modern language and device can make a HTTP request making the communication between the modules easier. By adopting SOA, reuse, growth, and interoperability of the system can be enhanced. SOA can be implemented through Simple Object Access Protocol (SOAP) based web services and Representational State Transfer (REST). By following REST methodology for implementing the architecture, the interface is typically stable and resilient to internal changes [49].

The VAA architecture, depicted in Figure 4.1, is presented as a three-layered architecture of logical computation. This provides means for modularizing the user interface, business logic, and data storage layers.

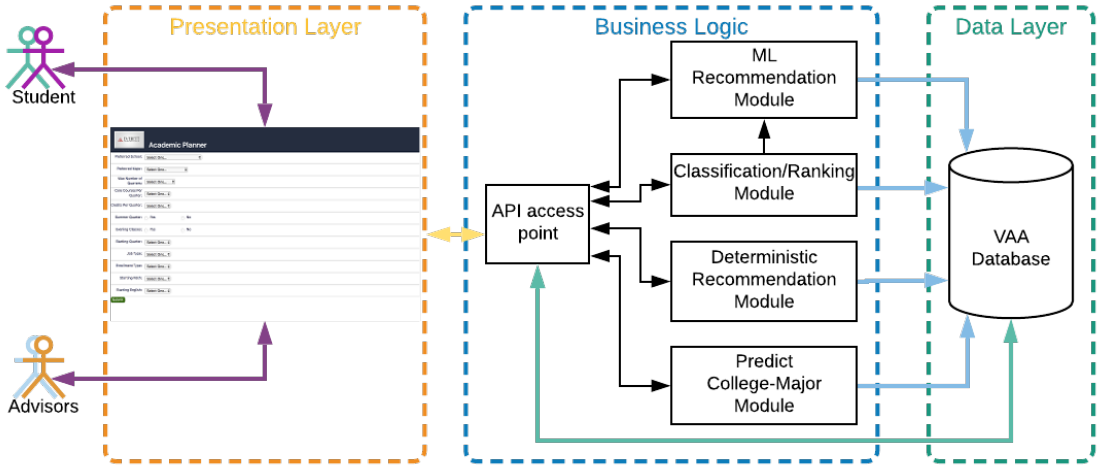


Figure 4.1: High-Level Architecture of VAA. We consider ML Recommendation Module, Classification Module and Predict College-Major Module.

Presentation Layer

The presentation layer is the front-end of the system providing a user interface. This user interface is accessible through the web application. The student's preferences such as the major, college, Math and English starting points are taken in from this layer and passed on to the business logic layer by making API calls to the back-end. The generated plans are displayed through the interface. Similarly, the list of designated courses is communicated as another API call so that the major the student can to go to can be predicted and displayed back to the user.

Business Logic Layer

The business logic layer (BLL) or the application layer is the core of the system providing the functionalities of VAA. The different ML modules are implemented in Python in this layer. Any language can be used to implement the services in this layer as the system is made language agnostic by the use of API calls. The API access point makes calls to the

different modules of the BLL. The different modules in the BLL are described below.

Deterministic Recommendation Module. The deterministic module uses an adapted version of the job shop scheduling algorithm [50] to generate a study plan for the student. Preferences and other parameters are taken in as input for the scheduling algorithm. The module also creates course prerequisite networks and stores it in a graph data structure.

Classification/Ranking Module. The classification module implements supervised learning classification strategies to rank study plans. The ranking determines the "goodness" of a plan. The plans generated by the deterministic algorithm or the plans that have been suggested by the recommendation engine are classified through this module. This module can be called by an advisor or can be triggered when a new plan is generated. The classification of the plan is essential for the efficient implementation of the ML recommendation module.

ML Recommendation Module. The ML recommendation module is implemented based on the principle of collaborative filtering as discussed in the Section 3.2. Collaborative filtering uses collective intelligence in generating a customized plan for a student. This module makes use of the preferences and the rating to make context aware recommendations to the student.

College Major Prediction Module. This module makes use of language processing techniques and uses sequence classification methods to predict a major a student could aim for. The sequence of course work designated to a student is taken in as input to the module. Refer to Section 4.4.3 for implementation details.

Data Layer

The database for the VAA system is a relational database hosted through Microsoft SQL server. The database can be accessed through a data access API or by the individual modules directly. This database was designed to scale out and support any kind of educational schedule besides the current scope (EvCC).

4.2 Supporting Software

The system was developed in integration with a number of softwares, frameworks and libraries. This section describes these technologies and their purpose in the context of our research.

Anaconda

Anaconda [51] was used as the main implementation environment. Anaconda's Navigator, is a Graphical User Interface that supports multiple applications, environments and channels simultaneously, which simplify data analysis, scientific computations, visualizations. The environment of choice is Python 3 (which is also the default in Anaconda), which has a number of libraries, making it a sensible choice for data analysis and machine learning research and development. Furthermore, the navigator supports different software like Jupyter Notebook, R studio, Spyder, VSCode, etc. [51]. Any other preferred software can be installed and used through Navigator.

Python

Python is a high-level object oriented programming language. High level data structures, with dynamic typing and dynamic binding make Python suitable for rapid application development and for connecting multiple components together. It supports a number of packages that enable development of modular and reusable code. Moreover, the absence of compilation step increases development productivity. It is complemented with a stack trace provided upon code exceptions to facilitate debugging [52].

While R is a powerful procedural language for statistical analysis and building data models, frequently used for research purposes [53], Python has been preferred because it enables more general purpose applications, while supporting a variety of ML and Deep Learning libraries that facilitate the development of sophisticated data models. Furthermore, Python's readability and performance allow for faster prototype and product development

[54]. For these reasons, Python is a better choice for the VAA data analysis and ML needs and target application-like functionality.

Machine Learning Libraries. Libraries such as NumPy [55] and Pandas [56] make it easy and convenient to manipulate data. NumPy is the most basic package offered by Python supporting scientific computing. In addition, NumPy can be used to accommodate multi-dimensional generic data, which is useful for integrating multiple database content [55]. NumPy is part of the SciPy ecosystem, which are is a bundle of open source software for scientific computing in Python [57] . Pandas is built on top of NumPy, deriving its name for the purpose is serves: Python for Data Analysis. Pandas has exclusive data structures-like series and data frames, which enable efficient handling of data tables [57].

Sci-kit learn, which is offered as a part of SciPy has a collection of algorithms and tools, which support various stages of Machine Learning. It has algorithms for PreProcessing, Model Selection, Dimensionality Reduction, Supervised Learning Through Classification, Unsupervised Learning through Clustering and many more [10]. Matplotlib, is a 2D visualization library offered by Python which has multiple features which support visualization enabling an efficient data analysis and classification module evaluation[58].

4.3 Recommendation of Study Plans

In this section, we discuss the methodology to design and implement an ML-based recommendation module and a classification/ranking module for the VAA system. These modules work together to generate a study plan for a student based on input preferences.

4.3.1 Data Source

The initial dataset used for our recommendation strategies research was created based on study plans collected from EvCC. Each study plan, together with associated input parameters, is considered to be an observation in a human-readable form. Initial plans were crafted by hand, over a few years, by faculty serving as academic advisors. Since the number of

human-crafted plans is too small for ML purposes, we decided to incorporate artificially-created data to increase the size and quality of our dataset.

Definition 1. Let $\mathbf{S}$ be the set of all study plans available in the scope of our research, and let $S_i \in \mathbf{S}$ be a study plan observation, where $i = \{1, 2, \ldots, M\}$ and M is the total number plans, including human-crafted and generated synthetically.

Human-generated study plans contain valuable information that can be used to extract significant features that capture relevant traits of a plan's structure and its creation, we will refer to such features as "metrics". Input parameters (Definition 2), together with the metrics (Definition 3), compound a set of features F suitable for classification, defined below in Definition 4. The dataset used for classification is a set of feature vectors of the form F, the set of all such observations is denoted by $\mathbb{F}$ (see Definition 5). The overall dataset $\mathbb{F}$ used for classification is composed by human-crafted plans mapped into feature vectors, as well as synthetically created feature vectors.

Definition 2. Let P be the set of input parameters, including user preferences and academic requirements, and let $p_i \in P$ be a single parameter, where $i = \{1, 2, \ldots, |P|\}$. This set is captured from a user through the UI (see Figure 4.1.

Definition 3. Let T be the set of metrics used to describe a study plan, and let $t_i \in T$ be a single metric, where $i = \{1, 2, \ldots, |T|\}$.

Definition 4. Let F be the set of features that describe a study plan, where $F = P \cup T$, and $f_i \in F$ represents a single feature, where $i = \{1, 2, \ldots, N\}$ and N is the total number of features, i.e., $N = |T| + |P|$. F is a mixture of both categorical data and quantitative data.

Definition 5. Let $\mathbb{F}$ be the set of feature vectors that describe study plans in $\mathbf{S}$. Let $\bar{x} \in \mathbb{F}$ be a feature vector that describes a specific study plan $S \in \mathbf{S}$, and let $\phi : \mathbf{S} \mapsto \mathbb{F}$ be a function that extracts the features from a human-crafted study plan and arranges them as the values of the feature vector as defined by F (refer Definition 4). Note that $|\mathbb{F}| = M$.

Note that the function ϕ used in Definition 5 to map from a human-readable plan to a feature vector, is really a combination of functions, each of which is used to map each specific feature. These mappings can be done by looking at the plan metadata and searching associated values in our database. Some of the mappings are straight forward, for instance, to obtain $SummerPref$ (Table 4.1), it is enough to check whether there are courses scheduled in the summer quarter. Another example is $LengthQuarters$, where it suffices to count the number of quarters as stored in the database for a specific plan. Other features like sequence breaks are more complex and will be discussed in Section 4.3.3.

Synthetic Data Generation. The amount of human-crafted plan is too small to properly design ML-based model, and manual generation of plans is tedious, time-consuming, and error-prone, thus, it becomes imperative to devise a better, automated way to generate plans for the dataset. The VAA system contains a basic recommendation module that uses a "deterministic" approach to generate additional plans and can be used for verification. This module is scalable but still in its early stages and it is out of the scope of this paper. Alternate methods for generating data need to be considered. There are a number of available software tools with enough functionality to create customized datasets, for instance, [59] generates data for certain predefined data types like human information, credit card information, geographical location and so on. This generated data can be exported into multiple formats. Another tool example is Mockaroo [60], which has a wider collection of customizable data categories classes, such as movies, IT, health, etc. However, both these aforementioned tools fail in creating a useful dataset to mimic study plans, and cannot be used for analyzing or modeling an educational planning system.

For these reasons, we need to create a synthetic dataset that will better simulate the data required by our system. For the scope of this document, we focus on a strategy to generate synthetic observations in the form of feature vectors rather than in the form of human-readable plans.

The data simulation captures human knowledge of the possible scenarios and translates

into a script. This knowledge ensures that the data generated is realistic. Python offers a number of libraries and packages to generate synthetic data, for instance, the *Faker* package is the core of many data generating algorithms [61], which provides functionality for generating data with information such as names, address, etc. Additionally, pydbgen [62] uses *Faker* to provide more features and generating data tables, data frames, or MS Excel formatted data.

To create suitable synthetic data, we use Algorithm 1, which is used to generate new observations in the form of feature vectors with features F, i.e., $\bar{x} \in \mathbb{F}$. The algorithm uses the valid range of values a feature can be in. This information is given by an advisor based on their experience with academic advising. The algorithm generates a new observation, as a vector of features F, based on the valid range of values for each feature:

Definition 6. Let V_f be the set of values a feature $f \in F$ can take. Given a vector $\bar{x} \in \mathbb{F}$, let $\bar{x}_i$ be the component corresponding to feature $f_i \in F$, where $i = \{1, 2, \ldots, N\}$, then $\bar{x}_i \in V_{f_i}$.

Table 4.1 contains all the features in the scope of our work, i.e., F. This table also contains the sets of values valid for each feature.

Algorithm 1 Synthetic data generation.

Inputs:

Set of valid ranges for each feature in F

Outputs:

A new observation $\bar{x} \in \mathbb{F}$ (Definition 5)

Begin:

Create a new feature vector $\bar{x}$ initialized to 0

for $i = \{1, 2, \ldots, N\}$ **do**

Choose a random value $rval$ from V_{f_i} (Definition 6)

$\bar{x}_i = rval$

end for

$\mathbb{F} = \mathbb{F} \cup \bar{x}$

Table 4.1: Features identified in the process of Feature Engineering

Feature Name	**Description**	**Value**
PrefCollege	The university a student is aiming for	Colleges in WA state
PrefMajor	The major a student is aiming for	Colleges in STEM majors
PrefMathStart	The starting point for the sequence of math classes to be taken	MATH courses
MathSeqBreak	Break or missing order in the sequence of Math classes to be taken	0 - 100
PrefEnglishStart	The starting point for the sequence of English classes to be taken	ENGL courses
EngSeqBreak	Break or missing order in the sequence of English classes to be taken	0 - 100
PrefNumQuarters	Student's preference of the number of quarters to finish the course	Integer, ideal value is between 8 and 20
LengthQuarters	Actual length of the plan in number of quarters	Integer, ideal value is between 8 and 20
PrefNumCore	Prefered number of core courses per quarter	Integer, ideal value is between 3 and 5
ActualNumrCore	Actual number of core courses per quarter	Integer, ideal value is between 3 and 5
JobType	Student's employment status. Useful for determining the length of the plan.	Full Time, Part Time Unemployed
EnrollmentType	Student's enrollment status. Useful for determining the length of the plan.	Full Time, Part Time
SummerPref	Student's preference of having classes in summer quarter	True/False
SummerOpt	Generated plan has classes in summer quarter	True/False
PrefStartQuarter	Student's preferred start quarter	Fall, Winter Spring,Summer
StartQuarter	Generated plan start quarter	Fall, Winter Spring,Summer

4.3.2 Data Labeling

As discussed in Section 3.1, supervised learning requires data to be properly labeled for model training and prediction. In our case, each element of our dataset comprises of feature vectors representing plans (refer Definition 5) needs to be categorized.

Original study plans are ranked by a human (advisor) based on a standard rubric. Table 4.2 contains the rubric for ranking a plan, which is defined based on advisor expertise and can be used to maintain a standard for ranking. The rank itself then becomes a label for a particular study plan, representing its "goodness" based on how likely a human advisor is to recommend the plan for a specific student. The standards listed in Table 4.2 are considered as part of the strategy to label synthetically generated data.

Definition 7. Let L be the label assigned to a study plan, where $L = \{-1, 1, 2, 3, 4, 5\}$, 5 is the highest possible rating, 1 the lowest, and -1 is used to indicate an invalid plan.

Table 4.2: Standards for ranking plans: the labels have been suggested by EvCC faculty/advisors based on their experience with educational plans.

Rank	Description
-1	The study plan is incorrect. This is usually associated with critical mistake made by the deterministic algorithm (or a human if created by hand).
1	Study plan has at least 3 or more major issues. Generally, this kind of study plan should not be recommended to students as it is beyond repair.
2	Study plan has at most 2 major issues. This type of plan needs major repair before recommending it to a student.
3	Study plan has at most one major issue or multiple minor issues. This is a plan requires the moderate extent of changes before recommending it to a student.
4	Study plan has minor issues like a few quarters may have medium gaps in course schedule or level of difficulty in each quarter is relatively consistent with at most 1 quarter of inconsistent difficulty.
5	Study plan is flawless.Courses are clustered together in a manner that level of difficulty in each quarter is consistent.

The rank captures the context of an advisor and can be used to generate context-aware recommendations (see Section 3.2 for detail on collaborative filtering).

To generate a label for a synthetic "observation", a penalty system is used. A penalty is calculated using a subset of features including metrics and input parameters to generate a new metric that is incorporated into F and factored by a specific weight.

Definition 8. Let $\mathcal{T} \subset T$ be a set of metrics derived for the purpose of helping the process of labeling synthetic data. The members of $\mathcal{T}$ are obtained using a subset combination of features from F, which we will refer to as $\mathcal{F}$. A metric in $\mathcal{T}$ is denoted as t_i where $i = \{1, 2, \ldots, |\mathcal{T}|\}$.

Definition 9. Let $\mathcal{W}$ be a set of predetermined weights, and let $w_i \in W$ be a specific weight intended to weigh the value of $t_i \in \mathcal{T}$.

The value of each $t_i \in \mathcal{T}$ is scaled by the corresponding w_i, gauging the contribution t_i value to the "quality" of a plan, i.e., the values of features in $\mathcal{T}$ value will affect the overall penalty of a given plan, and the penalty can be used to determine the label. Note that the value of the penalty is inversely proportional to the the rank, i.e., a higher the penalty implies lower rank.

The features in $\mathcal{T}$ that have been considered to calculate the penalty are:

- Sequence breaks: A break in the sequence of courses to be taken is captured by a value called sequence break. In this paper, we discuss math and English sequence breaks as they are required by all STEM majors across colleges.
- Length of the plan: The length of the plan in terms of quarters.
- Enrollment type, Job Type and Number of core classes per quarter: This combination is evaluated to ensure the students do not overload themselves. This ensures the quality of education for the student.
- Contrast in plan from student's preference: The student's preferences against the advisor recommended plan. For instance, the student does not want to take classes in summer, but the plan requires the student to take classes in summer.

Labeling Strategy. As part of synthetic data creation, a label L will be assigned to each artificial observation feature vector F. For artificial data, it will be derived based on a penalty calculated by weighing the values of features $\mathcal{T}$ (see Definition 8).

Definition 10. Let $\mathbf{C}$ denote the total possible penalty for a particular study plan S, and $0 \leq \mathbf{C} \leq C_{max}$. It is calculated by weighing the *values* of the $\mathcal{T}$ features corresponding that particular observation:

$$\mathbf{C} = \sum_{i=1}^{|T|} w_i t_i$$

.

For the purpose of our experiments, we set the maximum penalty $C_{max} = 330$. This value, however, could be changed and adapted depending on the problem, features examined, and human expertise is taken into consideration. We also split the range $[0, 330]$ into six sub-ranges of size 55, one for each of the possible label values (see Definition 7). This and the strategy used for programmatically labeling synthetically create data is addressed in detail in Algorithm 2.

4.3.3 Feature Engineering

Feature engineering can be used identify the set of features (see Definition 4) that can be used to describe a study plan (see Definition 1). Feature engineering is the process of identifying metrics $\mathbf{F}$ that describe the data observations (user parameters and metrics from study plans) to facilitate ML classification approaches [63]. Some of the metrics we have considered include the length of the plan, the number of courses per quarter, distribution of core courses, sequence of classes, among others (see Table 4.1). Domain knowledge is key in identifying the features, which help in classification. In our case, most of the features used in our experiments have been identified as relevant by faculty-advisors from EvCC.

Calculating Sequence Break. Calculating sequence breaks addresses one of the goals of the proposed research by adding new features to the feature set previously used by the original classification module [3].

Algorithm 2 Synthetic data labeling.

Inputs:

A set of weights W

A feature vector F_i corresponding to a plan $S_i \in \mathbf{S}$

The set of features $\mathcal{T}$

The set of valid label values $L = \{-1, 1, 2, 3, 4, 5\}$ in this order

$C_{max} = 330$ and $C_{increment} = 55$

Output:

A label $l \in \{-1, 1, 2, 3, 4, 5\}$ based on $\mathbf{C}$

Begin:

for each $t_j \in \mathcal{T}$ **do**

Extract the corresponding value for feature t_j from F_i

end for

for each $t_j \in \mathcal{T}$ **do**

$\mathbf{C} = \mathbf{C} + w_j t_j$

end for

$range_{lower} = C_{max} - C_{increment}$

$range_{upper} = C_{max}$

for each $l \in L$ **do**

if $range_{lower} \leq \mathbf{C} < range_{upper}$ **then**

return l

end if

$range_{lower} = range_{lower} - C_{increment}$

$range_{upper} = range_{upper} - C_{increment}$

end for

Calculating sequence break is important because there are certain sequences of courses that are lengthy but relevant and required for certain majors. For instance, many engineering students must take a minimum of 5 Math courses in a specific order to satisfy prerequisite requirements. Ideally, such a sequence should be scheduled as early as possible in a study plan, with its courses in consecutive quarters to avoid unnecessarily long schedules that would result in delayed student graduation or transfer. Furthermore, early Math courses serve as prerequisites for many other courses, like Physics, Programming, and Advanced Math. Breaks in particularly long sequences are undesirable and academic advisors try to avoid them, with the goal of making more efficient study plans. This precondition is captured through a feature called "Sequence Break".

We will use Table 4.2 to illustrate the sequence break approach. Consider an ideal sequence of Math courses exemplified in the top row of the figure; this sequence pertains to a course plan α_i. An ideal sequence of courses is built from an official course network. There can be more than one acceptable course plan for a given pair of starting and ending points of a sequence. The plan that is the most similar to the course plan α_i can be used to calculate sequence break value (score) using a similarity measure like the Levenshtein distance (refer Equation 4.1, i.e., the more similar α_i is to an ideal plan, the shorter the distance and the higher the sequence break value. For instance, in Figure 4.2, experiment 1 has a score of 100 because it is exactly the same as the ideal sequence, while experiment 5 has a lower score because it differs from the ideal sequence in at least three places.

$$lev_{a,b}(i,j) = \begin{cases} max(i,j) & \text{if } min(i,j) = 0. \\ min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1_{(a_i \neq b_i)} \end{cases} & otherwise \end{cases} \tag{4.1}$$

The sequence break feature is also used to calculate the penalty (Definition 10) incurred for missing the strict sequential ordering of the courses involved. It is an important feature in determining the label of a plan (Definition 7).

Ideal Sequence	*"MATH 142& MATH 151& MATH 152& MATH 163& MATH 264&"*	
Experiment	**Test Course Plan**	**Score**
1 : Course plan follows the exact sequence	MATH 142& MATH 151& MATH 152& MATH 163& MATH 264&	100
2 : Course plan follows has breaks later in the plan	MATH 142& MATH 151& MATH 152& MATH 163& break MATH 264&	95
3: Course plan follows has breaks early in the plan	MATH 142& break MATH 151& MATH 152& MATH 163& MATH 264&'	92
4: Course plan skips few	MATH 142& MATH 152& MATH 163& MATH 264&	89
5: Course plan follows order but has a number of breaks	MATH 142& break MATH 151& break MATH 152& break MATH 163& MATH 264&	85
6: Course plan has all courses but order is missed	MATH 163& MATH 264& MATH 142& MATH 151& MATH 152&	79

Figure 4.2: Examples of Math course sequence scenarios with different sequence breaks.

Study plans can be seen as a sequence of words, where the words represent the courses and the sequence in which they appear represent sentences. Applying the Levenshtein distance concept [64], we can compare the similarity of a generated plan and an ideal plan. Levenshtein distance is calculated based on Formula 4.1, which is used to calculate similarity between a course plan (Definition 11) and ideal sequence of courses (Definition 13). The higher the similarity, the lower the distance. Sequence break score calculation is detailed in Algorithm 3.

Definition 11. A course plan α can be defined as the sequential ordering of courses in a given study plan.

Definition 12. A directed acyclic graph $G = (V, E)$ where the set of nodes V represent different courses, and E is the set of directed edges representing dependencies between courses (e.g., precedence) can be considered as the course network.

Definition 13. A sequence of courses can be seen as a path I from a course network G where $I \subset V$, and the edges connecting courses in I strictly indicate a prerequisite relationship,

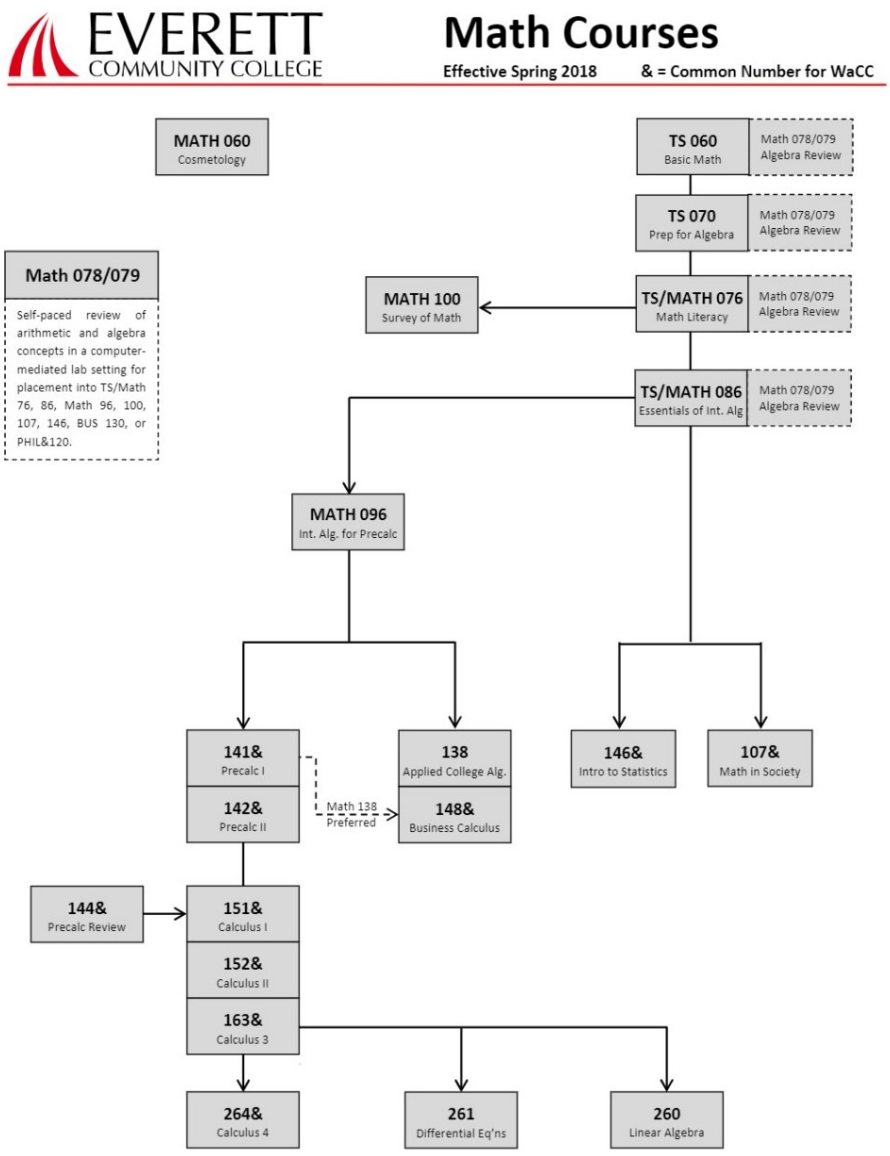


Figure 4.3: Math Course Network at EvCC. This course network is used to build the ideal course sequence for a given starting point [9].

i.e., i_j is a prerequisite to i_k where j and k are the indexes of the each course and $j < k$, and i_{start} is the first course in the sequence, and i_{end} is the last course in the sequence.

In this research, we only consider Math and English sequences, however, there are other

Algorithm 3 Algorithm for calculating Sequence Break

Inputs:

Subject of interest $Subj$ (either Math or English)

A course network G

A study plan $S \in \mathbf{S}$

A course sequence seq from the plan $\mathbf{S}$

A distance set $D = \emptyset$

$Subj_{start}$ and $Subj_{end}$

Output:

A sequence break b where $0 \leq b \leq 100$

Begin:

Build set of ideal sequences $\mathbf{I} = \{I_1, \ldots, I_H\}$ from G

for each sequence $I_j \in \mathbf{I}$ **do**

Calculate distance d_j between seq and I_j

$D = D \cup d_j$

end for

$b = min(D)$

return b the sequence break

subjects that have relevant sequences but they are out of the scope of this work. A starting point of a course sequence (e.g., Math) is the first course that a student should take in that sequence, and it is a prerequisite to following courses in that topic or other topics, for instance, a starting point for Math could be determined by a placement exam. On the other hand, an ending point of a course sequence is usually either a target course required for transfer or the highest level course offered at the college. Figure 4.3 shows EvCC's current Math course network, an example of a sequence would be starting point MATH 096 with ending point MATH 264&.

4.3.4 Data Preprocessing

Data preprocessing is the process of transforming raw data into understandable format [65]. Since our data comes from varied sources, it is necessary to remove inconsistencies to properly use it for experimentation, preprocessing and analyzing. From a high level, data preprocessing includes the following steps.

1) Sanitizing Data: Filling Missing Values
2) Data Analysis: Analyze the Correlation among different features.
3) Data Transformation: One hot encoding for categorical data.

The details of implementations at every step are discussed in the rest of this section:

Sanitizing Data. It is common to have missing values for data, especially when it comes from multiple sources. Python provides libraries for data analysis and statistics, which can be used to handle missing information [66]. According to [67] An elementary way of dealing with missing values is to remove the corresponding rows if the values are missing completely at random (MCAR). In the VAA data, this is not the case. The data, which was mainly missing was for the new features. Many plans in VAA have been collected before the new features were taken into consideration. These values are computationally calculated. For instance, Preferred Math Start ($PrefMathStart$) is a new feature, for the missing values, the first math class in the plan is taken as the preferred math start. Similarly, the preferred number of core courses per quarter ($PrefNumCore$) had many missing values. These missing values can be imputed with a statistical mean value [68]. The statistical mean offers a value based on general observation.

Data Analysis. One of the most important aspects of analyzing data is to find correlations between features. Correlation is used to observe relations between quantities of different features. We can estimate the value of one feature with respect to another feature [69]. Different methods can be used to find the correlation between different features. Generally, the

Pearson correlation coefficient is used to measure the linear correlation between continuous data [70]. It is calculated based on the deviation of an observation from the corresponding sample mean. For categorical data, the mean is non-existent, hence, this approach cannot be used to represent correlation [71].

Cramer's V (ϕ_C) [72] is an option to calculate the correlation between nominal values and it is calculated based on Pearson's chi square test χ^2 as shown in Equation 4.2 as follows:

$$\chi^2 = \frac{(n_{i,j} - \frac{n_i n_j}{n})^2}{(\frac{n_i n_j}{n})} \tag{4.2}$$

where n is the sample size, $n_{i,j}$ is the number of times each one of the variables X_i and Y_j were observed, and $i = 1, \ldots, r$ and $j = 1, \ldots, k$. This can be used to capture independence between two variables by taking into consideration the frequency of occurrence of each of these variables. This facilitates the analysis of correlations in categorical data. Equation 4.3 shows how ϕ_C is calculated.

$$\phi_C = \sqrt{\frac{\chi^2/n}{min(k-1, r-1)}} \tag{4.3}$$

One Hot Encoding. One hot encoding [10] is a widely accepted way to transform categorical data into a form that can be used by ML models. The most critical aspect of one hot encoding is that it does not introduce ordinality into the data. The data we have consisted of a number of features which are categorical in nature. They cannot be directly used by the ML models, and hence one hot encoding is performed on the features [10].

Grid Search with K-Fold Cross Validation. Grid Search and cross-validation are two different processes which are pipelined together to make an estimate of the best performing model. Grid search is used to perform parameter tuning. Parameter tuning is the process of finding the best values for the model's parameters to improve the accuracy of that model [73]. We specify a list of metrics against which we want to train our model. Grid search runs exhaustively on all the combinations of the metrics and finds the best performing set [74].

Cross-validation is a validation method used to ensure the performance of the model against new data is consistent as it was with training data [75], [76]. In k-fold cross-validation, we partition the dataset into k equal parts. We train the model on $k-1$ parts and validate against 1 part. We repeat this process until all the k parts act as validation sets. Figure 4.4 shows how the data is partitioned.

Figure 4.4: This figure illustrates how K-Fold cross validation splits the data for training and testing when K=5 [10].

In this experiment we use both of these methods together to make an estimate of the best performing parameter values while ensuring the model is consistent across the independent datasets.

4.3.5 Classification

The results from grid search with cross-validation Section 4.3.4 are used to perform supervised classification as discussed in Section 3.1. We train different models on both synthetic data and the original data set. In this section, we discuss the different models we use to train the data.

Training and Testing Split. The data is split in 4:1 ratio where 4 parts are used for training the models and the one part is used for testing the trained model. All the supervised

classification models in this module are trained and tested with this split ratio so that a comparison of models' performance against different data sets is consistent.

Decision Tree Classifier

As introduced in Section 3.1, by using Decision Tree classifier [5], we can build a Decision Tree which is a flow-chart like structure. Each internal node of the Decision Tree represents a test for an attribute $x \in F_i | F_i \in \mathbf{F}$ where $\mathbf{F}$ is the set of mappings between user parameters and metrics as discussed in Section 4.3.2 . Each branch represents the result of the test. The leaf nodes are the terminal nodes which hold a class label. In this experiment, we use CART (Classification and Regression Trees) to build a binary tree [10].

The Decision Trees are constructed greedily, in a top-down approach. Given a training set, it is recursively partitioned into subsets as the tree is built. The partitioning in the data happens until all the data at the node are under the same class for classification. In that case, the node becomes the leaf. The criterion to partition the tree at a node is determined by a measure of impurity. The split should happen in a way that the partition is as pure as possible. The two common measures of impurity are:

- Gini Index: It measures the probability that a given sample, F_i belongs to a class L_i. The measure of Gini impurity decreases with each split and ideally it should be 0 at the leaves where the probability of a random sample belonging to a Class is 1 [77].
- Entropy: Entropy measures the information gain [78]. Information gain can be defined as the difference between the amount of information know before the split and after the split [78].

We use both these criteria with grid search to determine the better performing combination for the dataset in hand and split the data as mentioned in Section 4.3.5.

Random Forest Classifier

As mentioned in Section 3.1, Random Forest classifier is a kind of an averaging Ensemble Classifier. A Random Forest is a set of tree predictors where each tree is dependent on the randomized subsamples of input data [79]. Random Forests reduce the risk of overfitting the data, which is quite frequent with Decision Trees [41]. The efficiency of a Random Forest Classifier is determined by the individual trees and the correlation between them. Given a training dataset, a Random Forest generates k Decision Trees. For each iteration, the data is sampled randomly with a replacement similar to the Bagged Classifier (Section 4.3.5). For splitting the data at each node, the best split is made based on a subset of total attributes. Whereas in the Bagged Classifier, the entire set of attributes is considered. The Decision Tree for a Random Forest is developed like a simple Decision Tree, (see Section 4.3.5).

Bagged Classifier

Bagging or bootstrap aggregation creates uniform data samples with replacement to create multiple data samples also known as bootstraps [80]. A single base learner is used to run different classifiers against the different bootstraps. The results of all the bootstraps are aggregated. In bagging the bootstraps are created with replacement [10]. This ensures the randomness of prediction in the base classifier. According to [81], [82], the instability of base classifier is desired, i.e., for a small difference in data, the prediction should vary a lot. Simply put, the base classifier should be a weak classifier [82]. Bagging algorithms are designed to improve the performance of weak classifiers.

Extra Trees Classifier

Extra Trees or extremely random trees, is a tree-based ensemble method [83]. Extra Tree functions are similar to Random Forests. In this method, the split at every node is made based on a subset of attributes selected randomly after which the best split is determined. Moreover, Extra Trees do not allow re-sampling the data, i.e., Extra Trees do not follow the concept of bagging [80].

4.3.6 Collaborative Filtering for Recommendation

Collaborative Filtering is the basic principle proposed by [3] based on which the ML recommendation module (Study Plan recommendation) has been developed. As mentioned in Section 3.2, collaborative filtering captures the context of the user [84]. In this paper, we use distance based collaborative filtering methods to implement a system to generate study plan recommendations for a student. The distance between the data points is used to calculate the similarity between users.

Distance based Collaborative Filtering. To calculate the similarity between two users, we can use methods such as Pearson Distance and Euclidean Distance Score [84]. This measure is called *similarity score*. In this implementation, we use Euclidean distance to calculate the similarity score. Let A, B be two sets of input parameters (see Definition 2), then, the Euclidean distance is given by Equation 4.4.

$$d(A, B) = \sqrt{(a_1 - b_1) + (a_2 - b_2) + \ldots + (a_i - b_i)} \tag{4.4}$$

Whenever there is a user request, a similarity score is calculated to find the set of neighbors. The study plans followed by the neighbors are recommended to the user. The user request consists of a set of input parameters (see Definition 14) which can be used to calculate the distance between other users.

Definition 14. Let $P\ '$ be a set of input parameters received in the user request where $|P\ '| = |P|$

The closest set of users with similar user parameters are found by calculating the distance with all the existing user parameters given in F (see Definition 4).

Let D represent the set of distances D_i from $P\ '$. To generate a set of study plan recommendation, R, for $P\ '$, the distance D_i of each $F_i \in F$ from $P\ '$ is calculated. The study plans S corresponding to the closest neighbors is given out as a recommendation R, such that $R \subset S$.

The steps followed to generate recommendations can be seen in the Algorithm 4.

Algorithm 4 Algorithm for generating recommendations

Inputs:

An input set of user parameters $\bar{y}\,'$ corresponding to each element of P

A set of feature vectors $\mathbb{F}$ and corresponding plans **S**

The labels L corresponding to each feature vector in $\mathbb{F}$

A set of distances $D = \emptyset$

Desired number of recommendations num

Output:

A set of recommended study plans R

Begin:

$Rank = 5$

while $|D| < num$ **do**

for $\bar{x}_i \in \mathbb{F}$ **do**

if $L_i \geq Rank$ **then**

Let $\bar{y}_i \subset \bar{x}_i$ where $\bar{y}_i$ corresponds to elements of P

Calculate distance d_i between $\bar{y}\,'$ and $\bar{y}_i$

$D = D \cup d_i$ add element to the distances set

$R = R \cup S_i$ add plan to the recommendations set

end if

end for

$Rank = Rank - 1$ since not enough plans with $Rank$ are found

end while

return R

4.4 Recommendation of College Major

In this section we discuss the methods followed to predict major for a student based on course work completed by students.

4.4.1 Data Source

As described in Section 4.3.1, initial data was extracted from a set of study plans provided by EvCC manually created by advisors. Useful information for our experiments included required course sequences, corresponding labels, major, etc. This information was used to train ML models to predict a major for a student. A combination of text processing and supervised learning strategies were used, however, since these plans are not enough for ML, we also use plans generated by the deterministic recommendation module (see Section 4.1) containing the same relevant features.

Data Labeling. As described in Section 3.1 and Section 4.3.2, supervised machine learning needs labeled data. In this experiment, the labels for a data item will be the college major. Student's preference of major is critical information for the advisor or for the deterministic module to generate plans. This information is already available for use. Existing information is used to implement new features for VAA to improve the functionality of the system and user experience.

4.4.2 Data Preprocessing

The general data preprocessing steps are described in Section 4.3.4. Next, we elaborate on these steps in the context of this experiment.

Data Sanitizing. Data sanitizing for text processing involves formatting strings and words in a way that most relevant information can be represented. Due to the nature of our original dataset, human-generated study plans provided by EvCC, it contained information and metadata irrelevant to our experiments. This is considered as noise that had to be removed to appropriately create training data. Along with that, there were various special

characters and trailing white spaces, every course name was enclosed in single apostrophes(‘’), all these had to be trimmed. Course names should follow a string pattern, e.g., “MATH 122”, “ENGL 233”, “ENGR& 345”, etc., and we want to consider only those courses that follow this pattern. Matching string patterns are created to identify valid courses and filter them out.

Data Transformation. As mentioned in Section 2, we use the concept of feature based sequence classification to solve our problem. In feature based classification, the sequence is transformed into a vector of features which are used to train a standard classifier [35], [45]. The following steps have been followed in data transformation:

1) Imbalanced Learning: The problem of imbalanced learning happens when there is more data belonging to a certain class, which introduces training bias because under-represented classes are given less weight. In the data set obtained for this experiment, most of the initial plans were designed for “Mechanical Engineering” (see Figure 4.5). To correct this problem, data oversampling is performed. Two common approaches used are “Random Over Sampler” and “Synthetic Minority Over-Sampling Technique”. We have chosen Random Over Sampling, where the samples from under-represented classes are repeated at random.
2) Count Vectorizer: It is used to convert the text of information into feature vectors [85]. Count vectorizer counts the frequency of a word (course) in the data. Using count vectorizer, the entire data can be represented in a 2-D matrix useful for getting insight into the data, like how some words occur more than other words, but not carry a lot of information. For instance, MATH classes are required for a lot of STEM majors, so the word “MATH” might not be useful to differentiate between the majors. To overcome this issue, we use TF-IDF explained next.
3) Term Frequency-Inverse Document Frequency (TF-IDF): It is another method to convert textual sequences into feature vectors [85] that can be used for training different ML models. This method assigns weights to the words instead of simply counting the words.

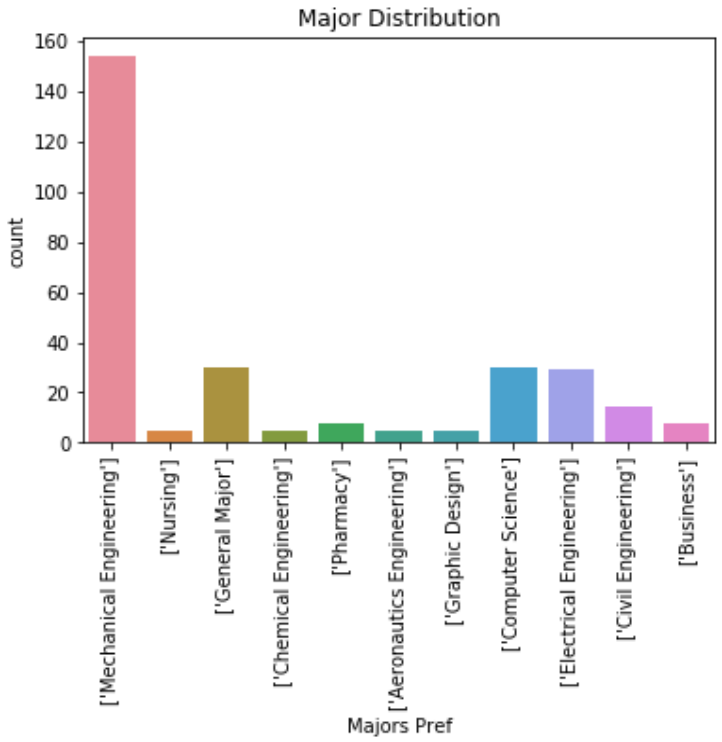


Figure 4.5: The distribution of number of plans with the intended major shows class imbalance in the data availble for training.

TF refers to term frequency and measures how often the word occurs, and IDF refers to inverse document frequency which measures the importance of a word [86]. Once we transform our data into vectors and handle class imbalance, we used supervised classification algorithms to predict the sequence.

Data Analysis. Data analysis was performed at different levels to gain more understanding of the available data. First, we use word clouds to explore the layout of courses in study plans. Word clouds are a straight forward and visually appealing method to visualize text data [87]. They serve as a starting point for a deeper analysis [88]. A word cloud is used to visualize words (courses) proportional to its frequency. Word clouds are a crude way to get a general idea regarding the most popular courses. The next step was to use count vectorizer

for more insightful information.

Second, we analyzed the data to see if follows Zipf's distribution. Zipf's law states that only a small fraction of the words are used most of the time, the vast majority of the remaining are rarely used. Simply put, it says that r^{th} most frequent word or in other words, r is the frequency rank of the word has a frequency $f(r)$ which is represented can be represented by the Equation 4.5

$$f(r) \propto \frac{1}{f(r)} \tag{4.5}$$

4.4.3 Classification

In this section, we discuss the classification models used to train the available data.

Training and Testing Split. The data was split using a 9:1 ratio where nine parts are used for training the models and the one part is used for testing them. This ratio was used across all the models used to keep consistency when comparing performance results.

Logistic Regression

As discussed in Section 3.1, Logistic Regression makes use of sigmoid function [40] to calculate the probability of a data point x belonging to a class. This function takes in all the values and maps them between 0 and 1. The logistic function equation is given by Equation 4.6, where $P(x)$ represents the probability that the data variable x belongs to one of the classes. b_0 is the bias of intercept and b_1 is the coefficient of the single input variable. The probability of a point belonging to any other class is then $1 - P(x)$.

$$P(x) = \frac{e^{b_0+b_1 x}}{1 + e^{b_0+b_1 x}} \tag{4.6}$$

In this experiment, we use a simple Logistic Regression function to classify the layout and sequence of courses in a plan to predict the major they would most likely belong to. Results are presented in Section 5.2.2.

Support Vector Machines

Support Vector Machines (SVM) identify borders in n-dimensional data space in order to classify the data points (see Section 3.1). The kernel transforms the data from a low dimensional input space into a higher dimensional space to enable efficient classification of data. The selection of the kernel that will be used for the classification is most critical in determining the effectiveness of the model. The different kind of kernels are:

- *Linear Kernel*: uses a dot product of the data points to transform the data. We use a linear kernel when the data has enough dimensions to identify the boundaries and does not require transformation into a higher space.
- *Polynomial Kernel*: a generalized form of a linear kernel that can distinguish between both curved and non-linear input space.
- *Radial Basis Function*: maps input space into infinite dimensional space.

In our experiments, we use Linear Kernel approach on the course sequences, due to its relative simplicity and computational low-cost compared to the other kernels. Results are presented in Section 5.2.2.

4.5 Metrics for Evaluation

In this section we discuss the metrics used to evaluate the different methods implemented to develop the proposed goals.

Synthetic Data Generation and Data Labeling. One of the goals with using synthetic data is to be able to generate more and better distributed data to train ML models. However, for synthetic data to be useful, it is necessary to make sure it is reliable, i.e., how close it is to real data (as generated by humans). To build confidence in our synthetically generated data, first we compare its statistical metrics and distributions against those of the original data. Then, we investigate the behaviour and performance of different classifiers on different sets of synthetic data. Results from these experiments are discussed in Chapter 5.

Classification Metrics. Several statistical methods and metrics were used to measure the efficacy of different classification models used in our experiments, including: Confusion Matrix, Accuracy, Precision, Recall, and ROC curves. These metrics are defined based on the following concepts and a confusion matrix (see Figure 4.6 below) [89]:

		Predicted class		
		yes	*no*	Total
Actual class	*yes*	*TP*	*FN*	*P*
	no	*FP*	*TN*	*N*
	Total	P'	N'	$P + N$

Figure 4.6: A simple representation of a confusion matrix for a binary classification [5].

- True Positive (TP): The number positive observations that are correctly classified as positive.
- False Negative (FN): The number of positive observations falsely classified as negative.
- True Negative (TN): The number of negative observations correctly labeled as negative.
- False Positive (FP): The number of negative observations incorrectly classified as positive.
- False Positive Rate (FPR): The ratio between total number of false positives and all the negative cases (FP + TN).
- True Positive Rate (TPR): The ratio between total number of true positive and all the positive cases (TP + FN).

We can now define the metrics in detail:

- *Accuracy*: is calculated using the Formula 4.7 and it is the basic metric for evaluating an ML model. It indicates the proportion of correct predictions.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{4.7}$$

- *Precision*: is calculated based on the Formula 4.8 and it is particularly useful since we seek to minimize the FP count when it comes to study plan classification, e.g., an incorrect plan with a real rank of -1 should never be miss-classified with other ranks (i.e., 1, 2, 3, 4, 5).

$$Precision = \frac{TP}{TP + FP} \tag{4.8}$$

- *Recall*: can be evaluated using the equation 4.9, this is also known as a "completeness" metric, as it can be helpful when checking that study plans, say, with a rank of -1 are indeed -1. In our system, we aim for high *precision* and low *recall*.

$$Recall = \frac{TP}{TP + FN} \tag{4.9}$$

- ROC-AUC curve: The Receiver Operating Curves (ROC) are plotted with TPR on y-axis and FPR on the x-axis to capture the *precision* and *recall* of a classifier. The area under the curve (AUC) indicates the accuracy with which the classifier can predict the class of an observation, i.e., how well the classifier can distinguish between the different groups. There is a one-to-one correspondence with precision-recall and the ROC curve [90].

Chapter 5

EXPERIMENTS AND RESULTS

In this chapter we present and discuss results from experiments based on the methodology described in Chapter 4.

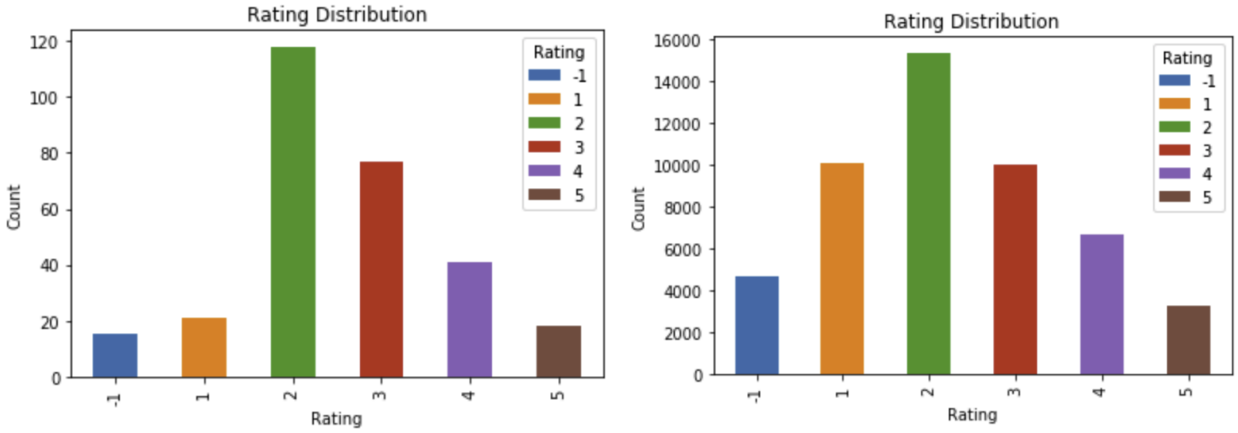
5.1 Recommendation of Study Plan

This section covers results from experiments related to the implementation of the study plan recommendation module as discussed in Section 4.3.

5.1.1 Validation of Synthetic Data

Figure 5.1 presents a side-by-side comparison of the distributions of the original data vs. the synthetic data in terms of plan ratings. This comparison is the first step to understand how the generated synthetic data behaves compared to real data. Compared to the original data, which didn't have many observations with low rating (e.g., -1, 1, 2), the synthetic data increases the number of such observations while maintaining the general distribution across the other rankings. The original data, which was generated by humans (EvCC faculty) had a higher number of plans with better rankings, which makes sense because these plans were carefully selected for our experiments, given that they were plans that were complete, clean, and mostly complied with scheduling requirements. In the synthetic data, we maintain the general distribution while increasing the number of plans which had a lower rating; as a result, the synthetic data has a more "normally-balanced" distribution.

Next, the mean and standard deviation of relevant features in human-generated data (from EvCC) are compared against the synthetic dataset. Results are summarized in Figures 5.2 and 5.3. Different sizes of synthetic datasets are used to get more insight into the synthetic data behavior. In Figure 5.3, we can see the that the mean of most of these features



(a) Distribution of Original Data

(b) Distribution of Synthetic Data

Figure 5.1: Comparison of Synthetic Data vs. Original Data distribution based on Rating.

is maintained, which is desirable to "simulate" realistic data. Then, from Figure 5.2, we can see that by using synthetic data, we can increase the variability of certain features to be more realistic, i.e., an increase in the standard deviation.There is a noticeable increase in the standard deviation for preferred number of quarters and the actual number of quarters, this is because the original data has this value set to a constant, but in an ideal scenario, the value would be expected to vary. On a different note, observe the drop in the mean value of $MathSequenceBreak$ and $EnglishSequenceBreak$ in Figure 5.3, this is due to the fact that the original data had more plans intended for "Mechanical Engineering", while the synthetic dataset includes various other majors with different requirements, so such features would be more outstanding in the original data.

5.1.2 Data Preprocessing

One of the most critical steps in preprocessing the data is to find the correlation between categorical and numerical data. Cramer's V correlation was used for this purpose. Figure 5.4 illustrates the correlation between the features in the feature set, where a darker

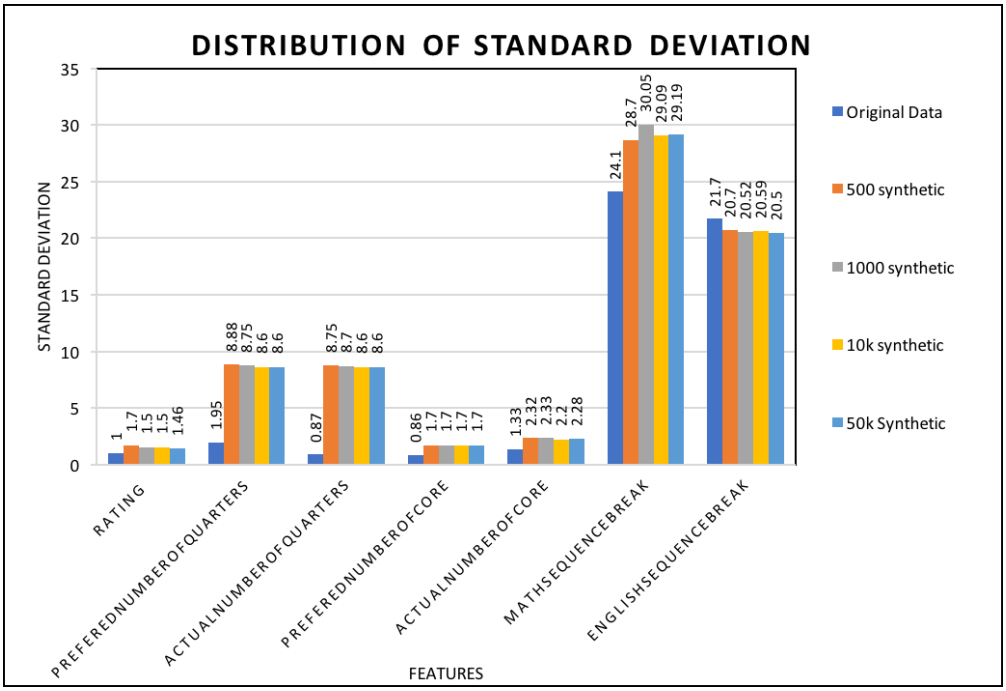


Figure 5.2: Distribution of Standard Deviation across relevant features.

color indicates lower correlation and lighter color indicates a higher correlation.

The figure shows that most of the features used for the classification are not highly correlated, suggesting that feature reduction need not be applied for our data. A couple of features that seem to show a high correlation are $PreferedMajor$ and $ActualStartQuarter$. The correlation between these features is not immediately intuitive, but could also be a result of available data being heavy in one major ("Mechanical Engineering"), where most of the students targeting that major are set to start, perhaps even by the same set of advisors, on the same quarter.

5.1.3 Grid Search with Cross Validation

Figure 5.5 shows the results for Grid Search (Section 4.3.4). Grid search was done to achieve parameter tuning to increase the performance of the model. A combination of original data

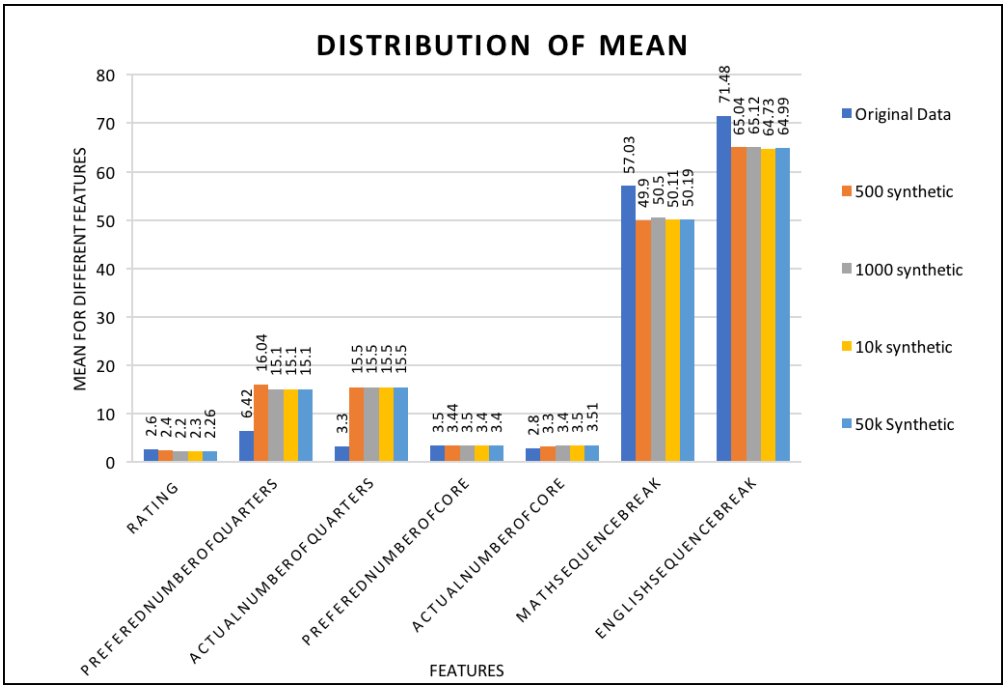


Figure 5.3: Distribution of Mean across relevant features.

and synthetic data was used to train the models and to perform the grid search. Each plot in Figure 5.5 shows the *accuracy* score of the model against the number of levels (for Decision Trees) or the number of estimators (for Random Forest and Extra Trees). The following observations have been made for different classifiers:

- **Decision Trees Classifier:** Best performing Decision Trees were searched with different combinations of measure of impurity and tree depth. Figure 5.5a illustrates the performance of the model against different tree depths. There was a significant increase in the *accuracy* of the model when the number of levels increased from 3 to 5, but it does not change much for tree depth 5 and 7. When the maximum depth is increased from 7 to 9, we find the best performing combination of levels and impurity index.

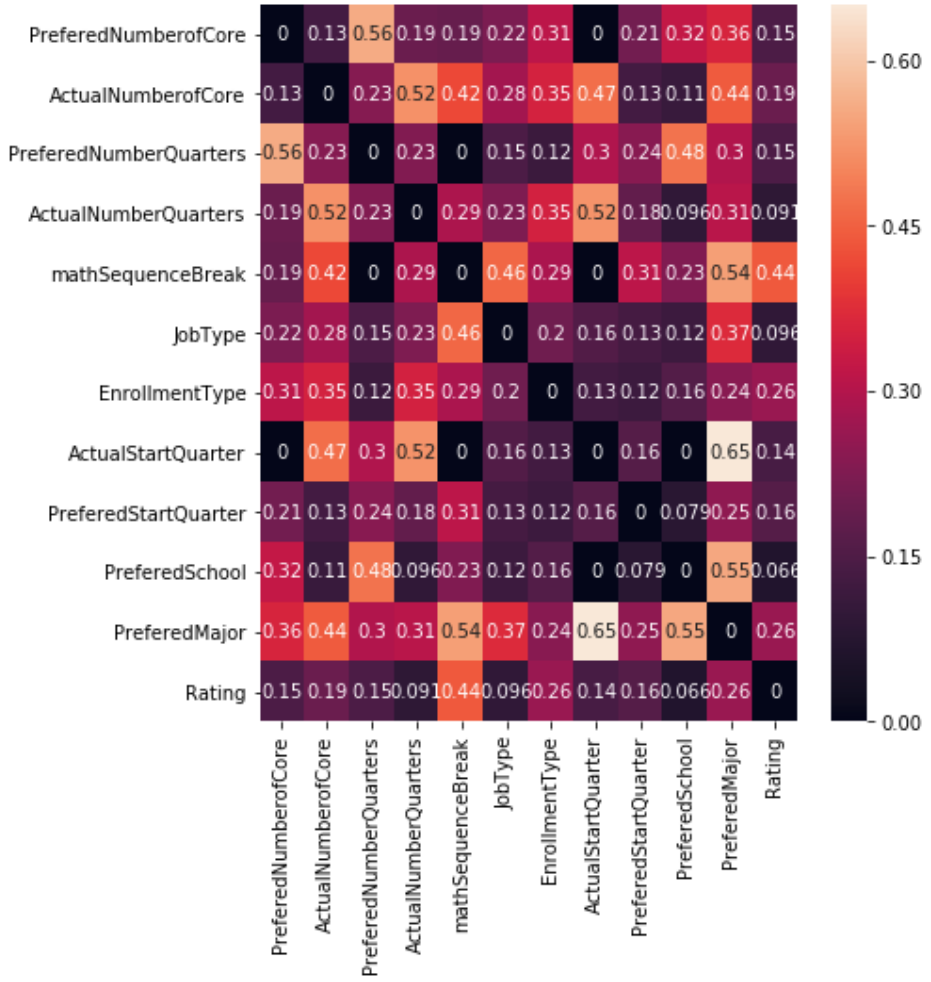
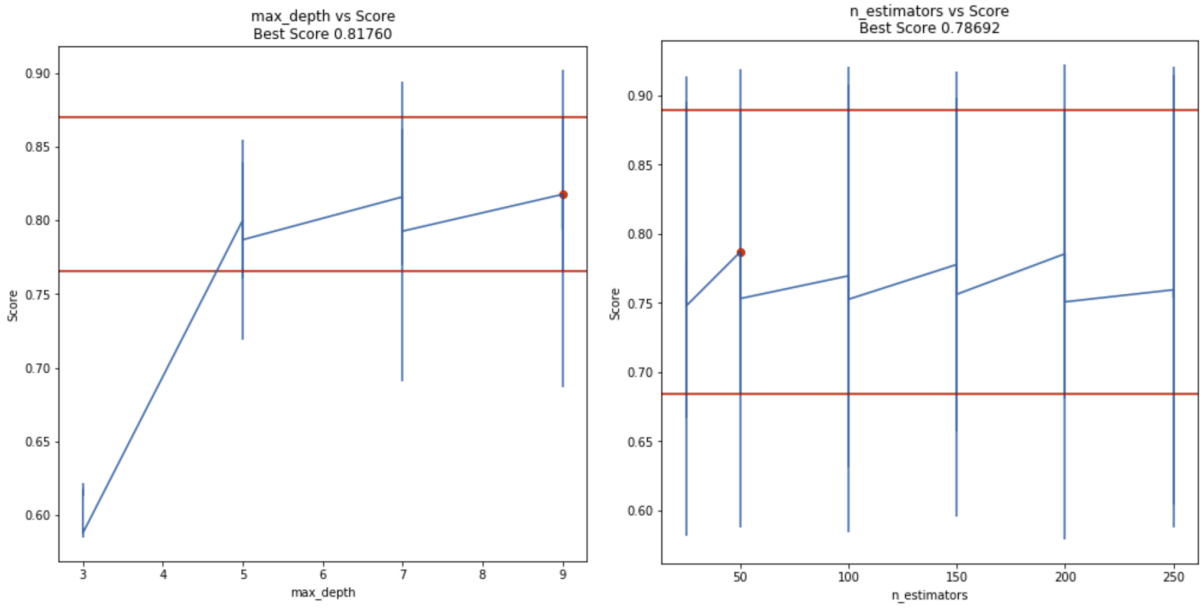


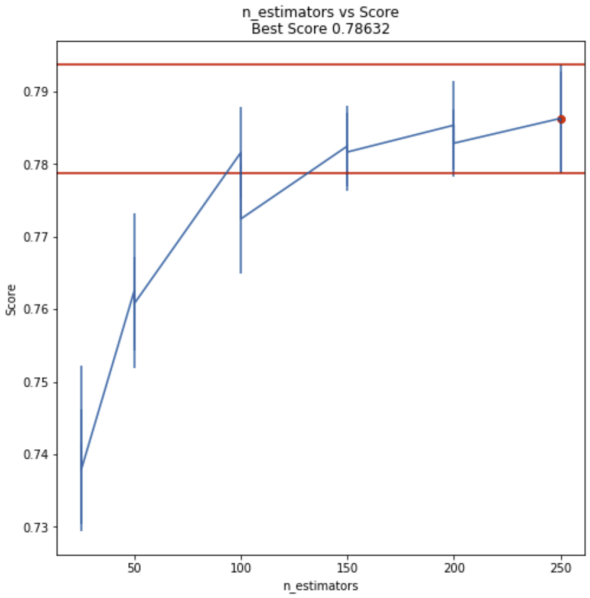
Figure 5.4: Correlation between all the features both categorical and continuous.

- **Random Forest Classifier:** The performance of Random Forest for both gini and entropy impurities was checked against a different number of estimators as seen in Figure 5.5b. Random Forest classifier performed best with 50 estimators.
- **Extra Tree Classifier:** Grid search was done for combinations of impurity metric and a different number of estimators. The increase in *accuracy* with the number of estimators was almost exponential in the case of Extra Trees.The best score was achieved with 250 estimators.



(a) Decision Tree

(b) Random Forest



(c) Extra Trees

Figure 5.5: Grid Search - Cross Validation Results for different classifiers.

5.1.4 Supervised Classification

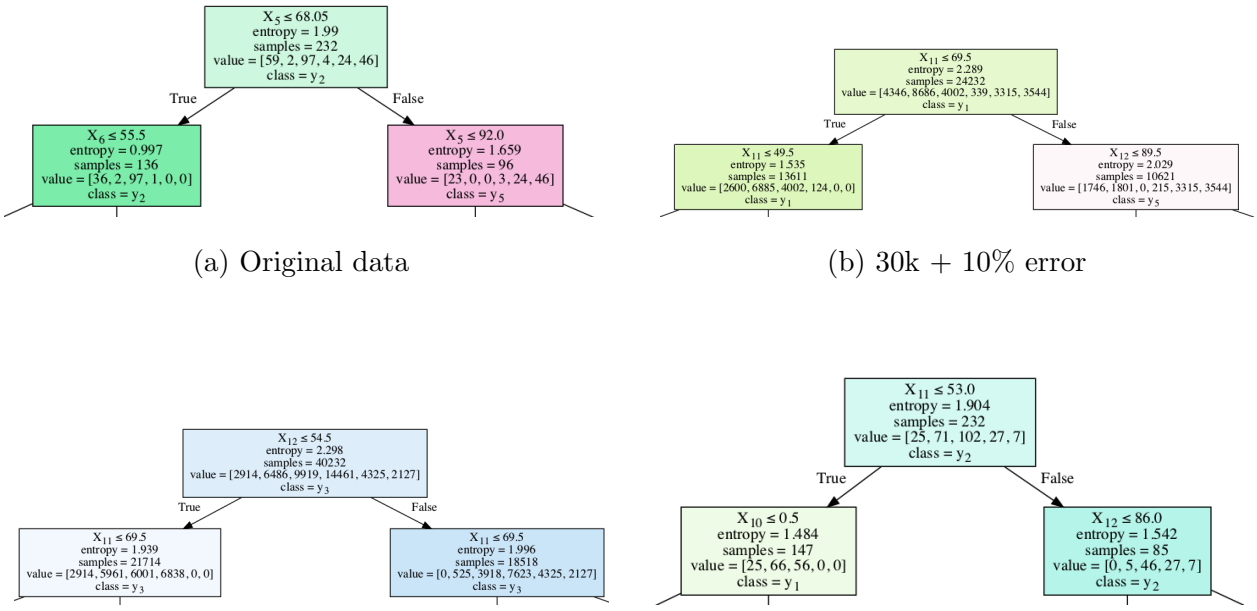
Several experiments were performed with different combinations of available data. The purpose of trying different data sets was to validate the synthetic data and analyze its performance against original data. Figure 5.6, summarizes various experiments performed to better understand the behavior of our dataset.

	ACCURACY	PRECISION	RECALL	F1-SCORE
Original Data (290)				
Decision Tree Classifier	0.74	0.71	0.74	0.7
Random Forest Classifier	0.65	0.71	0.66	0.67
Bagged Classifier	0.63	0.72	0.64	0.66
Extra Tree Classifier	0.67	0.73	0.67	0.69
30K Synthetic data + 10% error				
Decision Tree Classifier	0.62	0.52	0.63	0.56
Random Forest Classifier	0.65	0.61	0.65	0.59
Bagged Classifier	0.58	0.56	0.58	0.56
Extra Tree Classifier	0.64	0.6	0.65	0.58
50k + original + 20% error				
Decision Tree Classifier	0.75	0.8	0.76	0.75
Random Forest Classifier	0.75	0.79	0.75	0.74
Bagged Classifier	0.72	0.74	0.73	0.72
Extra Tree Classifier	0.62	0.68	0.68	0.67
50k + original + 10% error				
Decision Tree Classifier	0.83	0.84	0.83	0.83
Random Forest Classifier	0.83	0.84	0.83	0.83
Bagged Classifier	0.73	0.75	0.74	0.73
Extra Tree Classifier	0.78	0.79	0.78	0.78

Figure 5.6: Summary of the performance of different classifiers on various combinations of the data.

The performance of classifiers using the original data seems acceptable, however, could be due to over-fitting. Hence, it is necessary to introduce and test more, different data. The performance using only original data is similar to the performance using 50000 data

points plus the original data and 20% error induced. Recall that this error is induced in synthetic data to simulate certain human errors and bias (Section 4.3.2). Classifiers seem to perform best when trained using 50000 synthetic data points plus original data and 10% error induced. In terms of specific classifiers, Decision Trees perform consistently well, while Extra Trees and Bagged Classifier are the weakest, and Random Forest performance improves with larger datasets.



(a) Original data

(b) 30k + 10% error

(c) 50k + original + 20% error

(d) 50k + original + 10% error

Figure 5.7: The feature at root and the first level for different data sets.

Besides good performance, the Decision Tree model can also be used to identify features that are critical for classification. By observing features that appear at the root and first level of a good-performing Decision Tree model, we can gain insight into the importance of such features for discriminating across classes. Looking at Figure 5.7, we see that placed at the root of different Decision Tree models is either *EnglishSequenceBreak* or *MathSequenceBreak*, suggesting that these features are more important than others in determining the quality (rat-

ing) of a study plan. For instance, using 50000 plans with 20% error (Figure 5.7c), the root is *MathSequenceBreak*, and for the remaining 3 cases, the root is *EnglishSequenceBreak*. The reason might be that English is a subject common to all majors and all plans, and Math is more relevant only for certain majors. In the following sections we discuss in the detail the performance of different classifiers with 50000 data plans and 10% error.

Decision Tree Classifier

Figure 5.8 illustrates the confusion matrix for the Decision Trees model. Interesting observations can be drawn from the results; for instance, the overall *accuraccy* of the model is 83.44%. The classifier is performing well for all the classes with the exception of Classes 1 and 5. Class 1 has a high *precision* and low *recall*. The low *recall* in Class 1 is because of the high number of FN, i.e., the classifier cannot identify many instances of Class 1. This can also be seen in the ROC curve in Figure 5.9. The area under the curve is above 0.95 (excellent) for all the classes except Class 1, which has a higher FPR. Regarding Class 5, we observe that it has low *precision* and high *recall*. The low *precision* is due to a higher number of FP, i.e., observations, particularly from Class 4 are being miss-classified as Class 5. This problem could be due to the fact that there is little difference between consecutive classes, e.g., a very good plan could be rated as 4 for minimal reasons.

	-1	1	2	3	4	5	Recall
-1	784	67	0	0	0	0	0.92
1	110	1766	557	196	123	81	0.62
2	0	139	2638	0	0	0	0.95
3	0	16	70	1710	38	0	0.93
4	0	7	0	33	1031	136	0.85
5	0	9	0	0	85	462	0.83
Precision	0.88	0.88	0.81	0.88	0.81	0.68	0.834

Confusion Matrix

Figure 5.8: Confusion Matrix for Decision Tree Classifier.

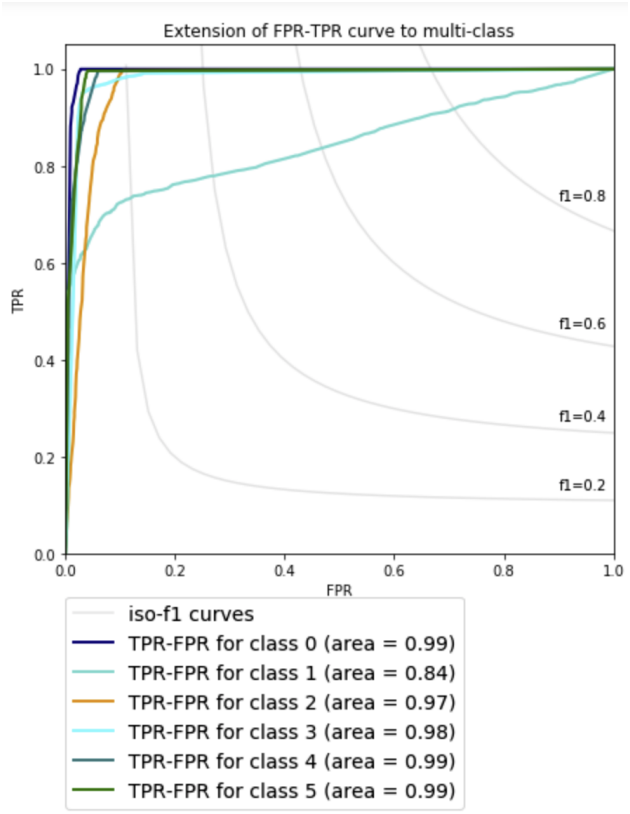


Figure 5.9: ROC-AUC curve for Decision Tree. Class 0 represents rank of -1, and subsequent classes represent rankings 1 through 5 correspondingly.

Random Forest Classifier

The confusion matrix for Random Forest classifier is presented in Figure 5.10. This model had an overall *accuraccy* of 83.4%, close to the *accuraccy* of Decision Tree. Random Forests select a subset of features randomly for doing the split at each node, which might be the reason it does not perform better than Decision Trees (which is an ideal observation). Recall from Figure 5.7, that *MathSeqBreak* and *EnglishSeqBreak* are the critical features for classifying the plans. If random selection does not consider these features, it could bringing down the overall *accuraccy* of the classifier. The confusion matrix in Figure 5.10, shows that Classes 1 and 4 behave differently to other classes.

Predicted Rating

Actual Rating	-1	1	2	3	4	5	Recall
-1	776	104	0	0	0	0	0.88
1	86	1790	517	178	169	58	0.66
2	0	153	2633	0	0	0	0.95
3	0	1	82	1628	67	0	0.92
4	0	1	0	10	1109	105	0.91
5	0	1	0	0	153	437	0.74
Precision	0.90	0.87	0.81	0.90	0.74	0.73	0.834

Figure 5.10: Confusion Matrix for Random Forest Classifier.

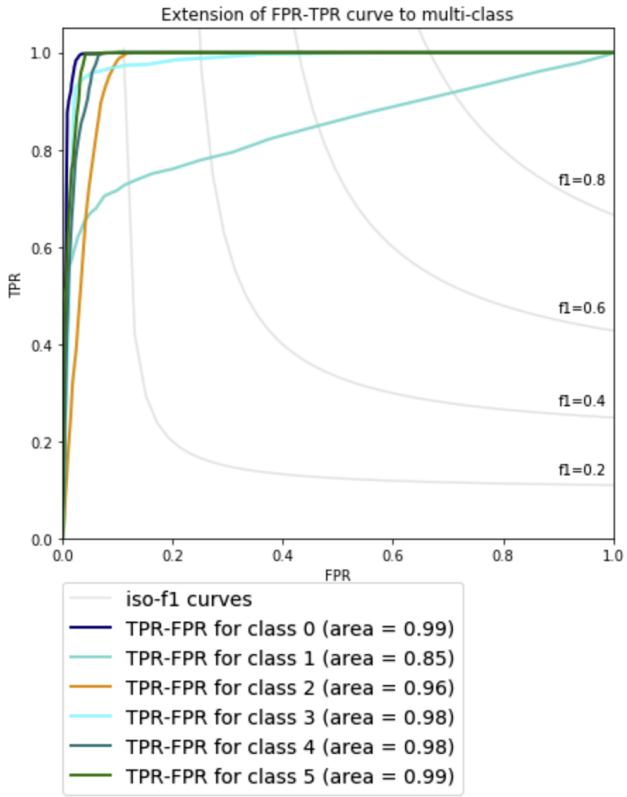


Figure 5.11: ROC curve for Random Forest where Class 0 represents rating of -1.

Similar to Decision Trees, these exceptions could be because of the small difference between consecutive-rank classes. This can also be seen in the corresponding ROC curves (Figure 5.11), which also happens to be very similar to the ROC curves for Decision Tree (see Figure 5.9). The area under the curve for Class 1 is again the lowest compared to other classes.

Extra Tree Classifier

Figure 5.12 contains the confusion matrix for the Extra Tree Classifier. Though Extra Trees work very similar to Random Forest, this model has an overall *accuraccy* of 78.2% – lower than Decision Tree and Random Forest models. This could be because Extra Trees work better for data with more numerical than categorical features [79].

		Predicted Rating						
		-1	1	2	3	4	5	Recall
Actual Rating	-1	684	196	0	0	0	0	0.78
	1	79	1801	517	197	154	50	0.64
	2	0	289	2450	47	0	0	0.88
	3	0	6	183	1509	80	0	0.85
	4	0	3	0	106	1039	77	0.85
	5	0	2	0	0	203	386	0.65
	Precision	0.90	0.78	0.78	0.81	0.70	0.75	0.782

Figure 5.12: Confusion Matrix for Extra Trees Classifier.

Furthermore, note that *precision* is inversely proportional to *recall*. This could be because Extra Trees classifier is an ensemble method which tries to improve the *accuracy* of the classifier by randomly selecting the split at a node instead of selecting the best split. This process is intended to increase the *accuracy* of the model, i.e., increasing the count of both TP and FP while decreasing the FN count. When the *precision* of a class is low, the FN count is higher, resulting in lower *recall*. The increase in FN, added to the fact that there

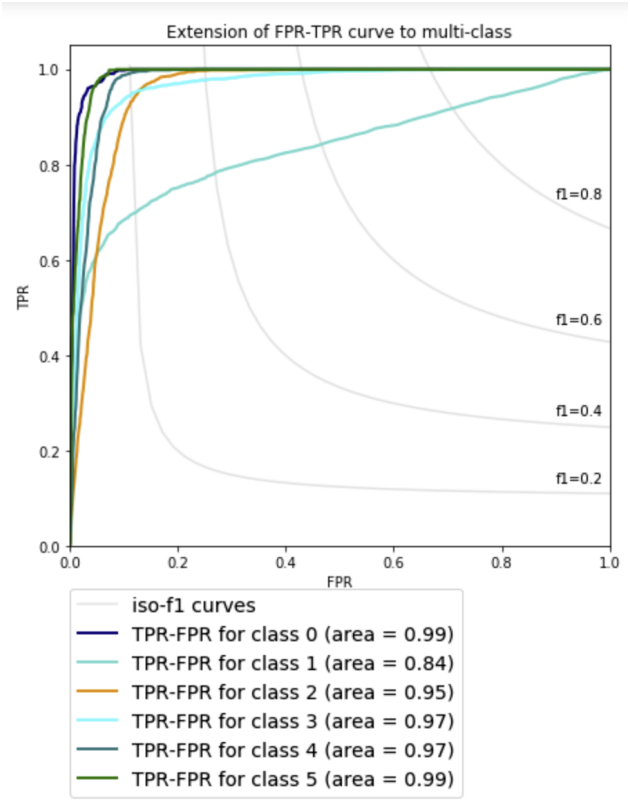


Figure 5.13: ROC curve for Extra Classification. Class 0 represents rating of -1, and subsequent classes represent ratings 1 through 5 correspondingly.

is little difference between consecutive-rank classes, results in an overall low *accuraccy*. The corresponding ROC curves in Figure 5.13 confirm these observations. Class 1 has the lowest area under the curve as it has higher FN.

Bagged Classifier

In Figure 5.14, we can see the confusion matrix for Bagged Classifier. This model has an overall *accuracy* of 73.7%, making it the lowest performing classifier. Bagged Classifiers work best on a weak base classifier with a lot variance. The poor performance of this classifier could be because the base classifier (Decision Tree) is stable and does not have much variance.

Therefore, bagging the Decision Tree does not yield better performance.

		Predicted Rating						
		-1	1	2	3	4	5	Recall
Actual Rating	-1	688	50	0	32	0	0	0.89
	1	39	1281	151	79	0	0	0.83
	2	0	111	2250	116	0	0	0.91
	3	189	405	659	2005	254	114	0.55
	4	0	0	0	177	846	90	0.76
	5	0	2	0	70	100	352	0.67
	Precision	0.75	0.69	0.74	0.81	0.70	0.63	0.737

Figure 5.14: Confusion Matrix for Bagged Classifier.

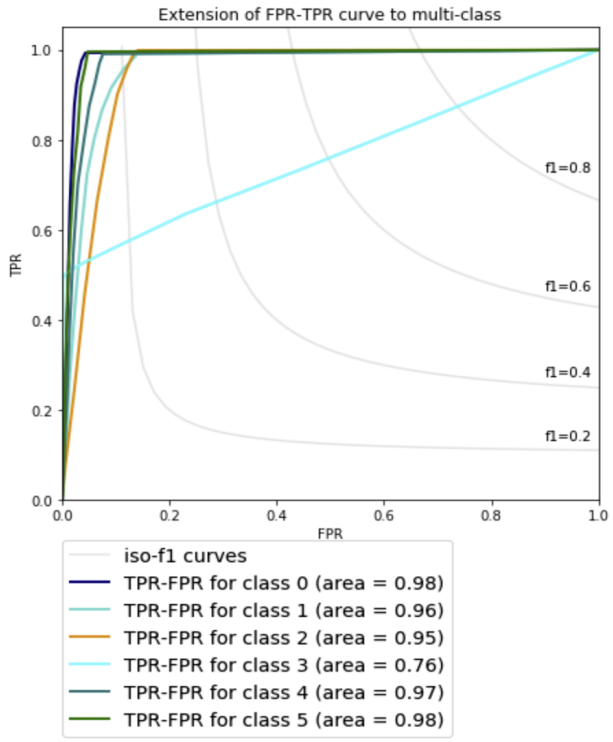


Figure 5.15: ROC curve for Bagged Classifier. Class 0 represents rating of -1, and subsequent classes represent ratings 1 through 5 correspondingly.

Precision is low overall across all classes – best performing for Class 3. *Recall* seems to be inversely proportional to *precision*, which could be because with bagged classification, the multiple classifiers try to adjust the threshold for different features to improve the *accuracy* of the system. When adjusting the thresholds, more cases are classified as positive increasing both the TP and FP for the classifier. The number of negative classes is also bought down, i.e., a drop in FN. With the increase in TP, the *accuracy* of the system improves, but with the increase in FP, *Precision* decreases. The decrease in FN improves the *recall* of the system. These results can also be seen in the ROC curves in Figure 5.15.

5.2 Recommendation of Major

In this section we cover the experiments and results related to the implementation of the Major Prediction module as discussed in Section 4.4. Note that in this module, the classes refer to the college Major for a student.

5.2.1 Data Analysis

As discussed in Section 4.4.2, we use Zipf's law to analyze the distribution of the courses w.r.t. the frequency. Zipf's law can be explained using a word cloud. Word-cloud indicates the "top" courses for a given major, and here the Word-cloud has been generated for "Mechanical Engineering" (see Figure 5.17).

This analysis provides insight into the courses that cannot be used for distinguishing between different majors. For instance in English language, "the" is the most frequently occurring word, but it does not help with text analysis. Similarly, the most common courses across STEM, "Math" and "English", but these words might not be useful to differentiate between majors.

5.2.2 Sequence Classification

As discussed in Section 4.4.3, Logistic Regression and SVM were used to train the data for classifying sequences of courses in study plans.

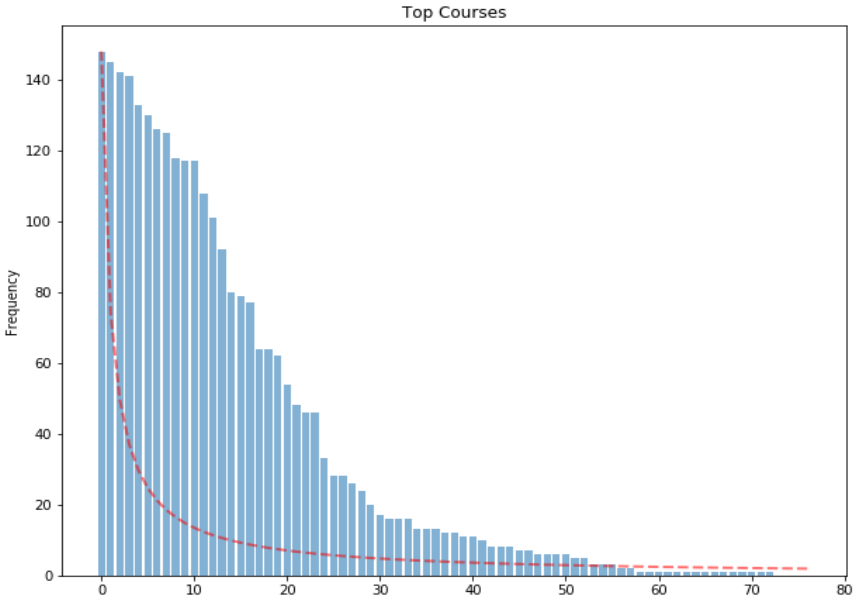


Figure 5.16: The course distribution w.r.t. the frequency of the course.

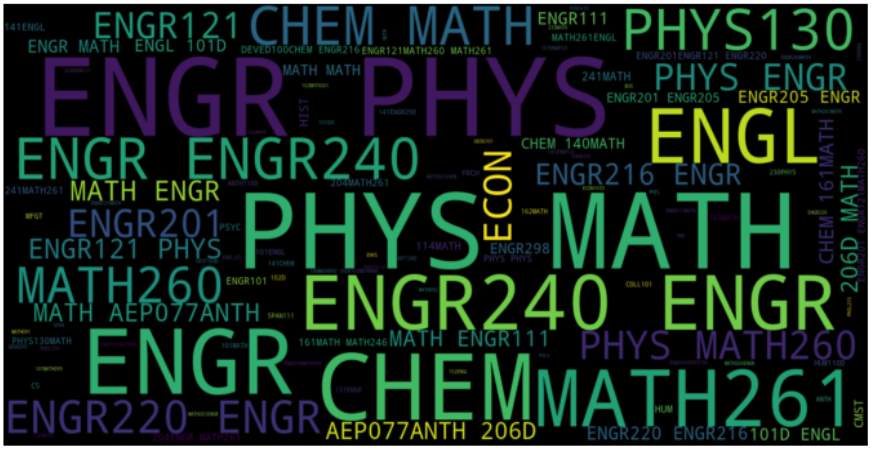


Figure 5.17: Word cloud representing the most frequently taken courses for "Mechanical Engineering" major.

Logistic Regression

The confusion matrix for Logistic Regression is presented in Figure 5.18, showing that this model achieves an overall *accuracy* of 90% making Logistic Regression better than SVM.

0	Aeronautical Eng.	5	Electrical Eng.
1	Business	6	General Major
2	Chemical Eng.	7	Graphic Design
3	Civil Eng.	8	Mechanical Eng.
4	Computer Sc.	9	Nursing
		10	Pharmacy

Actual Major

	0	1	2	3	4	5	6	7	8	9	10	Recall
0	1	0	0	0	0	0	0	0	0	0	0	1
1	0	1	0	0	0	0	0	0	0	0	0	1
2	0	0	1	0	0	0	0	0	0	0	0	1
3	0	0	0	2	0	0	0	0	0	0	0	1
4	0	0	0	0	6	0	0	0	0	0	0	1
5	0	0	0	0	0	4	0	0	1	0	0	0.80
6	0	0	0	0	1	1	4	0	0	0	0	0.67
7	0	0	0	0	0	0	0	1	0	0	0	1
8	0	0	0	0	0	0	1	0	29	0	0	0.97
9	0	0	0	0	0	0	0	0	0	1	0	1
10	0	0	0	0	0	0	0	0	0	0	1	1
Precision	1	1	1	1	0.86	0.80	0.80	1	0.97	1	1	90

Figure 5.18: Confusion Matrix for Logistic Regression. The exact label value of each class can be seen in the figure.

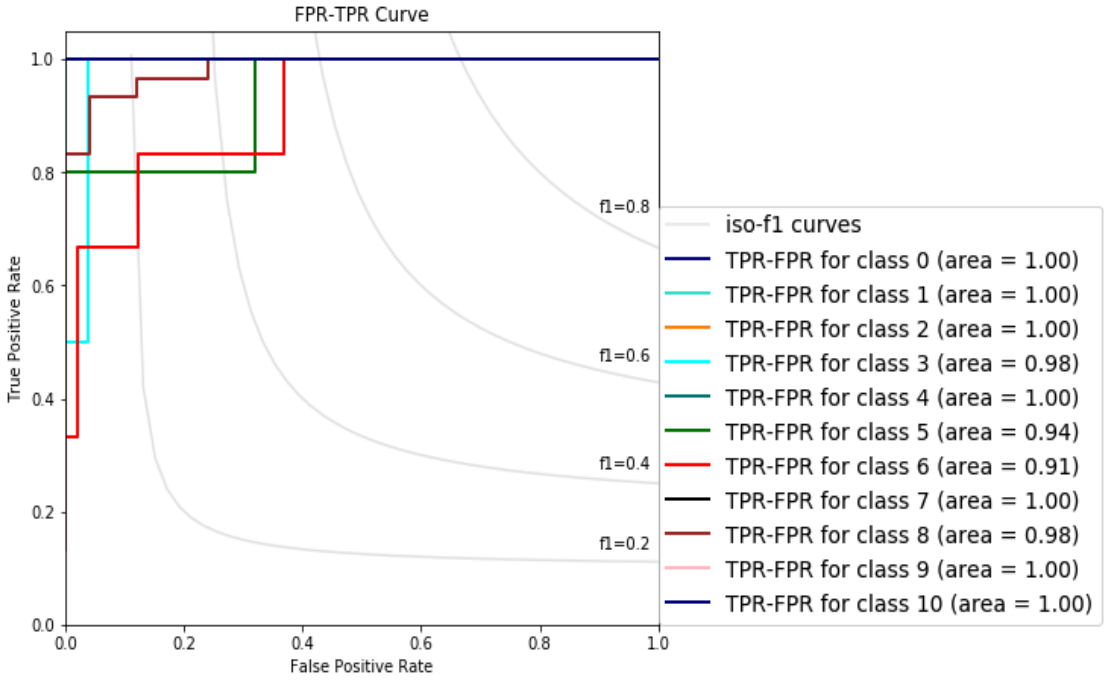


Figure 5.19: ROC-AUC for Logistic Regression.

From the confusion matrix, we can observe that *precision* and *recall* are 1 for most classes. The most interesting observations from the confusion matrix are for Class 6, "General

Major". Class 6 has high *precision* and low *recall*. The high *precision* and low *recall* could be because of the nature of the coursework for this major, which includes subjects from all the other majors. The corresponding ROC curves in Figure 5.19 show that the area under the curve is smallest for Class 6, indicating that it is harder to identify a course plan designed for "General Major". Also notice the "plateaus" in the ROC curves, these may be due to the size of the dataset and number of observations for each class. The curves can be more smooth with a larger, varied data set

Support Vector Machines

The confusion matrix for SVM is presented in Figure 5.20. The overall *accuracy* of the classifier is 88.7%, which is lower than the *accuracy* of Logistic Regression, and might be due to the the size of the dataset.

0	Aeronautical Eng.	5	Electrical Eng.
1	Business	6	General Major
2	Chemical Eng.	7	Graphic Design
3	Civil Eng.	8	Mechanical Eng.
4	Computer Sc.	9	Nursing
		10	Pharmacy

Actual Major

	0	1	2	3	4	5	6	7	8	9	10	Recall
0	2	0	0	0	0	0	0	0	0	0	0	1
1	0	4	0	0	0	0	0	0	0	0	0	1
2	0	0	2	0	0	0	0	0	0	0	0	1
3	0	0	0	2	0	0	0	0	5	0	0	0.29
4	0	0	0	0	14	0	0	0	1	0	0	0.93
5	0	0	0	0	1	10	0	0	3	0	0	0.71
6	0	0	0	0	1	1	12	0	1	0	0	0.80
7	0	0	0	0	0	0	0	2	0	0	0	1
8	0	0	0	1	0	0	2	0	74	0	0	0.96
9	0	0	0	0	0	0	0	0	0	2	4	1
10	0	0	0	0	0	0	0	0	0	0	1	1
Precision	1	1	1	0.67	0.88	0.91	0.86	1	0.88	1	1	88.74

Figure 5.20: Confusion Matrix for Support Vector Machines.The exact label value of each class can be seen in the figure.

The SVM was trained on a small data set which has high class imbalance. This could make it hard for the classifier to identify the boundaries properly. We can also see that *precision* and *recall* are rather low for Class 3, "Civil Engineering". Our dataset did not have many observations for Class 3, and so, they are also being overshadowed by Class 8 "Mechanical Engineering". The performance of Class 3 can be observed from the ROC curves

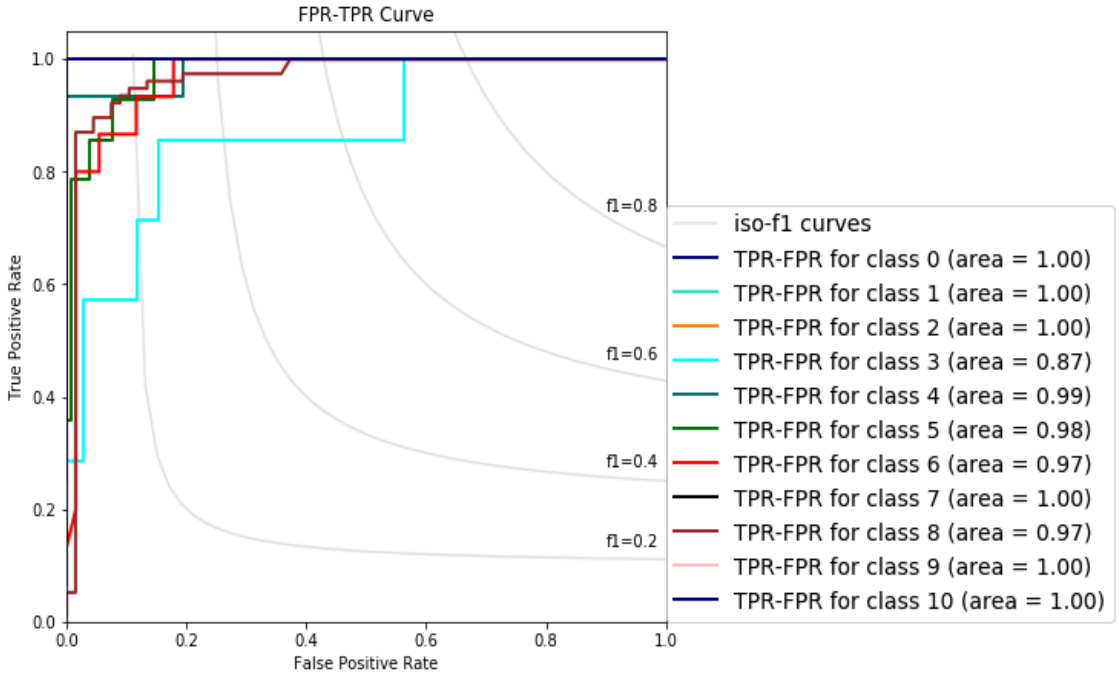


Figure 5.21: ROC-AUC for Support Vector Machines.

in Figure 5.21, where Class 3 has the smallest area under the curve. The "plateaus" in the curves could be because there are not enough observations to identify proper boundaries.

Chapter 6

CONCLUSION AND FUTURE WORK

In this section, we discuss our research achievements. We then discuss limitations and challenges encountered during our work. Finally, discuss the ways the research can be extended in the future.

6.1 Conclusion

6.1.1 Recommendation of Study Plans

In this section we talk about the conclusions we can draw from the results of the methods implemented to generate recommendations for study plans (Sections 4.3 and 5.1).

Generating Synthetic Data. Using the study plans provided by EvCC to understand the nature of the data, we developed a script to generate synthetic data. We also designed an algorithm to label the synthetic data to facilitate supervised classification. The synthetic data is used to capture a broader range of instances to train different classifiers. Large volumes of data can be produced, which allows better study of classifiers performance as the VAA system scales. Moreover, by having synthetic data, research in this area is decoupled from the need of having large volumes of original data.

Data Preprocessing. Correlation between the different features considered was analyzed to perform feature reduction. Cramer's V correlation method was used to calculate the correlation between the different categorical and numerical data. We found that there is no high correlation between most existing features. Furthermore, grid search for parameter tuning was performed on various combinations of training metrics for all the classifiers. This

process was critical in identifying the parameters to improve the accuracy of classifiers.

Classification. Multiple classification models have been trained with different sets of data. This provides confidence in the synthetic data that was generated. The results can be observed in the Figure 5.6. We have seen how the classifiers can scale based on the results observed on the different sizes of data set. We also found that decision trees performed consistently better with different datasets. Different ensemble methods are not efficient in improving the performance of the decision tree classifier. This shows that the decision trees is the best classifier to use to train data, which is a combination of both categorical and numerical. The performance of decision trees and random forests improved with an increase in the size of the dataset. The decision trees had TPR higher than FPR for most of the classes.

Recommendation Module. A recommendation module was implemented based on the concept of distance based Collaborative Filtering for generating study plan recommendations based on student's preferences. The distance-based method was an effective approach to generate recommendations suitable for given input parameters. This module makes use of the classification results in finding the most relevant and appropriate plans. Moreover, the recommendation module has been developed as a RESTful API; this way, it is easy to integrate the module in the business layer with the rest of the VAA system. The input is sent to the API as a JSON string can be modified to add or remove input parameters. The API makes the system both scalable and modular.

6.1.2 Predicting Major Module

In this section, we discuss the conclusions from the experiments performed and the results observed for predicting a college major based on the course work designated for a student (see Sections 4.4.3 and 5.2 for detailed descriptions).

Data Preprocessing. Data preprocessing was done to convert the sequence of courses into feature vectors that can be used for training the classifiers. TF-IDF vectorizer is preferred over count vectorizer as the TF-IDF vectorizer assigns weights to the frequency of the words. Count Vectorizer keeps track of the frequency of the words and can be used for data analysis. The data follows Zipf's distribution, indicating it supports the principles of natural language, and further research can take place on the data.

Classification. Logistic Regression and Support vector models have been trained using available data. Logistic regression performs better with higher *accuracy* of 90%. SVM, on the other hand, performs reasonably well with an *accuracy* of 88%. Logistic Regression has higher *precision* and *recall* than SVM, indicating that the Logistic Classifier can better distinguish between classes than SVM. We used ROC curves to analyze the performance of the different classifiers and found that the area under the curve is close to 1 for all the classes. Furthermore, we can understand that the models require more varied data to smooth out the curves. This experiment shows that we can further extend the research to add more features to classify study plans to suggest a major based on the courses included in a study plan.

6.2 Limitations and Challenges

The main challenges faced in the research were related to the dataset. The data was limited in terms of the size and the information it provides. Additionally, the database schema has not been fully defined to support efficient ML approaches. The original data provided by EvCC has to be translated from HTML format, making it harder to push into the existing database directly. Moreover, the plans had to be sanitized and manually labeled. A reasonable amount of time has been spent in creating tools to help with data collection, which will need to be updated in the future.

The amount of original data was insufficient to train ML models because it lacked variability and had a significant class imbalance. To fix this issue, data had to be synthetically generated to train and test different machine learning methods. Generating synthetic data

to resemble real data was challenging and required careful analysis. This process has a high risk of being over-fitted, so we have included error induction in the algorithms to simulate some forms of human error.

It is also challenging to work with a system that is still under development, where many design decisions have to be made and with dependencies on various team members. For instance, different existing modules have been developed on different platforms. A communication method and pattern had to be established to make the integration more efficient. Also, reading data and adding new features takes more time and effort.

6.3 Future Work

The recommendation module developed in this work has to be integrated with the rest of the VAA system. The VAA system can be made more scalable and modular by considering some architecture design changes, for instance, localizing particular database and ML operations.

As an alternative to supervised classification, unsupervised classification methods like clustering can be explored. Clustering may improve the speed of the recommendation module and reduce the dependency on labeled data for training. As with any recommendation system, it is important to consider the security aspects of the system. Student's information is being used to generate recommendations, and it could be misused, this needs to be taken into account as the system scales or goes into deployment.

Future work involves gathering a wider range of data to enable more efficient data analysis and exploration. With diverse data, the pain points of the users can be identified and addressed, for instance, data that extends beyond STEM majors, special courses for transitional students, time preference for a student, etc. In terms of ML, knowledge graphs can be developed. Knowledge graphs help in entity recognition and handling the metadata more efficiently. Knowledge graphs would make it easy to identify critical information about the academic plans or any other service that could be added to the system.

Additional features can be used to perform sequence classification. For instance, map the student's performance along with the course work to predict the major. Apart from that,

Deep Learning methods like Bayesian networks and LSTM neural networks can experiment for sequence prediction.

BIBLIOGRAPHY

[1] Engineering Transfer, March 2015.

[2] Gates Bryant. The Evolution of Planning and Advising in Higher Education. *Part 2: Supplier Landscape*, March 2019.

[3] Rashi Goyal. Building a Machine Learning Based Recommendation Engine for the Virtual Academic Advisor System. June 2018.

[4] Theano Development Team. Theano: A python framework for fast computation of mathematical expressions. November 2017.

[5] Jiawei Han, Micheline Kamber, and Jian Pei. Classification : Basic Concepts. In *Data Mining - Concepts and Technologies*, volume 3, chapter 8, pages 113 – 114. Morgan Kaufmann Publishers, 2012.

[6] Larry Orimoloye. Are ensemble classifiers always better than single classifiers? March 2017.

[7] Jiawei Han, Micheline Kamber, and Jian Pei. Classification : Advanced Methods. In *Data Mining - Concepts and Technologies*, volume 3, chapter 9, pages 393 – 415. Morgan Kaufmann Publishers, 2012.

[8] Sanket Doshi. Brief on Recommender Systems. February 2010.

[9] Mathematics Associate in Arts Sciences Direct Transfer (DTA). "url :https://www.everettcc.edu/files/programs/mathematics.pdf", March 2018.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau,

M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[11] Jennifer Varney. Intrusive Advising. September 2007.

[12] Gates Bryant. The Supplier Landscape. *Driving Towards Degree - The Evolution of Planning and Education in Higher Education*, March 2015.

[13] Yi (Leaf) Zhang. An Overlooked Population in Community College: International Students (In)Validation Experiences With Academic Advising. *Community College Review*, 44(2):153–170, April 2016.

[14] George D. Kuh, Stanley O. Ikenberry, Natasha Jankowski, Timothy Reese Cain, Ewell, Pat Hutchings, Jillian Kinzie, and Timothy Reese Cain. *Using Evidence of Student Learning to Improve Higher Education*. December 2014.

[15] Jane E. Pizzolato. Complex Partnerships: Self-authorship and Provocative Academic-Advising Practices. *NACADA Journal*, 26(1):32–45, March 2006.

[16] Dianne T. Castor. Challenges for Two-Year College Advisors, June 2005.

[17] Alan Seidman. The Evaluation of a Pre/Post Admissions/Counseling Process at a Suburban Community College: Impact on Student Satisfaction with the Faculty and the Institution, Retention, and Academic Performance. *College and University*, 66(4):223–32, 1991.

[18] Margaret Steele. The Challenges of Advising Science, Technology, Engineering and Mathematics Students, September 2008.

[19] Ronald C. McArthur. Faculty-Based Advising: An Important Factor in Community College Retention. *Community College Review*, 32(4):1–18, March 2005.

[20] Thomas J. Grites, Marsha A. Miller, and Julie Givans Voler. *Beyond Foundations: Developing as a Master Academic Advisor*. John Wiley & Sons, September 2016.

[21] B. Beaudin and J. Breiner. Academic Advising at a Distance: Student Communication Preferences. *Int. J. Continuing Engineering Education & Lifelong Learning*, 11(1-2):128–134, July 2001.

[22] Pamela M. Golubski. Online Academic Advising. *Encyclopedia of Information Technology Curriculum Integration*, February 2008.

[23] E. Vance Wilson. A Standards Framework for Academic e-Advising Services. *International Journal of Services and Standards*, 2004.

[24] G Kena, S Aud, F Johnson, X Wang, J Zhang, A Rathbun, S Wilkinson-Flicker, and P Kristapovich. The Condition of Education 2014. page 256, May 2016.

[25] Erica MacKellar. Performance-Based Budgeting in the States. September 2016.

[26] Nicholas Hillman. Why Performance-Based College Funding Doesn't Work. *The Century Foundation*, May 2016.

[27] Nathalie Mainland. Introducing Salesforce Advisor Link, A New Salesforce.org Solution Designed for the Art of Advising. December 2017.

[28] Krista Carver. Student Success Best Practices: The Perfect Combination. "url = https://www.avisoretention.com/why-aviso-retention", May 2019.

[29] Southwest Tennessee Community College: Texting to Increase Retention and Enrollment. "url = https://www.signalvine.com/case-study/southwest-tennessee-community-college-texting-to-increase-retention-and-enrollment", 2018.

[30] Optimize : Plan for Future Demand Term by Term, 2019.

[31] Heng Luo, Changyong Niu, Ruimin Shen, and Carsten Ullrich. A Collaborative Filtering Framework based on both Local User Similarity and Global User Similarity. *Machine Learning*, 72(3):231–245, September 2008.

[32] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *ACM Conference on Computer Supported Cooperative Work*, pages 175–186. ACM, 1994.

[33] Andriy Shepitsen, Jonathan Gemmell, Bamshad Mobasher, and Robin Burke. Personalized Recommendation in Social Tagging Systems Using Hierarchical Clustering. In *ACM Conference on Recommender Systems*, pages 259–266. ACM, 2008.

[34] Diane Neal. Folksonomies: Introduction: Folksonomies and Image Tagging: Seeing the future? *Bulletin of the American Society for Information Science and Technology*, September 2008.

[35] Charu C. Aggarwal and Mohammad Al Hasan. Discrete Sequence Classification. In *Data Classification*, chapter 14. Chapman and Hall/CRC, 2015.

[36] Ngoc Giang Nguyen, Vu Anh Tran, Duc Luu Ngo, Dau Phan, Favorisen, Rosyking Lumbanraja, Mohammad Reza Faisal, Bahriddin Abapihi, Mamoru Kubo, and Kenji Satou. DNA Sequence Classification by Convolutional Neural Network. *Journal of Biomedical Science and Engineering*, pages 280–286, 2016.

[37] Bo Pang and Lillian Lee. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04. Association for Computational Linguistics, 2004.

[38] Terran Lane and Carla E. Brodley. An Application of Machine Learning to Anomaly Detection. February 1997.

[39] K.B. Kotsiantis. Supervised Machine Learning : A Review of Classification Techiniques. *Emerging Artificial Intelligence in Computer Engineering*, 160:3–21, 2007.

[40] Pratheek Joshi. Logisitic Regression- How it Works. In *Python Machine Learning Cookbook*, volume 2. Packt Publishing, 2019.

[41] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Decision trees. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[42] Michael J. Pazzani and Daniel Billsus. Content-Based Recommendation Systems. *The Adaptive Web*, pages 325 – 341, 2007.

[43] Katrien Verbert, Erik Duval, and Denis Lindstaedt, Stefanie N.and Gillet. Context-aware Recommender Systems. *Journal of Universal Computer Science*, October 2010.

[44] Jason Brownlee. Making Predictions with Sequences. *Long Short-Term Memory Networks*, September 2017.

[45] Zhengzheng Xing, Jian Pei, and Eamonn Keogh. A Brief Survey on Sequence Classification. *SIGKDD Explor. Newsl.*, 12(1):40–48, November 2010.

[46] C Matthew MacKenzie, Kenneth Laskey, Francis Mccabe, Peter F. Brown, and Rebekah Metz. Reference Model for Service Oriented Architecture 1.0. *Public Rev. Draft*, 2, August 2006.

[47] Kishor Wagh and Ravindra C Thool. A Comparative Study of SOAP Vs REST Web Services Provisioning Techniques for Mobile Host. *Journal of Information Engineering and Applications*, (5), 2012.

[48] Michael Sansoterra. Implement a Service-Oriented Architecture with REST APIs. October 2014.

[49] Yuan Meng. The SOA, the api and "REST" of the api. June 2017.

[50] Takeshi Yamada and Ryohei Nakano. Job Shop scheduling. *IEE Control Engineering Series*, pages 134–160, 1997.

[51] Getting started with Anaconda. url = http://docs.anaconda.com/anaconda/user-guide/getting-started/., 2019.

[52] Python Software Foundation. What is Python? Executive Summary. 2001 - 2019.

[53] W. N. Venables, D. M. Smith, and R Core Team. An Introduction to R. *Notes on R: A Programming Environment for Data Analysis and Graphics*, 3.6, April 2019.

[54] Karlijn Willems. Choosing R or Python for Data Analysis? An Infographic. May 2015.

[55] Oliphant Travis E. A guide to NumPy. 2006.

[56] Wes McKinney and PyData Development Team. pandas: Powerful Python data analysis toolkit. March 2019.

[57] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–.

[58] John Hunter, Darren Dale, Eric Firing, Michael Droettboom, and Matplotlib Development Team. Matplotlib documentation. February 2019.

[59] Ben Keen. Generatedata.com. url = https://www.generatedata.com/.

[60] Mark Brocato. Mockaroo. url = https://www.mockaroo.com/, 2019.

[61] Faker 1.0.2 documentation. url = https://faker.readthedocs.io/en/latest/, 2014.

[62] Tirthajyoti Sarkar. Random database/dataframe generator. March 2018.

[63] V. N. Garla and C. Brandt. Ontology-guided Feature Engineering for Clinical Text Classification. *Journal of Biomedical Informatics*, (5):992–998, October 2012.

[64] Frank Hofmann. Levenshtein Distance and Text Similarity in Python. January 2018.

[65] Jiawei Han, Micheline Kamber, and Jian Pei. Data preprocessing. In *Data Mining,The Morgan Kaufmann Series in Data Management Systems*, volume 3, chapter 3, pages 83–124. 2012.

[66] Wes McKinney. pandas: a Foundational Python Library for Data Analysis and Statistics. 2011.

[67] Daniel A Newman. Missing data: Five practical guidelines. chapter 17. 2014.

[68] Roderick J A Little and Donald B Rubin. Statistical Analysis with Missing Data. 1986.

[69] Ruslana Dalinina. Introduction to Correlation. *Oracle-Datascience Blog*, January 2017.

[70] Benesty J., Chen J., Huang Y., and Cohen I. Pearson Correlation Coefficient. In: Noise Reduction in Speech Processing. *Springer Topics in Signal Processing*, 2, 2009.

[71] Richard J. Light and Barry H Margolin. An Analysis of Variance for Categorical Data. *Journal of the American Statistical Association*, 66:534–544, 1971.

[72] Harald Cramer. The two-dimensional case. In *Mathematical Methods of Statistics*, chapter 21, page 282. Princeton: Princeton University Press, 1946.

[73] Chris Albon. Cross Validation With Parameter Tuning Using Grid Search. December 2017.

[74] Marcos Lopez de Prado. Hyper-Parameter Tuning with Cross-Validation. In *Advances in Financial Machine Learning*. John Wiley Sons, Inc., 2018.

[75] Alan Fontaine. Cross-validation and Parameter Tuning. In *Mastering Predictive Analytics with scikit-learn and TensorFlow*. Packt Publishing, 2018.

[76] Srinivasan. The Importance Of Cross Validation In Machine Learning, jul 2018.

[77] Gavin Hackeling. Training Decision Trees. In *Mastering Machine Learning with scikit-learn*, volume 2. Packt Publishing, 2017.

[78] Max Pumperla, Alex Tellez, and Michal Malohlava. Information gain. In *Mastering Machine Learning with Spark 2.x*. Packt Publishing, 2017.

[79] Leo Breiman. Random Forests. January 2001.

[80] Joseph Prusa, Taghi M. Khoshgoftaar, and Daivd J. Dittman. Using Ensemble Learners to Improve Classifier Performance on Tweet Sentiment Data. *2015 IEEE International Conference on Information Reuse and Integration*, August 2015.

[81] Ludmila I. Kuncheva. Ensemble methods. In *Combining Pattern Classifiers: Methods and Algorithms*, volume 2, chapter 6. John Wiley Sons, 2014.

[82] Luca Massaron and Alberto Boschetti. Bagging with Weak Classifiers. In *Python Data Science Essentials*, volume 3. Packt Publishing, 2018.

[83] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely Randomized Trees. November 2005.

[84] Toby Segaram. Making Recommendation. In Mary Treseler O'Brien and Sarah Schneider, editors, *Programming Collective Intelligence*, volume 1, chapter 2, pages 7–28. O'Reilly, August 2007.

[85] Mike Bernico. Count and TF-IDF vectorization. In *Deep Learning Quick Reference*. Packt Publishing, 2018.

[86] Sinan Ozdemir and Divya Susarla. The TF-IDF vectorizer. In *Feature Engineering Made Easy*. Packt Publishing, 2018.

[87] Florian Heimerl, Steffen Lohmann, Simon Lange, and Thomas Ertl. Word Cloud Explorer: Text Analytics Based on Word Clouds. *2014 47th Hawaii International Conference on System Sciences*, March 2014.

[88] M. Burch, S. Lohmann, D. Pompe, and D. Weiskopf. Prefix Tag Clouds. *17th Int. Conf. Information Visualisation*, pages 45–50, 2013.

[89] Jiawei Han, Micheline Kamber, and Jian Pei. Model Evaluation and Selection. In *Data Mining - Concepts and Technologies*, volume 3, chapter 8, pages 364 – 367. Morgan Kaufmann Publishers, 2012.

[90] Jesse Davis and Mark Goadrich. The Relationship Between Precision-Recall and ROC Curves. In *Proceedings of the 23rd International Conference on Machine Learning*. ACM, June 2006.