Building a Machine Learning Based Recommendation Engine for the Virtual Academic Advisor System

Rashi Goyal

A thesis

submitted in partial fulfillment of the

requirements of the degree of

Master of Science in Computer Science & Software Engineering

University of Washington

2018

Reading Committee:

Erika Parsons, Chair

Marc Dupuis

Wooyung Kim

Program authorized to offer degree:

Computer Science and Software Engineering

Abstract

Machine Learning is presently being used to tackle various problems from voice recognition to self-driving vehicles. However, there are many areas where modern software applications have not reached due to lack of interest or budget. The education sector [10] is one of these areas, and itself poses a set of interesting questions suitable for applied Machine Learning and modern Data Analysis approaches, which can greatly benefit the community. A relatable example is the problem of a student choosing a career path, the basis of which is an appropriate academic plan. In our state, community college students have difficulty choosing a career path as they do not have a well-defined academic path to transfer to a university and major of their choice. This is due to the fact that most of the advising is done with archaic tools (if any), and faculty also often pose as academic advisors when they are already overwhelmed by their daily responsibilities. Moreover, each student has specific preferences like the choice of school, major, budget, time preference, etc., making the task of generating the study plans burdensome. Study plan creation is a form of scheduling problem, and it is not trivial. There is little research on scheduling algorithms that address the problem of finding and recommending multiple paths going from multiple starting points to multiple goals (e.g., building prerequisite networks). The goal of this research is to help community college students and advisors by implementing a Machine Learning recommendation system that automates the selection of most suitable academic plans, specifically, to transfer to four-year institutions, based on personal preferences.

Contents

List of Tables

List of Figures

Acknowledgement

I would like to thank my chair Professor Erika Parsons for her supervision and support during this Capstone thesis. Ever since my very first grading job to taking a class on High-Performance Computing course, to this thesis work, Professor Parsons has always been a great teacher and 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 Parsons 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 loving and supporting husband, Sumit, without whom I would not have been able to achieve my goals and my parents, whose sacrifice and unconditional love made me the person I am today.

Chapter 1

Introduction

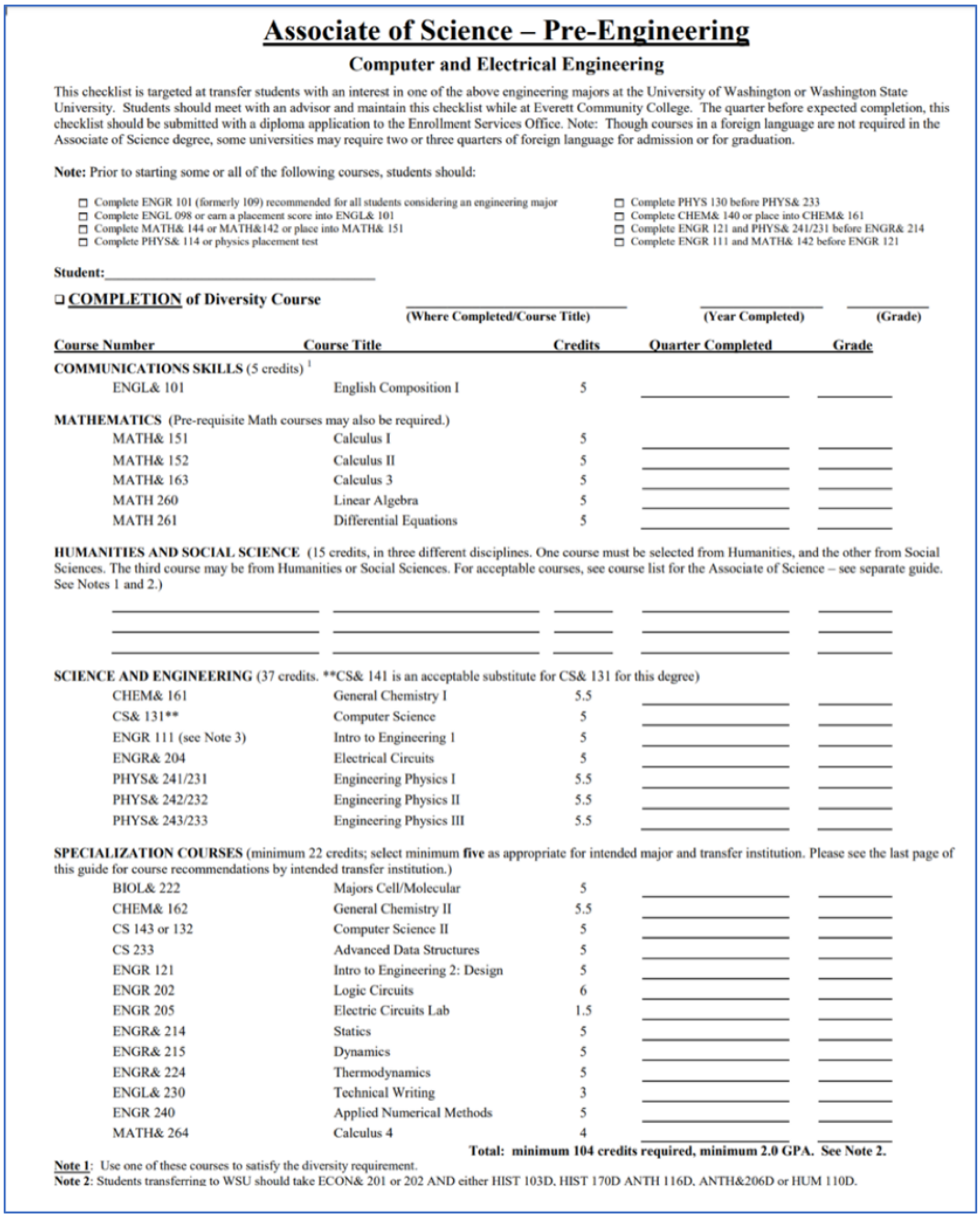
1.1 Background

Many students pursuing higher education often have a difficult time when choosing a career path. Making the basic choice of major and school is often not a simple choice. Students are often confused by the wide range of options, ambiguity and/or lack of information about different study programs offered in different schools. This is particularly true for many students in community colleges. The student population in these colleges is typically composed of first-generation students, returning students, veterans, single parents, working students, etc. In addition, community college students transfer into universities, a process which requires taking specific courses, number of credits, etc. The task of helping these students make a study plan usually falls on the faculty, who have to perform this on top of their regular responsibilities.While community colleges generally offer academic advising sessions, many times advisors (who are also faculty), find themselves overwhelmed with the number of students, or even lacking enough experience to efficiently walk students through the process. The process of choosing a career path and course selection becomes a complex problem

affected by various factors, both personal and logistic. For starters, each university has specific transfer requirements for each major, i.e., courses that need to be taken to transfer into a university academic program may widely differ between institutions. For instance, Figure 1.1 below shows a typical example of list of courses that are required to get admission in University of Washington or Washington State University to get into Computer and Electrical Engineering.

Also, many courses are prerequisites to other courses while some courses may not need prerequisites at all, which makes planning and creating an academic study plan (schedules) challenging unless there is experience and/or knowledge of this information. Figure 1.2 shows a representative example of a study plan for the first quarters of a Mechanical Engineering major student at a community college, wanting to transfer to WSU.

In addition, some courses, prerequisites or not, may not be offered every quarter or as desired by the student. For these and other reasons, students are often confused about the pathway they need to take to obtain a specific degree or to transfer to a university, for instance, what classes to take, the best time to enroll, number of credits required, financial issues, transfer requirements and other personal-life deciding factors. In this context, most community colleges do not have efficient means (e.g., technology and logistics) to provide proper guidance for students on their academic goals. During an academic "advising" session, most of the time is spent is making study plans from scratch using inadequate (or non-existent) software tools. Also, each student comes from a different background and with different life aspirations and restrictions as well as different academic expectations, so they require the study plan to be tailored according to their needs so that they can follow a clear path to success. Faculty, who also have to act as academic advisors, are overwhelmed by the process and the additional amount of work it entails.



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.

Figure 1.1: *Engineering Transfer guide from Everett Community College (EvCC). This paper is given to students to fill out and keep track of courses they need to take to transfer and when they take it*

A proposed solution to help address these issues is an automated recommendation system that creates and recommends study plans considering various factors, con-

Year 1

Fall	Credits	Winter	Credits	Spring	Credits
MATH 091	5	MATH 141	5	MATH 142	5
ENGR 101	2	CHEM 140	5	CHEM 161	5.5
ENGL 101	5	DIV	5	ENGR 111	5
HU/SS	5			PHYS 130	1

Year 2

Fall	Credits	Winter	Credits	Spring	Credits
MATH 151	5	MATH 152	5	MATH 163	5
CHEM 162	5.5	ENGR 121	5	ENGR 214	5
PHYS 114	5	PHYS 241	4	PHYS 242	4
		PHYS 231	1.5	PHYS 232	1.5

Year 3

Fall	Credits	Winter	Credits	Spring	Credits
MATH 264	5	MATH 260	5	MATH 261	5
ENGR 215	5	ENGR 225	5	ENGR 240	5
PHYS 243	4	ENGL 230	3	ECON 202	5
PHYS 233	1.5	ENGR 114	4		

Figure 1.2: *Example of a hand-crafted EvCC academic study plan for a Mechanical Engineering student planning on transferring to WSU.*

straints and preferences, for instance, desired major, transfer university, enrollment type, course placement, course availability, etc. With such an aid, Faculty/Advisory sessions can then be more focused on students' personal and career growth and selecting the best fitting study plans.

The Virtual Academic Advisor (VAA) software system is an initial approach towards a recommendation system that automates the process of making study plans (and hence improves the academic advising experience for community college students and faculty). This system was envisioned by Dr. Parsons as an interactive software application that could provide pre-university students coming from community colleges with academic plans tailored to their current background and interests, while minimizing the amount of unnecessary work that academic advisors must do to help students develop their study plans.

The scope of the work described in this document targets Everett Community College Engineering Department courses and transfer data to Washington State's uni-

versities.

1.1.1 Problem Statement

Thinking about an application that can address the problem of automated study plan recommendation, there are many aspects and entities involved. This makes the design problem a difficult one, partly due to the wide range of possibilities that need to be taken into account to automate decisions, for instance, from the different available majors at different universities and their corresponding enrollment requirements, to varied students' interests and backgrounds. In addition, it is necessary to build a logic, sequential workflow that conforms to the intended university's curriculum per quarter/semester, considers course prerequisites, adapts to unexpected changes in curriculum and suggests multiple alternatives and academic pathways. Automated recommendation is just part of a larger software system, so a solution needs to consider existing requirements, limitations and implementations. Based on the project requirements, the proposed work is an integration of a complex multi-module system that should be able to satisfy scalability, robustness, efficiency, and modularity standards.

1.1.2 Existing System and its Limitations

The current VAA implementation uses algorithms and data structures to deterministically create multiple study plans based on user's input, building known prerequisite networks for all the involved courses. Prerequisites and "post-requisites" are also an input to the system. The architecture of the VAA system has various interacting components. The proposed strategy aims to incorporate a ML-based module that with the use a feedback-loop approach will refine recommendations in the future. The original system's architecture is presented in Figure 1.3.

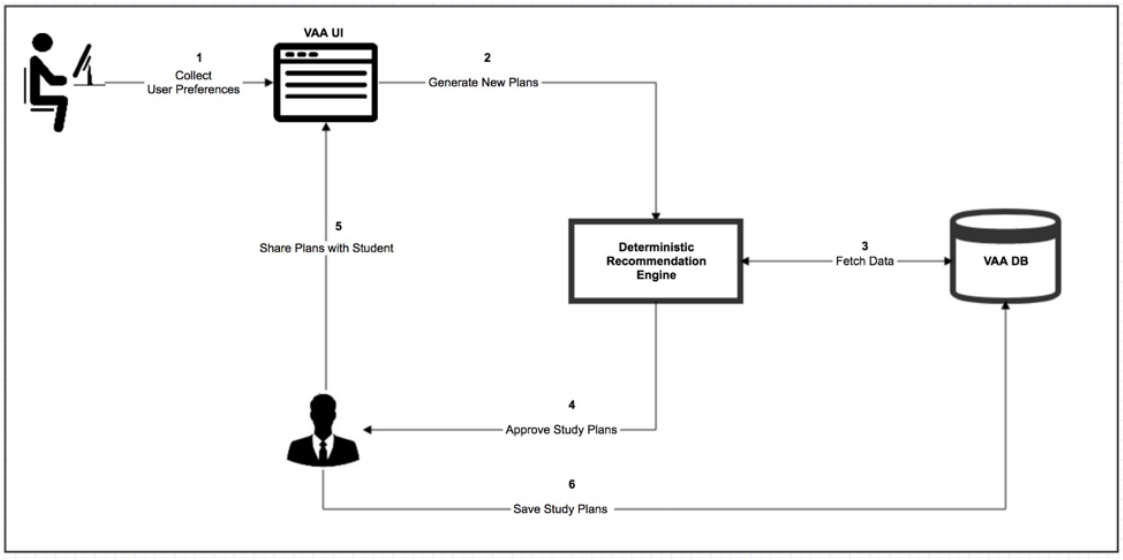


Figure 1.3: *Original System Architecture Overview: user preferences are collected through the UI and are used by a deterministic recommendation algorithm to generate study plans.*

The main goal of this thesis is to investigate machine learning (ML) strategies and technologies to implement a recommendation engine for VAA system. This recommendation system will select suitable academic study plans. The suitability of a study plan will be based on rankings first provided by professors, academic advisors, and the system itself. The current VAA algorithm is based on deterministic principles and exhaustive exploration of solution space search; it can generate as many as 50 or more study plans for a single combination of input parameters for a student.

The proposed improvements are to develop a module that uses the study plans generated by the Deterministic Recommendation Engine paired with rankings provided by faculty to build a knowledge base of high-ranked plans. In addition, we will use a classification model to identify "good" and "bad" study plans, also based on human-provided rankings.

In terms of limitations, the first issue is related to the deterministic nature of the current system. In this implementation, for a single combination of inputs (e.g.,

Major, University, class timings preferences, Enrollment type etc.), the algorithm can generate approximately 50 plans. From an advisor's perspective, looking through and filtering through all this plans generates additional work, defeating the purpose of an automated system. While faculty has advising knowledge and experience, they lack time and budget. Community colleges are already understaffed and advising is done in addition to their every-day duties. Student to teacher ratio is higher compared to universities. An automated recommendation system should aim to reduce their work rather than adding to it.

Other limitations are from a technical perspective: the existing algorithm lacks validation checks to verify the correctness of a plan, e.g., a study plan that looks reasonable to the inexperienced eye, may be missing a major course requirement. Furthermore, testing has shown that it is time consuming to run such an algorithm for every input combination because paired with all the possible combinations of feasible schedules, it is computationally expensive to generate all possible plans.

1.2 Proposed Solution

The proposed solution will generate an organized list of three or four study plans starting with the most adequate – based on user preferences – plan at the top. Figure 1.4 provides a full detailed data flow of the proposed VAA system. The recommendation model will take into consideration plausible student inputs such as target degree, intended transfer school (if applicable), student budget, time preference and graduation timeline. It will also consider the starting point of the student (e.g., placement), different course offerings in different quarters etc. For the recommendation system to be able to identify the best type of plans, a sub-module of classification will be implemented. Classification will also take into consideration features like target degree,

intended transfer school (if applicable), student budget, time preference and graduation timeline, etc. This, paired with the scores and reviews provided by faculty and advisors can be used to train the model. These reviews will be pre-set of statements given as an option with every study plan. As the VAA algorithm will generate n number of plans for each set of input provided by the students, classification model based on its previous training will divide these plans into six categories: Score 1 to 5 (5 being the best plans) and Incorrect plan category.

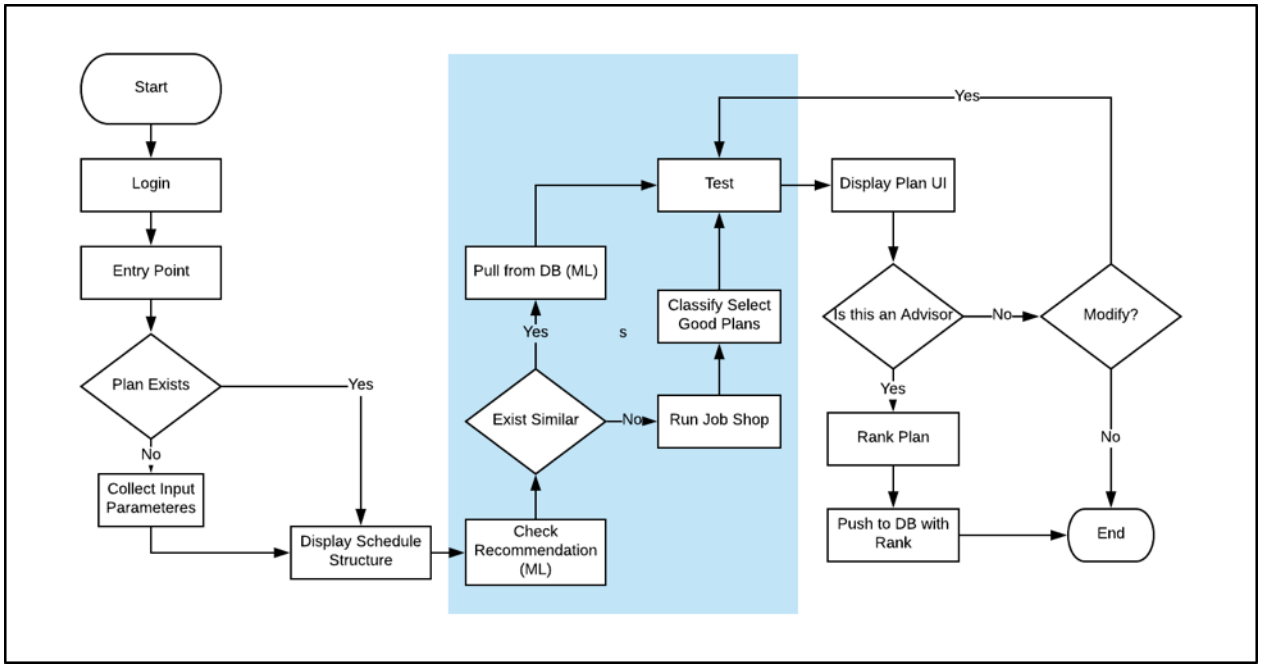


Figure 1.4: *Data Flow of the proposed solution*

Recommendation system can use knowledge collected from classification data to incrementally recommend better plans to the students. As soon as students enter their preferences in form of inputs, the recommendation system will run the algorithm and generate list of best possible plans for the student.

This system will have the effect of guiding the students in a personalized way to interesting and useful study plans in a large space of possible options. Recommended

plans can be divided into different classes by exploiting several representations of students' inputs and study plans to generate recommendations.

As an analogy, consider using Expedia© to buy a flight from departure point A to arrival point B. Given specific user provided parameters, such as air carrier, date, time, etc., Expedia© will provide a sorted list of options based on the search relevance.

In the case of the VAA, there are various departure points A_1, A_2, A_3, ... A_n (i.e., starting courses, placements), and many arrival goals B_1, B_2, B_3, ... B_n (i.e., target courses required for transferring to a specific university/major). Similarly, the system will take-in relevant parameters for making study plans, and correlate those input parameters with the recommendations to differentiate the best study plans from the not so useful study plans. However, as opposed to the flight example, study plan recommendation involves multiple starting points and multiple destinations.

1.2.1 Scope

Premise: *An automated Machine Learning Recommendation System can reduce the workload for advisors and help out community college students by selecting the most suitable academic plans to guide them for transferring to a four-year institution.*

The scope of this project, takes into consideration strict time constraints and can be outlined as follows:

- Use and perform feature engineering on the data provided by Everett Community College (EvCC).
- Investigate Machine Learning strategies suitable for building a Recommendation Module based on this type of data.
- Implement Classification and Recommendation modules using most suitable investigated ML strategies.

- Evaluate results from different ML approaches implemented.

1.2.2 Motivation

This project mainly targets community college faculty and students. Typically, community colleges are underfunded and few schools have resources to get and support advanced degree planning tools. Most community colleges handle academic degree planning by hand or with very primitive software tools, i.e., ML approaches are not even considered. Schools often have ad-hoc platforms, which include academic planning solutions that are hard-coded and do not provide dynamic recommendations. Community colleges have begun exploring, and in some cases, adopted available commercial solutions, which generally lack automation beyond hard-coded plans [10]. With the VAA system, students will be able to find the best-suited academic degree plan and thus increase their chance at completing their degree.

While the initial (experimental) version of the VAA system has been targeting Everett Community College, the project has been planned as a scalable solution, i.e., data structures, database, APIs, and other interfaces, have been designed to potentially support other colleges. Beyond creating study plans, the ideas researched in this project can be applied to similar scheduling problems with multiple starting points and multiple goals.

1.3 Related Work

Generally speaking, there is a lack of research and software development targeting the education sector, particularly in terms of identifying and creating academic degree plans. There is little knowledge or tools that can dynamically generate college-level student academic degree plans to help students plan their academic plan. This section

discusses external efforts that focus on identifying academic pathways; several commercial tools are used for creating academic plans; finally we discuss the internal (UWB) work that has been part of developing the VAA system.

Software Tools

From a technical standpoint, not much effort has been put into scheduling tools for education purposes. There are little or none proper software applications for use in education or academic settings. Many schools have internally developed ad-hoc tools that are too specific and cannot be extended to be used by other institutions or solve similar problems. These, and other commercial tools like Starfish from Hobsons [26] involve using hard coded information and decision paths, with simple, deterministic approaches at the best. Current tools, like the one used by EvCC [40] used for creating academic plans, are simple web forms with "notepad" functionality (see Figure 1.5) and scheduling is still done by hand.

Scheduling Approaches

Similar to software tools, there are few ML approaches for scheduling, let aside specifically targeting applications like academic planning. There are some non-ML approaches for scheduling, for instance, dynamic programming which is widely used in assembly-line scenarios, particularly, you can talk about the "Job Shop Scheduling" (JSS) [16]. The JSS approach is an optimization problem that, given a task T composed of n jobs with known duration, dependencies, and m machines to perform jobs, it can find feasible schedules to achieve T. This approach can yield up to $n!$ possible schedules, which is not very practical in many real life applications. This article [9] presents practical applications of the JSS such as: flexible manufacturing, multiprocessor task scheduling, robotic cell scheduling, railway scheduling and air traffic control

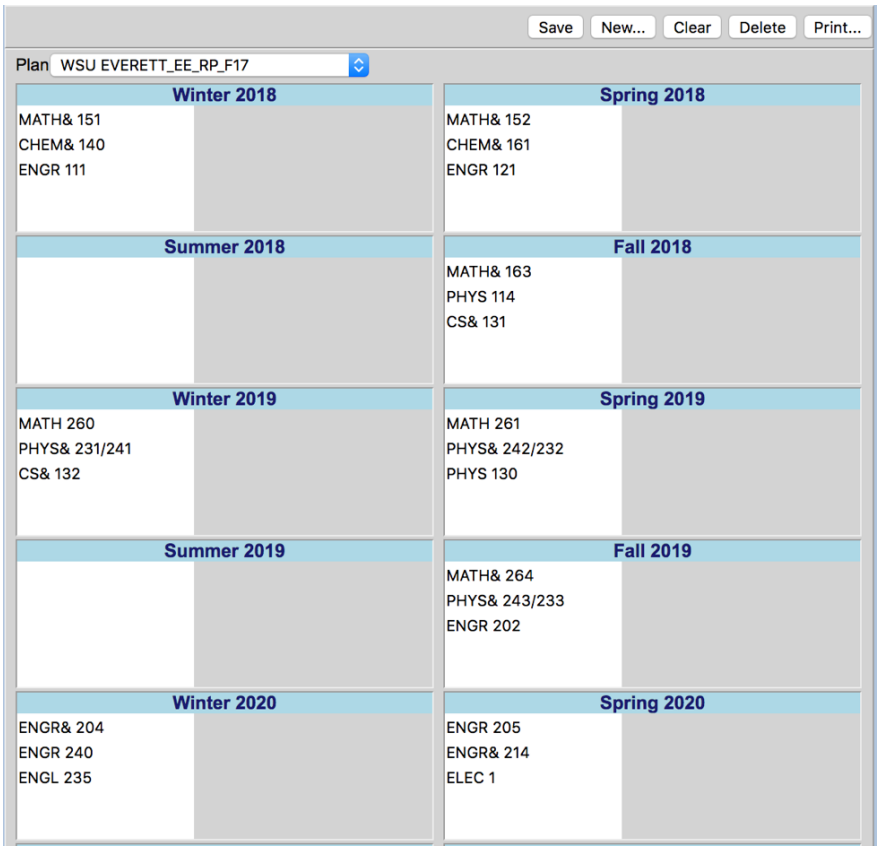


Figure 1.5: *Example of a Study Plan manually generated by faculty at EvCC for student who wants to transfer to WSU in Electrical Engineering program*

scheduling.

Internal Work

The VAA project has piloted a couple of basic prototypes for academic plan recommendation. The first approach used static data and plain hard-coded plans to create generic schedules without taking into consideration user preferences. A second approach was a deterministic (yet combinatorial) approaches that used very basic heuristics to rank and recommend top 20 plans out of hundreds. The former one had little flexibility, while the latter one was convoluted and computationally expensive.

Non-technical Efforts

The research from paper [10] confers the importance of advising in higher education and the challenges faced by this sector. According to their research, close to a billion dollars are spent on student success and retention services annually, but the student on-time graduation rates have generally not improved over the last decade. Studies show that less than 20% of total students finish the two-year degree in three years [10]. It also discusses amount of burden advisors carry as they may sometimes have to attend to thousands of students per year resulting in mistakes and inefficient advising. Figure 1.6 depicts an academic journey of a student and the obstacles they face along their path. A clear need for better and more precise academic planning could help the 43% of students who are unclear of their path to the workforce and 52% that struggle financially.

In 2017, Washington State began work to increase academic success in state community and technical colleges. Guided Pathways [21] addresses four pillars: clarifying the paths, helping students get on a path, helping students stay on the path, and ensure students are learning. Technology was identified as a major need for this effort.

1.4 Outline

The document is organized as follows: Chapter 2 presents a survey of some of the technologies we are using to solve our problem. Chapter 3 focuses on technologies used for the VAA recommendation system, as well as descriptions of implementation of methodologies investigated. The experimental setup and test results are discussed in Chapter 4. Finally, Chapter 5 presents research contributions and future work.

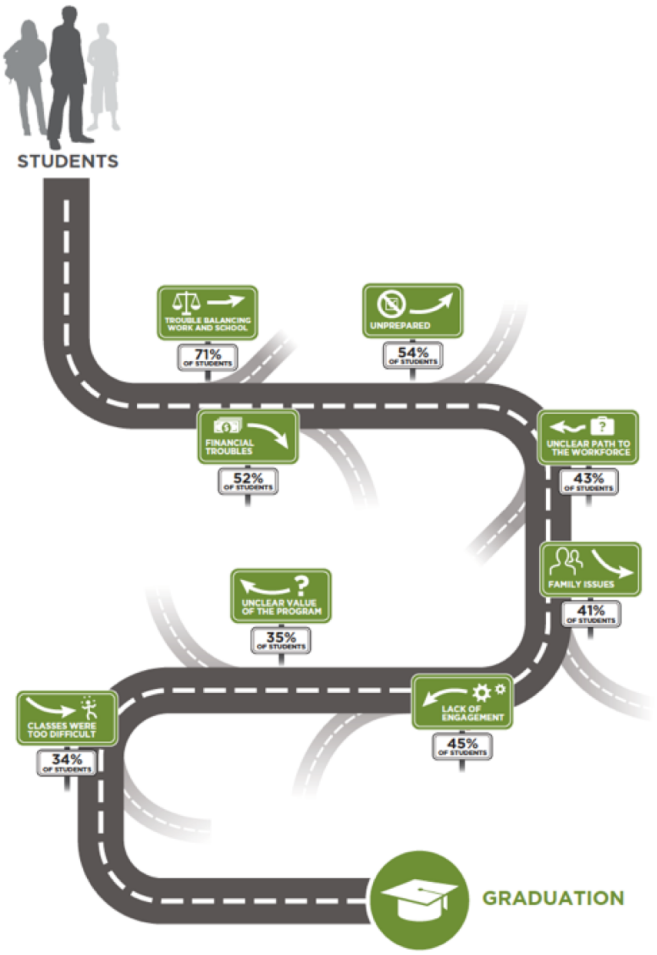


Figure 1.6: *Student's Academic Path and obstacles they face along the way. [10]*

Chapter 2

Background Concepts

This section provides information about the related works on different machine learning strategies and neural network models used in this thesis [12]. This section also discusses some major types of scheduling problems and the correlation between VAA and existing approaches.

2.1 Model Selection

ML algorithms can be categorized into Supervised and Unsupervised Learning. In this project, various models of both Supervised and Unsupervised Learning models are used to evaluate which of the algorithms are best-suited model for the VAA recommendation system.

Supervised and Unsupervised Learning

In supervised learning, the dataset is split into Training and Testing. The Training dataset contains the corresponding outcome variable Y for each observation X, given a space of input and output variables, e.g., $(X_1, Y_1), (X_2, Y_2), (X_3, Y_3)...(X_n, Y_n)$ [52].

Supervised learning models are trained to fit observations into a function $f(X)$ such that it can accurately predict the given outcome for the variable $Y : Y \rightarrow f(X)$. An example of supervised learning is Logistic Regression.

In unsupervised learning, the outcome variable is not available. We need to work with the observation (X) and identify relationships between each data point [28]. This information is used to split observed data points into distinct groups. This process is also known as clustering. An example of this strategy is kNN.

Artificial Neural Networks

This subsection contains an overview of foundations of Artificial Neural Networks and the definition of Deep Forward networks.

Artificial Neural Networks (ANN) have an extensive history. Researchers have been working on ANN for almost three decades. ANN can be described as a computational model that is inspired by the human brain [5] and are meant to mimic how neurons work. The human brain is responsible for consuming sensory inputs from adjacent setting to make complex everyday decisions, to be able to attain that information for some time. The human brain can be thought of as a complex computational system, consisting of approximately 100 billion biological neurons, connected by 10-100 trillion synapses [59].

ANN consist of interconnected processing elements called nodes or neurons that work together on some data to produce an output. These artificial neurons can be described as building blocks of ANN. The structure of a basic single input neuron can be described using Figure 2.1. "An input p with some weight w associated with it is passed to the Sigma function. The Sigma output n goes into a transfer function f also known as activation function, which produces the final output of a neuron. There is a bias b attached to another end of the Sigma function." [22]

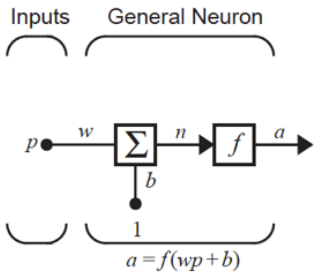


Figure 2.1: *Illustration of a single neuron [22]*

There are several types of activation functions that can be used to produce a final output of the neuron, e.g., Linear and Sigmoid. An activation function is required to set up like an upper limit or threshold for the input data in neuron which needs to be satisfied in order to create an output.

Multiple Layer Neuron: A layer of an ANN is many neurons functioning together in parallel on a set of inputs and weights associated. A multiple layer Neural Network(NN) can have many such layers depending on the problem. The first layer of such ANN is generally an input layer where the values of extracted features are fed into. The final layer yields the final output of the NN. Hence, it is called output layer. All the layers in between are called *hidden layers* because the functionality behind them cannot be defined.

An ANN can be of different types depending on the type of input data and solution model required to examine a particular type of output e.g., to differentiate different types of handwriting, the best type of ANN is Convolutional Neural Network. Below is a description of Feed Forward Neural Network which is relevant to this research.

Feed Forward: Data in Feed forward NN flows in one direction, i.e., forward direction from an input layer to next layer until it reaches the final layer and produces some output. Figure 2.2 depicts a very basic illustration of Feed Forward NN. Inputs are passed through neurons of input layer. There is no data flowing back from the

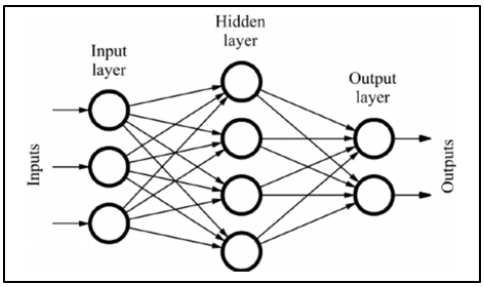


Figure 2.2: *Example of a simple Feed Forward NN [1]*

output layer to any of its preceding layer. Each layer represents a different function and the composition of all these layers together form a Feed Forward NN [20].

Backpropagation: This is a method of navigating the information from the output to each preceding layer in backwards ordering. Backpropagation provides feedback back to the neural network to improve its performance [56]. For example, in some cases of supervised learning where the target output is known, backpropagation is used to delegate the information about the difference in target output and observed output (i.e. output from Neural Network). This information helps NN adjust the weights and functions accordingly to produce output value closer to the target output value. This technique in combination with Feed Forward approach is used in our classification model.

2.2 Classification

Classification is the process of predicting a qualitative response for an observation [28]. In Supervised Learning, we want to predict the class of the test data after the classification model already has been trained with labeled data [33]. The data used for classification is called qualitative or categorical data [28]. Training Data is labeled with the known class of each observation. New data can be classified into different classes depending on the labels.

In the context of this work, we attempt classification using different ML approaches and then draw a conclusion based on the best performing algorithm. The goal of the VAA is to recommend feasible study plan (schedules) based on user parameters, exploring a full solution space of possible schedules using a deterministic approach, can yield $O(n!)$ number of solutions. We can use an ML-based classification approach to select the top N "best" study plans [35]. An overview of the approaches used is presented next.

Classification Using Neural Networks

In [57], [35], suitable NN models for multi-class classification are discussed, using k-class classification, which is a form of multi-class classification using different ANN models like Probabilistic Neural Network and Binary NN with additional decision module. Paper [25] compares classification using ANN over traditional ML approaches. When using a minimal training set, NN has been able to provide superior results over conventional ML approaches (detailed performance comparison of NN with back propagation is discussed in [25], [38], [8]).

Classification using Logistic Regression

Logistic Regression is used for two-class (binary) classification, it predicts the probability of an observation belonging to one of the two pre-defined classes or if an event will occur or not [60], [58]. This approach works with the assumption that the observed data belongs to one of the two categories of pre-defined classes [28]. Figure 2.3 below shows the plot of a logistic or Sigmoid function's values ranging between 0 to 1 – values close to 0 indicate that an event doesn't occur and values close to 1 indicate the opposite.

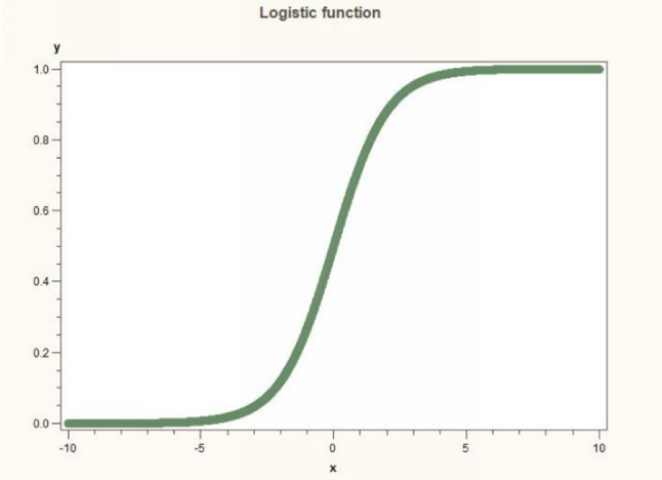


Figure 2.3: *Representation of Sigmoid function to predict probability[28]*

The equation for Sigmoid function is

$$Y = P(X) = \frac{e^{(b_0+b_1*X)}}{(1+e^{(b_0+b_1*X)}})$$

where

- Y is the outcome variable
- X is the observed variable
- b_0 is the bias or the intercept
- b_1 is the coefficient of the single input variable.
- $P(x)$ represents the log-odds of the outcome variable

Papers [60], [39] follow similar approaches to identify if the fault in the system is going to occur or not. The predictive performance of real-time environment such as VAA system developed using logistic regression needs to be evaluated in terms of two components: reliability or calibration (the agreement between predicted probabilities

of occurrence and observed proportions of study plans), and discrimination capacity (the ability of a model to correctly distinguish between "Good" and "Bad" study plans) [39].

Classification using k- Nearest Neighbor

k-Nearest Neighbor (kNN) is a simple supervised learning algorithm used for predicting the class for new data points based on the relationship known to nearby data points [54]. Figure 2.4 below illustrates an example.

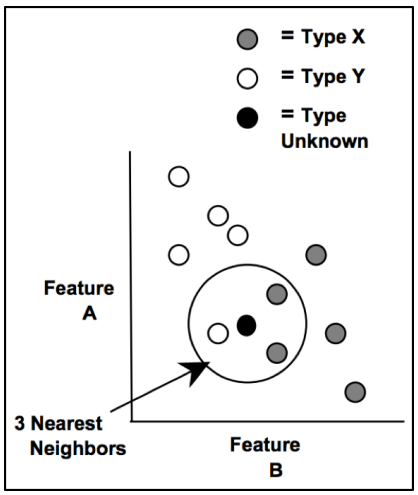


Figure 2.4: *A Simple kNN example [42]*

There are two known classes X and Y. Assuming the value of $k = 3$, a new unknown observation has three nearest neighbors, and since two out of three data points are of type X, this new observation will be classified in X [42]. kNN is not recommended for very large datasets since it is computationally expensive [51]. However, in the context of the VAA, we are dealing with a rather small dataset with a finite number of features so it is a reasonable approach for initial experimentation.

2.3 Recommendation System

A recommendation system is an ML strategy that takes into account users 'Rating' or 'Preferences' to predict the result and help to make future decisions [62], [14]. These systems track users' patterns observed in the past and then make recommendations based on their preferences. Recommendation systems are broadly classified into [13]:

Content Based systems

The history of a user is examined. For example, if a user likes to buy a certain brand, the system will recommend products of that brand.

Collaborative Filtering

A collaborative filtering algorithm [50] usually works by finding similarities between users [45], combining their preferences to create a ranked list of suggestions. This approach generally uses Euclidean distance score and Pearson correlation score to calculate the similarities between different users.

2.4 Job Shop Scheduling

There is little research/work on scheduling algorithms that address the problem of finding and recommending multiple paths going from multiple starting points to multiple goals (e.g., building prerequisite networks) [59]. There are some common strategies to find a single starting point to single goal paths based on prerequisites, for instance using graph algorithms (e.g., Global Positioning System). Even for simpler problems, the number of possible outcomes increases with the number of input parameters. Furthermore, the problem of finding paths from multiple starting points to multiple goals

grows exponentially based on the size of the input (hence a motivation for using ML).

Chapter 3

Methods

This chapter describes the architecture of the Virtual Academic Advisor (VAA), research methodologies, and explains how the proposed recommendation module solution adds to the VAA end-to-end system. We cover in detail the system's architecture and the various logical phases, and their corresponding implementation.

Environment Setup

Implementation environment is setup using Anaconda [43] software. This is an open source package software, popular for data analysis and scientific computing, suitable for our research. This environment comes with various of pre-installed packages and libraries like NumPy, SciPy, Matplotlib, Scikit-Learn, etc. Anaconda also comes with a variety of pre-installed software like "JUPYTER Notebook", "R Studio" and other virtualization tools. "JUPYTER Notebook" is a web-based python development environment where entire project development is accomplished [55]. Using this kind of software saves development time and environment setup.

Python

In recent years, Python has gained a lot of traction in ML space. So far, it has one of the biggest open source communities and has on of the most number of valuable ML libraries for developers [32]. These are the relevant/libraries used in this research:

- ***Data (Pandas)***: this is one of the most prevalent python package used for data analysis. A one-dimensional dataset can be stored as a series object and data frames are used for larger dimensional dataset[32]. This package offers easy-to-use features like aggregation, visualization through graphs, mathematical formulas that can be directly applied to a data frame.

- ***Visualization (Matplotlib)***: this package is one of the most popular and powerful visualization libraries [32]. Histograms, scatter plots, etc., can be easily generated.

- ***Seaborn***: this is a Python library based on Matplotlib [4]. It facilitates special functions for exploratory data analysis as well as dynamic graphing functions like heat maps, which are used to represent the correlation between every two variables.

Machine Learning Libraries

Scikit-Learn is one of the most popular and very well documented ML libraries. It supports all ML algorithms that are within the scope of the project. It also supports a lot of accessible packages for feature engineering. Also, it complements with Pandas library very well[41]. Some of the packages used are:

1. **Preprocessing**

   (a) *Feature Scaling*

(b) *Feature Engineering*

(c) *Feature Selection/Reduction packages*

(d) *Data Sampling*

(e) *Test Train Split*

2. **ML Algorithms**

(a) *Logistics Regression*

(b) *Artificial Neural Network (ANN)*

(c) *k-Nearest neighbor(kNN)*

(d) *k-means clustering*

3. **Evaluation Metrics**

(a) *Cross Validations*

(b) *Confusion Matrix*

3.1 Architecture

One of the goals of this research is to develop a deep architecture which accepts study plans generated by a Deterministic Recommendation Engine algorithm (Refer figure 3.2 Step 4). These study plans are ranked by a human on a scale of -1 or 1 to 5. The higher the score, better the recommended study plan is. As mentioned in the Introduction 1 chapter, ML recommendation systems require existing data-knowledge about the "Goodness" of a study plan obtained through a classification model. The main goal of the classification model is to rank these study plans based on the knowledge obtained from prior study plans.

3.1.1 High-Level Architecture

From a high-level standpoint, the software architecture is a multi-tier framework that includes a presentation layer, business logic layer and a data layer (Figure 3.1).

Presentation Layer

This is the front-end of the system, containing the graphic user interface (GUI) that provides a platform for users to interact with the VAA system. Inputs like University Name (students intend to transfer/enroll), preferred major, etc., are obtained from users. This information is passed to the Business Logic Layer to automatically generate study plans. In addition, it enables the display of multiple results based on user inputs/preferences(e.g., multiple recommended study plans, and rankings) .

Business Logic Layer

The Business Logic Layer (BLL) is the core of the back-end, it contains business rules and work-flow of the VAA system. The algorithm for each component/sub-component is embedded in this layer. It also includes the ML model, which defines the way in which a study plan is generated. It maintains to and fro communication with UI by sending and receiving data. The BLL also communicates with the data access layer. The two main components presented in this layer are:

Deterministic Algorithm Component: This component creates study plan a using job shop scheduling method [59] and graphs data structure algorithms. The student requirements provided through UI are used as input by Deterministic Recommendation Component to generate/schedule study plans accordingly.

Machine Learning Component: The purpose of this component is to reduce cost of generating plans, increase the efficiency of VAA system and provide useful

data to the user. This model recommends suitable study plans given user inputs (preferences and requirements). With the help of an ML component, the deterministic algorithm will be called only in case there are no preexisting study plans matching user inputs that can be recommended. This module also implements another ML-based sub-component used for classification. The Classification sub-component helps to identify the "Goodness" of the plan, a ranking strategy that doesn't involve human intervention.

Data Layer

This layer provides simplified access to user information and study plans stored in the database. The layer also enables storing user inputs and the data used by the system.

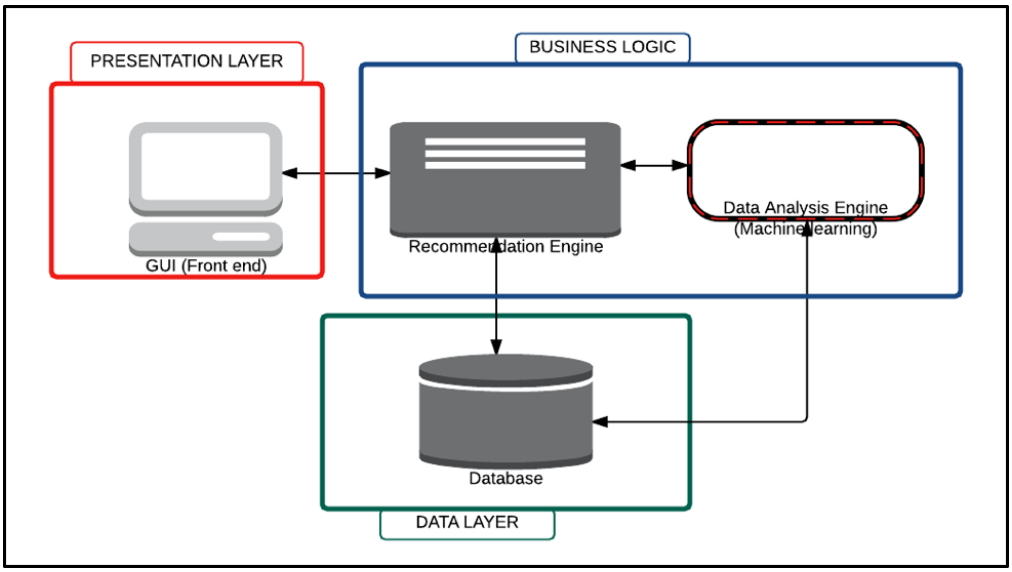


Figure 3.1: *High-level architecture*

3.1.2 Data Flow

Figure 3.2 below, shows the big-picture work flow of the Virtual Academic Advisor system. The high-level steps it follows are:

Step 1. Input parameters, and preferences, are entered by an advisor or a student through User Interface (UI).

Step 2. The UI feeds this information into the system's middleware and the ML Recommendation Engine, which then uses a database to checks if there exists a plan to recommend it to new students.

Step 3. The ML recommendation engine is able to find existing plans that are suitable based on the input parameter/preferences, it fetches those plans from the Database.

Step 4. If the ML Recommendation engine is not able to find any suitable plans, the input information is passed to the Deterministic Recommendation algorithm.

Step 5. The Deterministic Recommendation algorithm fetches data from database and creates study plans.

Step 6. These newly formed study plans are passed and evaluated through a classification model and categorized as "Good" or "Bad" plans.

Step 7. The outcome of this classification is new knowledge that is then saved in the database.

Step 8. These newly formed study plans are also ranked by the Advisor/Faculty to create more knowledge that can then be used by Machine Learning models.

Step 9. Advisor ranking is saved with each plan inside database.

Step 10. The approved final plans are shared with students.

The output of the deterministic algorithm is a list of study plans associated with the set of inputs entered through the UI. Each set of inputs is a combination of user preferences/requirements. This algorithm will generate at least 50 study plans. For each study plan, there will be an initial ranking score associated to it. At this point human intervention would be needed, i.e., the initial ranking needs to be provided by faculty/advisor.

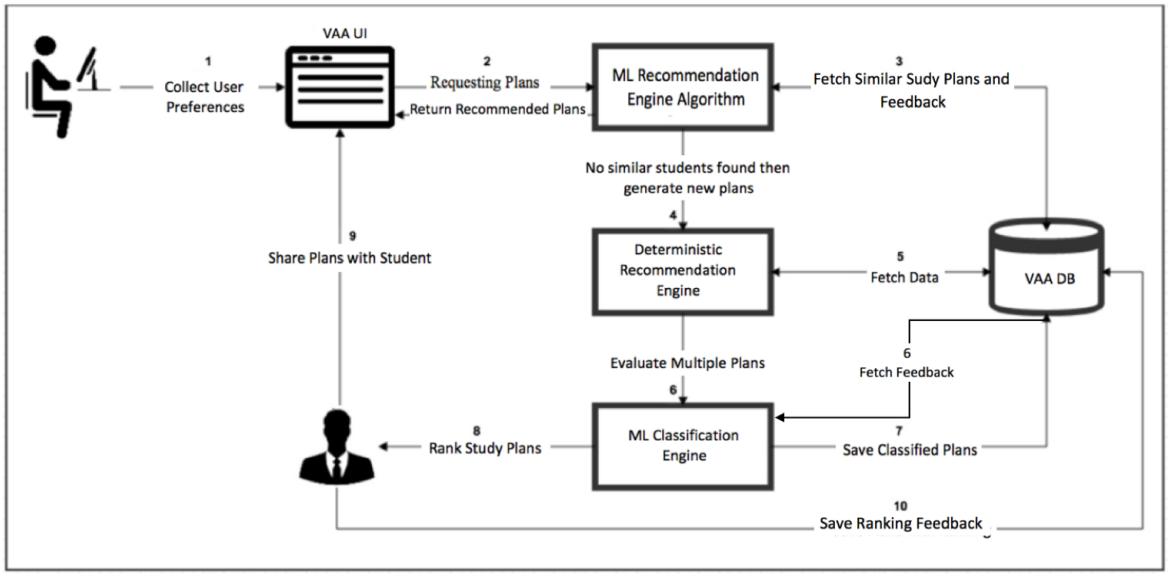


Figure 3.2: *Overview of VAA Workflow*

3.2 Exploratory Data Analysis

3.2.1 Data Source and Terminology

The data used in experiments has been provided by Everett Community College (EvCC) faculty/advisors. Some of this data consists of existing study plans manually created by faculty over a couple of years for their students. This data contains approximately 60 study plans. These plans exhibit diverse features since they were created by various faculty for diverse students with different requirements, backgrounds and goals. However, the size of this dataset is inadequate for the purpose of training ML algorithms but since the software tools used by EvCC are quite rudimentary, it is very difficult for them to collect or even keep track of consistent, reliable, suitable study plan data. For this reason, we need to carefully strategize how and what data we collect for testing a ML-based recommendation engine.

In addition to the sample plans provided by EvCC, we are also able to use the

deterministic algorithm to generate plans for testing. These plans are generated automatically, but again, they need to be ranked by a human (faculty/advisor). The purpose of this part of the process is that, with the help of human expertise, we can evaluate the validity and suitability of plans generated by the deterministic VAA algorithm. The feedback provided by humans will be used as key information for the implementation of supervised learning algorithms. Human intervention is critical: since there is no system like this in existence, there is no knowledge of how any algorithm performs when automatically generating study plans. This human feedback, along with the output from the deterministic algorithm, will be saved into the database and it can then be used in a classification model to separate "good" plans from the "bad" ones.

All the knowledge related to student preferences and recommended study plans is stored into the database and is used by the ML Recommendation Engine to provide suitable study plans based on requirements and/or preferences for each student.

Terminology

The following tables describe the terms we have defined to designate data features used in this work. These terms will be used through the rest of the document and are particularly relevant in feature engineering and classification. Table 3.1 contains terminology used to identify user preferences when creating a study plan. Table 3.2 contains plan ranks with associated descriptions determined through discussions with EvCC faculty/advisors. These users are familiar with reasons for plan suitability, common issues with plans, and they will be ranking plans as part of the system's feedback loop.

Table 3.1: User preferences terminology.

Term	**Description**
Core Courses	List of courses required by a target university to get admitted into a specific major. There is one list per university-major combination.
Prerequisite	Any course required to be completed before enrolling into a core course.
Ranking	Score between 1 and 5 indicating the "Goodness" of a study plan (similar to a 5-star rating system); in addition, a value of -1 indicates incorrect plans (i.e., with some logic error).
Reasons	A list of options indicating the reason for giving a score to a plan (e.g., why a plan is bad).
Preferences	(Student Preferences) Set of inputs entered through UI, containing preferences that affect plan configuration, e.g., desired university, major, enrollment type, employment status, etc.
Study Plan	The academic schedule (set of classes) that will allow a student to transfer to a major at a university of their choice. Courses are scheduled based on input preferences and constraints such as day/quarter offerings.

Table 3.2 contains plan ranks with associated descriptions. These were a result of discussions with EvCC faculty/advisors, who are the ones who ultimately will rank a plan, and are also familiar with the various reasons that can make a plan more suitable or not. The difference between each level of ranking.

Table 3.2: Plan ranking categories and criteria for assigning each ranking to a plan. These categories will be used for classification.

Rank	**Description**
5	Study plan is flawless. Courses are clustered together in a manner that level of difficulty in each quarter is consistent.
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.
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.
2	Study plan has at most 2 major issues. This type of plan needs major repair before recommending it to a student.
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.
-1	The study plan is incorrect. This is usually associated with critical mistake made by the deterministic algorithm (or human if created by hand).

3.2.2 Feature Engineering and Selection

Feature engineering is the process of enriching a dataset with meaningful features that complement ML models and yield better results [37]. For example, many ML algorithms cannot use categorical features values because they must rely on mathematical formulae. Such categorical features need to be engineered into new features using techniques like One Hot Encoding [11]. Feature extraction is very important for supervised

Table 3.3: Description of **Qualitative** features with corresponding values.

Feature Name	**Description**	**Values**
StrtQtr	Starting quarter of the existing study plan	Winter, Fall, Summer, Autumn
School	Target school of the existing plan	UW, UWB, WSU, WWU etc.
PrefDayNight	Student preferred schedule time for taking classes	Day or Night
StdntType	Student Enrollment Type	Full Time, Part Time
JobType	Student Employment Status	Full Time, Part Time or unemployed
PrefSchool	Students' preferred university for transferring	UW, UWB, WSU, WWU, etc.
PrefMajor	Students' preferred major	CSE, ME, EE, CS, etc.
PrefSummerQtr	Students' preference for taking Summer classes.	Yes, No

learning because observations characterized by features are labeled and then used for model training.

Features can be categorized into Categorical or Qualitative and Discrete or Quantitative [61]. The former is used for variables with values that can be positioned into specific categories, for instance, values described by words. The latter is used when a variable that can be described numerically, for instance, a continuous range.

Tables 3.3 and 3.4 contain the names of extracted features from the currently available VAA dataset with associated descriptions, type and approximate range of values.

Table 3.4: Description of **Quantitative** features with corresponding values.

Feature Name	**Description**	**Values**
PLANID	Unique identification number	Random positive integer
LengthInQtr	Length of the plan in terms of quarters	Integers like 8, 12, 15, etc.
QtrsWithCRSE	Number of quarters with courses	Integers like 2, 5, 6, etc., at most equal to the number of quarters
QtrsWithoutCRSE	Number of quarters without any course	Integers 1, 2, 4, etc., but should not exceed half of the total number of quarters
NmbrOfCRSE	Total Number of courses in a plan	Integer values, should at least have 3 courses and at most the maximum allowed per quarter times number of quarters
ReasonId	Predefined reasoning behind the different ratings	See Table 3.2
AvgNmbrOf CRSE	Average number of courses in a plan	Average calculated on number of courses per quarter, assuming that the maximum per quarter is 3 courses
LengthInYrs	Length of the plan in terms of years	Integers, could span between 1 and 5
YrsWthCRSE	Number of years with courses	Integer value, should be at least 1
AvgNmbrCRSE Day	Average number of courses scheduled in morning or afternoon	Taken with respect to maximum number of courses
AvgNmbrCRSE Night	Average number of courses scheduled in the evening	Taken with respect to maximum number of courses
Rating	Score assigned to a plan measuring correctness and "goodness"	Integers from 1 to 5 and -1
PrefCoreCRSE PerQtr	Preferred number of courses to take per quarter	Integer ranging between 1 and 7

Encoding Categorical Variables

ML algorithms like kNN or Logistic Regression are incompatible with categorical variables. Categorical features need to be converted into some form of numerical values.

In real-world data representation, categorical features may have more than two values e.g., a feature like "Preferred School" can have n number of values like University of Washington(UW), Washington State University(WSU), Seattle University(SU), etc.

In such cases, assigning arbitrary numerical values will likely negatively affect the performance of the classification algorithm because numbers would represent an ordinal value of significant order [61]. For instance, if we assign ordinal values to the "Preferred School" feature (e.g, see Table 3.2.2), the algorithm may interpret WSU to be the mean of UW and UWB.

To handle such features, encoding techniques like the One Hot Encoding [11] can be used. There are many other techniques that can be used for encoding categorical variables but due to the limited scope and requirement of this thesis, we focus on this only one technique.

Table 3.5: Example of Assignment of Ordinal values to Categorical Data.

Feature	**Ordinal Value**
UW	1
WSU	2
SU	3

One Hot Encoding

Each value in feature (as mentioned formerly) is represented as a distinct feature using this technique [11]. A binary value (0 or 1) is assigned to every value of that feature (table 3.2.2 presents an example of this method in the context of our problem). Similarly, other features are encoded into multiple values. An important point to be noted is that this technique may not be essentially applied to all features as some features can be easier to encode because they may already have Boolean values e.g., a feature

like “Summer Quarter Preference” has categorical values “Yes” or “No”. Hence, it can easily be represented as 0 or 1.

Table 3.6: Example of the application of One Hot Encoding technique on University categorical data.

Plan ID	**UW**	**WSU**	**SU**
1	0	1	0
2	1	0	0
3	0	0	1

Data Preprocessing

The VAA dataset, like many other real-life datasets, has a lot of noise. Detecting outliers is one of the most important aspects to address. Outliers can severely affect the performance of some ML models like kNN, which is very sensitive to outliers, causing issues with the statistical analysis[36] (for instance, consider a set of values 1,2,3,4,5, with the mean value of 3, but with an outlier of 100, the mean value would be a misleading 19.6).

Another common issue with datasets is missing values. The VAA dataset is manually generated dataset so it suffers from this issue [34], [31]. Missing values can significantly affect model conclusions. There are various types of missing values, but only one will be relevant in the context of our problem statement: Missing Completely at Random (MCAR).

MCAR means that values in the dataset are randomly missing, i.e., there is no relationship between the missing values of one feature with missing values from another feature. For instance, the feature “preferred number of core courses in a quarter” doesn’t have any values in the initial dataset because this is a preference that is in-

troduced by our software engineering approach. For such kind of scenarios, "Variable Deletion" technique is used.

Data Transformation

Many ML algorithms are affected due to the scale of input features. If a model is sensitive, Feature Scaling (also known as Feature Normalization) is used. Sensitive means that the accuracy of the model will be affected by feature scale. For the purpose of this project, we use the min-max scaling technique which transforms each feature by scaling it into a range between 0 and 1. The formula for scaling features is

$$\tilde{x} = \frac{x - min(x)}{max(x) - min(x)}$$

where $\tilde{x}$ represents the scaled value and x is the original value. Results of data

Univariate Analysis

This is the process of summarizing and exploring patterns in a single feature [49]. It is best attained with the use of histograms.

The **"Student Enrollment Type"** (StdntType) feature is useful to exemplify the difference between "Good" and "Bad" study plans, simply because an enrollment choice of "Part Time", would indicate that a suitable ("Good") plan should have no more than one or two courses per quarter. However using this feature alone for plan categorization could be misleading, for instance, it is possible that a random occurrence or an inadequate data set may mislead classification. As an example, an initial dataset was used to generate the histogram plots in Figure 3.3 (a) show that it is possible that study plans generated for "Full Time enrolled" students have higher chances of being categorized as "Good" regardless of the Enrollment Type choice.

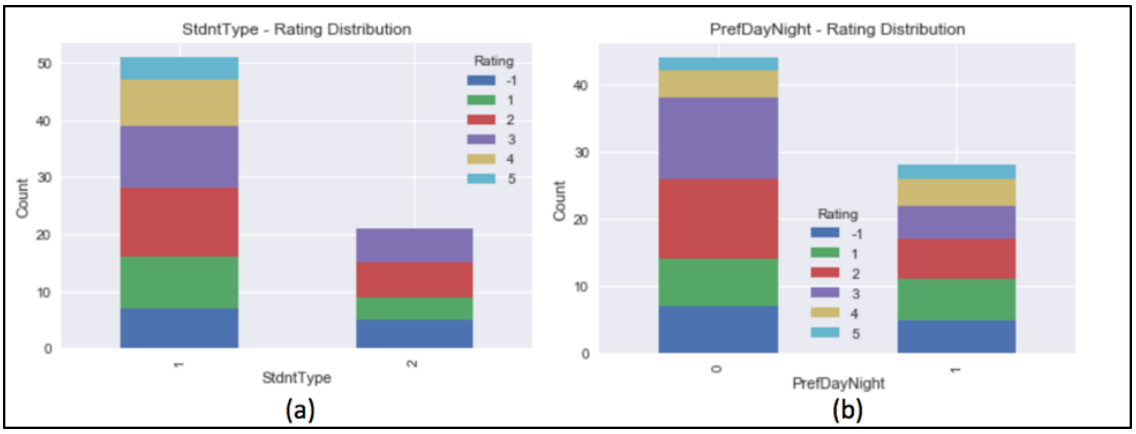


Figure 3.3: *Univariate Analysis*

Figure 3.3 represents a frequency distribution histogram for the Student Type feature. Each bar represents One Hot encoded values of this feature (Full Time as 1, Part Time as 2). The y-axis represents the count of occurrences in the population. From this histogram, it can be implied that a majority of students are interested in full-time course as opposed to part-time. Histogram bars are shaded based on the observations distribution across different ratings in the dataset. For example, the blue shaded portion of each bar represents study plans that were rated as -1.

On the other hand, the feature indicating a student's **"Day/Night preference"** (PrefDaynight) may not have any significant impact on the classification as it can be observed from Figure 3.3 (b). However, this feature is critical for ranking a plan, so it will be important to make sure that the dataset is capturing the desired behavior.

"Ranking" (Rating) is a dependent variable we use for multi-class classification. There are 6 different labels in which observations can be classified (see Section). Figure 3.4, shows that there are very few available observations for which a Ranking of 4 or 5 was given. This is an issue for any statistics-based methodology like ML. Hence, for the purpose of training the ML model, we have used Under-Sampling and Over Sampling techniques to rectify this problem. Under-Sampling is done by eliminating some of the

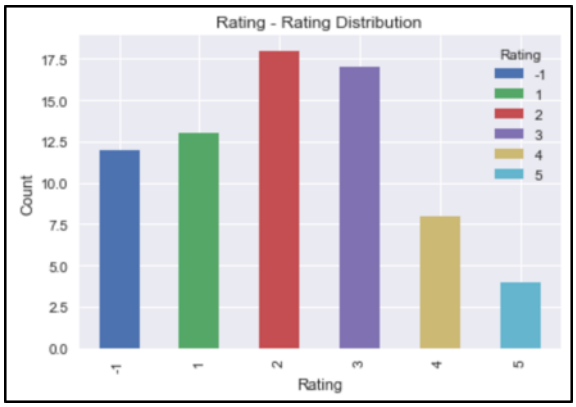


Figure 3.4: *Rating Distribution*

selected data (Study Plans in this case) from the classes that have the most data [15]. Over-Sampling is done by adding more data to the classes that have least data. For example, Adding more Study Plans with rating 4 and 5 as they have the least data (Figure 3.4) [15].

Bivariate Analysis

Bivariate analysis is the process of investigating an association between any two features. It is used to measure the strength of a relationship between two features. It is crucial to determine the correlation between features because some ML algorithms are based on the premise that the features are independent of each other (as opposed to correlated). Scatter plots are commonly used for an initial overview of relationships [30]. Three examples of scatter plots using our initial dataset are shown in Figures 3.5, 3.6 and 3.7.

The scatter plot in figure 3.5 represents two-dimensional space where the x-axis is "Length of a study plan in quarters" and y-axis is "Quarters with Core Courses". Each blue dot represents all data points in the dataset. The blue line represents a linear regression line that fits very well on this scatter plot. In Figure 3.5, we can see a straight line indicating that as the value of "Quarters with Core Courses" increases,

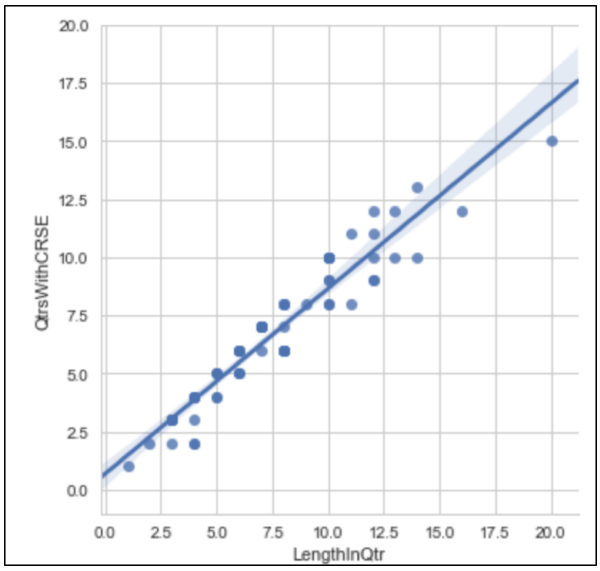


Figure 3.5: *Scatter Plot to explain correlation between LenghtInQtr and QtrWithCRSE*

so does the plan's "Length in Quarters". This shows a clear correlation between these features, which while intuitive, also indicates that there are not many samples with quarters that have no core courses.

On the other hand, in figure 3.6, the points are a bit more scattered, indicating higher variance, but still showing a positive relationship.

Another intuitive strong correlation is depicted in Figure 3.7, between features "Length in Quarters: LengthInQtr" and "Length in Years: LengthInYrs". It has already been established that LengthInYrs = (length in Qtr) /4 which represents a regression line with a slope of value .25. But also, this indicates that there are not many samples with "empty" quarters (no courses in them).

A *heat map* plot, like the one presented in Figure 3.8, is another tool used to demonstrate and summarize such relationships. Heat maps are based on *correlation coefficients*, which are used in statistics to measure the strength of a relationship is between two features. A correlation coefficient is a value between -1 and +1, where 0 means that there is no relationship between the features. While -1 indicates perfect

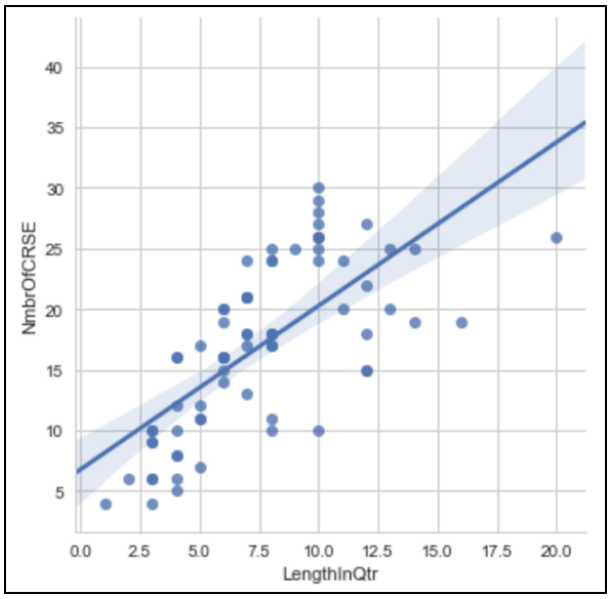


Figure 3.6: *Scatter Plot where x-axis represents "Length of a study plan in Quarters" and y-axis represents "Total number of core courses"*

inverse correlation, and +1 a perfect positive correlation. In a heat map like the one in Figure 3.8, bright yellow colored boxes represent highly correlated features, and the darker the color, the lesser the correlation. For example, feature "Length of a Plan in Quarters" (LengthInQtr) is highly correlated with the feature "Number of Quarters with Courses" (QtrsWithCrse) and slightly less correlated with the feature "Total Number of Courses" (NumberOfCrse).

Handling Class Imbalance in a Dataset

The VAA is a new system under current development. For this reason, and because EvCC advising software has extremely limited functionality to collect existing study plans, our initial dataset is small. Related to this issue, two out of six classes are under-represented, creating a class imbalance. This imbalance becomes even more problematic when the overall data is split into 80% for the training set and the remaining 20% for testing. A reasonable approach to deal with this problem is to generate

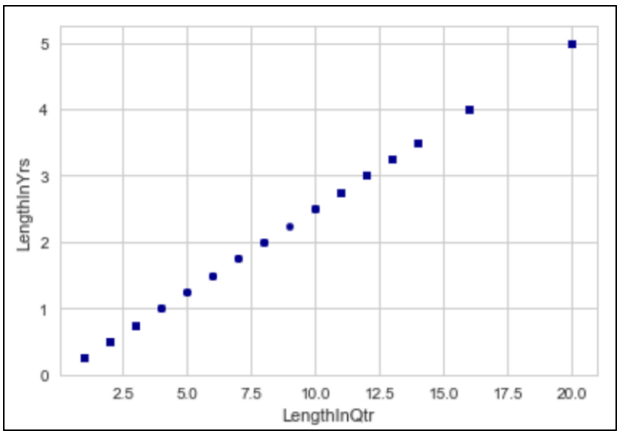


Figure 3.7: *Scatter Plots to explain correlation between "Length in Quarters" on y-axis and "Length in Years" on x-axis*

new samples. Imbalanced-Learn is one of the contributions from the **Scikit-Learn community** [19] . This library provides provides a RandomSampler function that can be used to balance the underrepresented classes [27].

3.2.3 Feature Selection

It is important to provide relevant features to ML algorithm as irrelevant features increase the complexity of the model, ML Algorithm may take longer to train or the accuracy of the model may decrease.

The features are selected using Recursive Feature Elimination method. This algorithm helps identify best features of the subset. The method recursively creates ML models by removing a feature and ranks each model based on its accuracy. This process is implemented continuously till all features in the dataset are exhausted [46]. In the end, the features are ranked based on model performance. Features with bad rankings are removed manually from the dataset. For the purpose of this thesis, RFE with Cross-Validation is used.

In supervised learning, it is necessary to have a test and a training set. To implement

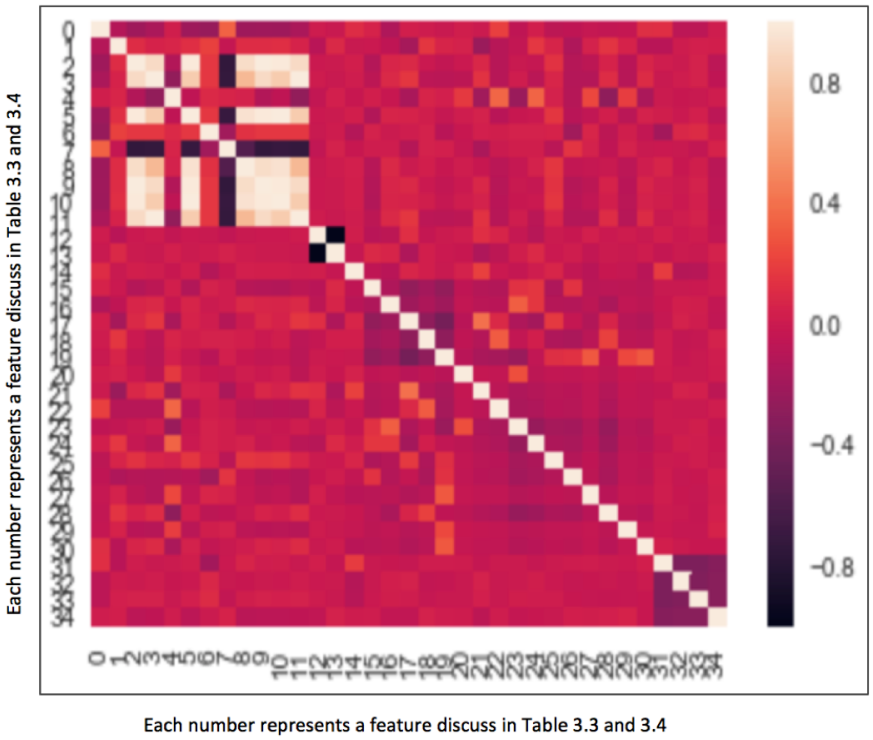


Figure 3.8: *Bivariate Analysis using a Heat Map: feature names are omitted due to space constraints, instead numbers are used to enumerate them. Highest correlations correspond to 1 and -1. Correlation in features 2 through 9 (associated with preferences) and are due to artificial data issues.*

this, we have used the *K-Fold Cross Validation* technique. In this approach, the dataset is split into K subsets, where K - 1 subsets are used for training and a kth subset is used for testing. This process is repeated K times where every Kth subset is different. With this approach, all data points participate in training and testing. The average of error in K iterations is used to compute the model performance. In stratified K-Fold Cross Validation method, each fold will have data for all classes and in equal proportion between the folds [2].

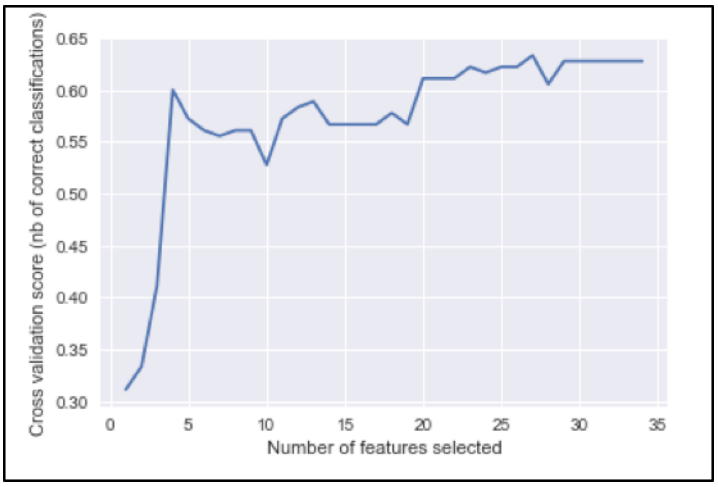


Figure 3.9: *Scatter Plots to explain correlation between LenghtInQtr and LengthInYrs*

3.3 Classification Implementation

In this section we discuss the strategies used for implementing ML-based classification of study plans based on ranking.

Training and Testing Dataset Split

The data was split into 80%:20% training:testing dataset using Quota-based Random sampling [44]. This technique causes the dataset to split into n sub-group based on quotas (filter Conditions). Once the dataset is split, the samples are taken from each sub-group randomly. We have used this technique to split train and test dataset equally for each class.

3.3.1 ANN

This system is based on a Feed Forward ANN using back propagation method. Back-propagation is commonly used with gradient descent optimization to adjust the weight of nodes by calculating the gradient of the error function.

An ANN optimization is a two-phase cycle, feed forward and weight update. When an input feature vector is presented to the network, it is propagated forward through the network, layer by layer, until it reaches the output layer. The output of the network is then compared to the desired output using an error function. The error values are then propagated from the output back through the network, until each node has an associated error values that contributes to the original output. This model consists of three layers: Input, Hidden, and Output layer. It can have more than one hidden layers. The number of hidden layers is based on the complexity of the problem or the type of ANN.

Input Layer

The Input layer represents the input vector $I(x)$ from the dataset. The number of neurons is equal to the number of features to be passed as input to the model. In this model (Figure 3.10), we have 21 neurons corresponding to the 21 number of features used for training the model.

Hidden Layers

The number of hidden layers (i) can be based on the complexity of the problem or the type of ANN. It represents the dot product of the input vector and weight ($\mathrm{W_{i,j}}$) .

The size of the matrix is based on the number of neurons in the hidden layer All the neurons in the hidden (j) and output layer use an activation function. The output of the activation function is transmitted to the output layer through weighted links.

Output Layer

This layer has 6 neurons for classifying the plans. As shown in figure 3.10, a score will be calculated for each neuron. Whichever neuron has the highest score will be the class

or label for that study plan.

Training and Testing

The artificial neural network is trained using backpropagation technique (discussed in detail in 2.1). Preprocessed features are passed to the input layer of the network. A feature matrix $I(x)$ is generated with randomly selected weights. Feature inputs are processed through multiple hidden layers. The inputs then are multiplied by the assigned random weight matrix ($W_{i,j}$). After the multiplication is done, an activation function is applied to the product. This process is then repeated between multiple hidden layers before it finally produces an output through output layer. In the end, the error between desired and obtained output is computed. With the backpropagation technique, the difference between existing weight and desired weights is computed and weights are updated accordingly. Once the weights are updated for all the layers, the same process is repeated for a new set of sample records in the training dataset. With enough training samples, it is expected that the neural network weights will converge to an optimum value that can be used for prediction on test data set. The inputs for the algorithm are as follows:

- Input vector $I(x)$
- Weight Matrix $W_{i,j}$
- Output Vector
- Number of Hidden layers i
- Number of Neurons in each hidden layer j

Test data is evaluated using the trained model. The outcome can be evaluated using a confusion matrix (discussed in Chapter 4).

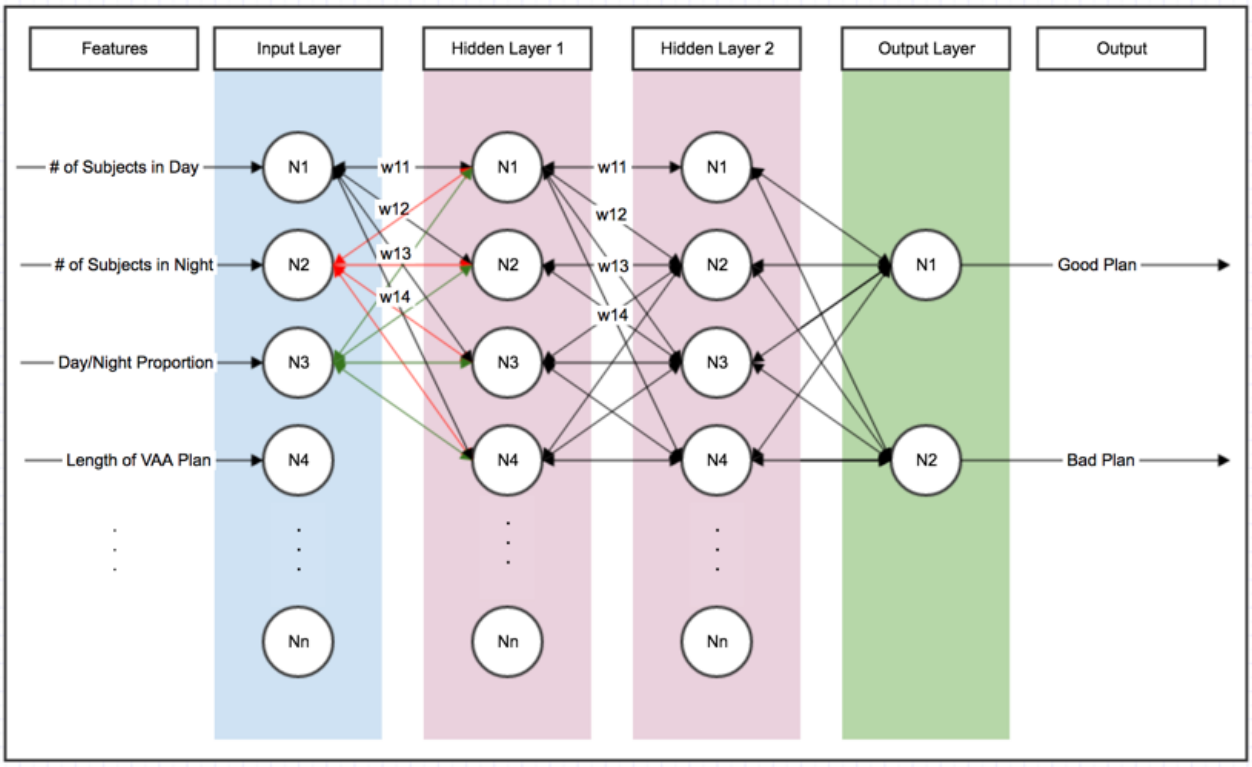


Figure 3.10: *Neural Network implementation overview*

3.3.2 Logistic Regression

Training

For a multi-class classification problem, a **One vs. All** classifier is used. The final ML model consists of multiple classifiers that generates probabilities for each class. Each classifier is trained to classify a specific class in a binary classification setting.

Testing

Test data is used to predict the classes and evaluate the performance of the classifier. The **One vs. All Model** evaluates test data with all the classifiers and evaluates probabilities obtained for each class $P_1(X)$, $P_2(X)$..P_i (X). **The class with maximum**

probability is used for classification.

3.3.3 k Nearest Neighbor (kNN)

Training

The k-Nearest Neighbors classifier estimates conditional probability of a data point from its neighbors. For each observation X_0, classifier finds k nearest neighbors one using Euclidean distance (d).

$$d(x, x') = \sqrt{(x_1 - x_1{}')^2 + (x_2 - x_2{}')^2 + ... + (x_n - x_n{}')^2}$$

The k nearest neighbors are represented as N_0. The conditional probability of a data point X_0 in test dataset that it belongs to class j is evaluated using the average of a count of k nearest neighbor's in N_0 neighborhood.

Testing

If $K = 6$ and K_1 , K_2, K_3, K_6, belongs to class j and K_4 , K_5 belongs to other class then probability would be

$$Pr(Y = j|X = X_0) = 4/6$$

3.4 Recommendation System Implementation

The proposed implementation for the VAA Recommendation System is to use Collaborative filtering method [45] and K-Means clustering [29].

From a high-level, the recommendation architecture is depicted in Figure 3.11.

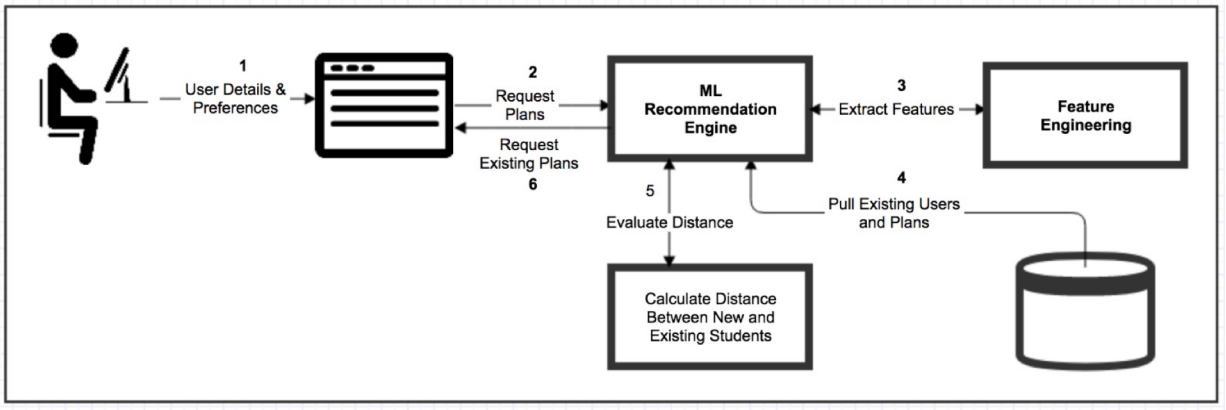


Figure 3.11: *Data flow description of Recommendation System*

The collaborative filtering algorithm works to identify a group of data points that are similar to a given instance in the dataset. For example, during the creation of a new plan for a student, a set of associated inputs is provided, associated to preferences, $S_1 = (x_{1,1}, x_{1,2}, x_{1,3},...,x_{1,n})$, and possibly, there is another set of inputs (for a different student) that has a similar set of features $S_2 = (x_{2,1}, x_{2,2}, x_{2,3},...,x_{2,n})$ ... $S_m = (x_{m,1}, x_{m,2}, x_{m,3},...,x_{m,n})$. The Collaborative Filtering algorithm will compute their similarity in terms of distance.

Algorithm:

1. The assumption is that there is N number of students that with associated preferences in the dataset.

2. Take an instance of a student's preferences and compute its Euclidean distance from all other students' preferences.

3. Sort these data points by computed distance, ordering from lowest to highest.

4. All students' preferences that are at a distance below the threshold T are selected, where T represents the maximum distance allowed to consider an existing set of

preferences for recommendation.

5. Existing study plans from selected preferences are displayed in the order of distance.

The distance (d) between students' preferences can be calculated using Euclidean distance approach

$$d(S_1, S_2) = \sqrt{(x_1 - x_1{}')^2 + (x_2 - x_2{}')^2 + ... + (x_n - x_n{}')^2}$$

Example:

1. Given the preferences for a student's study plan S_1, with feature tuple ¡Major, School, JobType, PrefDayNight, StudentType, PrefCoreCRSEPerQtr, PrefSummerQtr¿, and possible values discussed in tables 3.4 and 3.3, assume a study plan with the following preferences:

$$x = (' ME ',' WSU ',' FullTime ',' Preferredtimeday ',' FullTime ', 3,' Yes ')$$

or

$$x = (`ME', `WSU', 1, 0, 1, 3, 1)$$

2. Features "Major" and "School" are categorical variables, so they are pre-processed and converted into numerical variables using One Hot encoding method: Pre-processed

$$S_1 = (3, 3, 1, 0, 1, 3, 1)$$

3. Calculate Euclidean distance between a given student's plan with another student's plan, based on their preferences, S_2 = (1, 3, 1, 0, 1, 3, 0)

$$d(S_1, S_2) =$$

$$\sqrt{((3-1)^2 + (3-3)^2 + (1-1)^2 + (0-0)^2 + (1-1)^2 + (3-3)^2 + (1-0)^2)}$$

$$d(S_1, S_2) = \sqrt{5}$$

$$d(S_1, S_2) = 2.23$$

4. Similarly, we can calculate the distance between a given instance of a student's preferences and all other known preferences associated to existing plans.

5. We then sort similar study plans using the distance (short to long) from the new study plan:

$$(S_{\mathrm{n}}, d) = ((S_2, 2.23), (S_3, 2.28), ..., (S_n, 8.23))$$

6. Using the distances between two plans' preferences, create a list of "closest" plans, e.g., those within a distance d smaller than a threshold $T = 3$ ($d \leq T$).

7. Most similar student preferences in the list above are S_2 and S_3. The plan associated with these preferences can be recommended.

Based on the example above, note that a recommended study plan with $d = 2.23$, would still not be appropriate for a student S_2, since the associated preferences show that the selected Major is different, which is a critical error that would result in a plan ranking of -1 (unusable).

To overcome such problems, we have implemented a weighted feature distance calculation approach where each feature has different weight according to its importance. This approach is similar to weighted KNN approach (Refer 3.12).

The weights are allocated according to importance of each feature when calculating distance. Hence, using figure 3.12, we can state S_1 as

$$S_1 = (`ME', `WSU', 1, 0, 1, 3, 1) = (3000, 3000, 10, 0, 500, 300, 500)$$

$$S_2 = (`CSE', ' WSU' 3, 1, 0, 1, 3, 1) = (2000, 3000, 10, 0, 500, 300, 500)$$

$$x_3 = (`ME', `WSU', 1, 0, 1, 3, 0) = (3000, 3000, 10, 0, 500, 300, 100)$$

School		Major		Job Type		CRSE / Qtr.		Day / Night		Summer Qtr.		Student Type	
UW	1000	EE	1000	1	10	1	100	0	100	0	100	0	100
UWB	2000	CSE	2000	2	20	2	200	1	1000	1	500	1	500
WSU	3000	ME	3000	3	30	3	300						

Figure 3.12: *Weight Matrix for feature distance calculation*

Re-calculating Euclidean distance(d) of student S_1 with another student S_2 that has same school preference but different Major S_2 = (1, 3, 1, 0, 1, 3, 0)

$$d(S_1, S_2) = 100$$

Re-calculating Euclidean distance(d) of student S_1 with another student S_3 that has same school and major preference but different preference for classes in summer quarter

$$d(x, x_2') = 20$$

Thus, after sorting distance from low to high,the most similar study plan for S_1 should be recommended accordingly study plan of student S_3.

Disadvantages

This algorithm is calculating distance of one student with each and every student available. Consequently, time complexity of the logic is high because with increase in number of students, time required to make recommendations will increase as well. Hence, we replicated the same solution using K-Means clustering machine learning algorithm [29] where students can be grouped into clusters and do not require such calculation for every student.

3.4.1 K-Means Clustering

K-means clustering is an unsupervised clustering algorithm. It groups dataset into user specified number of clusters. K means clustering algorithms takes a set of inputs $x = (x_1, x_2, x_3, ..., x_n)$ where each x_i (for $i = 1$ to n) represents a data point in VAA dataset. The K-Means clustering algorithm partitions data into K subsets called clusters $C = (c_1, c_2, c_3, ..., c_k)$ where $k \leq n$ and $k \geq 0$.

This algorithm is an optimization solution that aims on minimizing the sum of squares of distance within-cluster [29]

$$\sum_{i=0}^{n} min_{\mu_j \in C}(||x_j - \mu_i||^2)$$

where μ_j =centroid of the data points

Algorithm

1. For a provided K value (where K is the number of clusters), randomly choose K centroids in the feature space that are farthest from each other. They would represent initial centroids location for clusters.

2. Measure distance between each data point and centroid locations.

3. Allocate each data point to its closest centroid.

4. Re-calculate centroid value of a cluster by calculating the mean of all data points within each cluster.

5. Repeat Step 1 to 4 until...

   (a) Data points stop moving from one cluster to another

   (b) Centroid value does not change

For the purpose of this research, K means clustering algorithm is being performed to identify student that are similar to each other.

Example

1. The assumption is that we have data for 8 students, and they need to be grouped into 2 clusters. Refer figure 3.13

#	Major		School		Job Type	CRSE / Qtr.	Day / Night	Summer Qtr.	Student Type
1	ME	**3**	WSU	3	1	2	0	0	1
2	ME	**3**	WSU	3	2	3	1	0	0
3	ME	**3**	WSU	3	3	3	0	1	1
4	ME	**3**	WSU	3	1	4	1	1	0
5	CSE	**2**	WSU	3	2	2	0	0	0
6	CSE	**2**	WSU	3	3	3	1	0	1
7	CSE	**2**	WSU	3	1	3	0	1	0
8	CSE	**2**	WSU	3	2	4	1	1	1

Figure 3.13: *Dataset containing feature values for 8 study plans*

2. The feature space is of 7 dimensions. Due to the curse of dimensionality, it is not possible to visualize these many dimensions on a single plot, but this computation can be explained on a matrix.

3. Let's assume that initial centroid value for two cluster ($K = 2$) are as follows:

   $K1 = (1, 1, 1, 1, 0, 0, 0)$

   $K2 = (4, 5, 4, 5, 1, 1, 1)$

#	Major	School	Job Type	CRSE / Qtr.	Day / Night	Summer Qtr.	Student Type	Distance From K1	Distance From K2	CLUSTER
1	3	3	1	2	0	0	1	3.16	5.00	K1
2	3	3	2	3	1	0	0	3.74	3.87	K1
3	3	3	3	3	0	1	1	4.24	3.32	K2
4	3	3	1	4	1	1	0	4.36	4.00	K2
5	2	3	2	2	0	0	0	2.65	4.90	K1
6	2	3	3	3	1	0	1	3.87	3.74	K2
7	2	3	1	3	0	1	0	3.16	4.80	K1
8	2	3	2	4	1	1	1	4.24	3.61	K2

Figure 3.14: *Data points with corresponding Euclidean distance from cluster $K1$ and $K2$*

4. After the first iteration of clustering, it is found that 8 data points are split into two distinct clusters.

5. The mean value of each cluster is recomputed using values of the data points in each cluster

   For cluster $K1$, refer to Figure 3.15

Major	School	Job Type	CRSE / Qtr.	Day / Night	Summer Qtr.	Student Type
2.5	3	1.5	2.5	0.25	0.25	0.25

Figure 3.15: *Mean values of data points in K1*

For cluster K2, refer figure 3.16

Major	School	Job Type	CRSE / Qtr.	Day / Night	Summer Qtr.	Student Type
2.5	3	2.25	3.5	0.75	0.75	0.75

Figure 3.16: *Mean values of data points in K2*

6. Recompute distance of each data point between two means and rearrange the clusters.

#	Major	School	Job Type	CRSE / Qtr.	Day / Night	Summer Qtr.	Student Type	Distance From K1	Distance From K2	CLUSTER
1	3	3	1	2	0	0	1	1.2	2.29	K1
2	3	3	2	3	1	0	0	1.2	1.32	K1
3	3	3	3	3	0	1	1	1.98	1.32	K2
4	3	3	1	4	1	1	0	1.98	1.66	K2
5	2	3	2	2	0	0	0	0.97	2.06	K1
6	2	3	3	3	1	0	1	1.98	1.32	K2
7	2	3	1	3	0	1	0	1.2	1.8	K1
8	2	3	2	4	1	1	1	2.11	0.87	K2

Figure 3.17: *Data points with re-computed Euclidean distance from cluster K1 and K2*

This is a simplified example that demonstrates K-Means clustering algorithm. Therefore, these 8 data points are grouped into two clusters in one iteration only. However, in real life examples, finding clusters may not be that easy. The grouping of data points into clusters varies with different starting centroid values or K values. Sometimes, it may require multiple iterations to come to a conclusion about correct formation of clusters.

3.4.2 Recommendation Approach

K-Means Clustering is an unsupervised machine learning algorithm that groups dataset into given number of K clusters. This algorithm is dependent on K value provided by a user. This algorithm requires some methods to identify the right number of K clusters in the dataset. In the scope of this research, we focus on the use of the Elbow Method and Silhouette Analysis to find appropriate values of K in K-means clustering. Both of these methods provide good visual representations for analysis, making it easier to interpret results within a fairly unknown problem like ours.

Elbow Method

This method represents the variance as the sum of squared distance for changing values of K. Variance is explained as a function of the number of clusters that should be chosen in such a way, that adding another cluster should not give much better modeling of the data [18]. The assumption is that as the number of K cluster increases, the sum of squared distance should decrease [7]. For an appropriate value of K, the rate of change in the sum of squared distance will slow down. Hence it will form an elbow on the graph. In mathematical terms, the aim of elbow method [65] is

$$arg_s min \sum_{i=1}^{k} \sum_{x \in S_\mathrm{i}} ||x - \mu_\mathrm{i}||^2$$

Figure 3.18 represents the percentage change in variance with changing value of K. At $K = 4$ the rate of change has slowed down compared to $K = 2, 3$ where change is variance is almost close to 20%

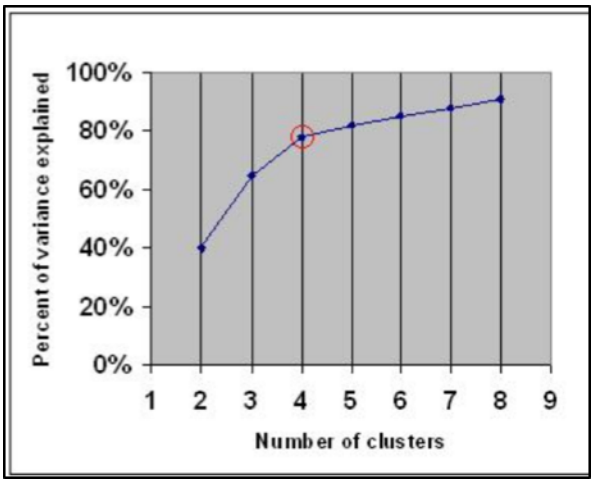


Figure 3.18: *Example of Elbow method for depicting appropriate number of K [18]*

Silhouette Method

The Silhouette analysis is used to analyze the distance between K clusters. It measures the distance between data points in a cluster with another cluster. The silhouette plot is used to visualize this closeness between clusters [47]. The measure used to perform this analysis is called Silhouette coefficients or score [48]. The Silhouette coefficients $s(i)$ or score value can be calculated using the following formula [47]:

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))}$$

where, $a(i)$ is the average distance between a data point and other data points within the same cluster, $b(i)$ is the minimum of an average of distance between a data point and other data points from other clusters. The silhouette score, ranges between -1 and 1 [48], where

- -1: data point may be incorrectly assigned
- 0: data point is at the boundary line between two clusters
- 1: data point is correctly assigned

Figure 3.19 represents a silhouette coefficient plot for $K = 3$. This graph can be read as follows:

1. This x-axis of the graph represents silhouette coefficient value between -1 to 1
2. The red dotted line represents the average of all the coefficient values
3. The 3 different colored boxes represent each cluster
4. The height of the boxes represents number of data points in each of these clusters

5. The curve at the end of each cluster represents different coefficient values of each data point. It is ordered from highest to lowest order

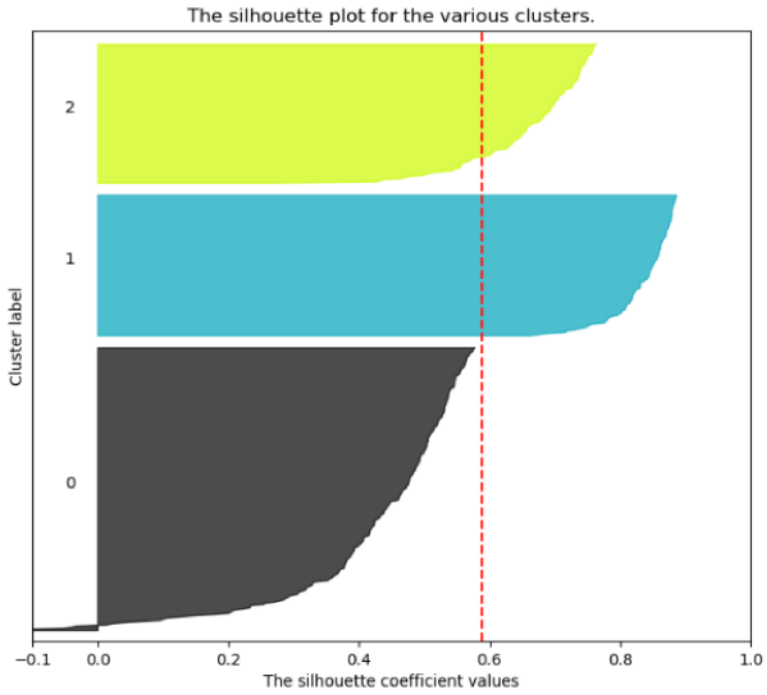


Figure 3.19: *Example of Silhouette analysis for depicting appropriate number of K [48]*

3.5 Metrics for Result Evaluation

3.5.1 Classification Metrics

1. **Confusion Matrix:** A confusion matrix is a square matrix ($n \times n$) where n is number of classes being projected [24] (see example in Figure 3.20). It is used to evaluate performance of a classification model. The observed data can be perceived in two ways: Actual and Predicted. Actual represents the original value of the data. Predicted represents how new data has been classified as. The confusion matrix defines the relationship between Actual and Predicted data using following terms:

Confusion Matrix		Target			
		Positive	Negative		
Model	Positive	TP	FP	Precision	TP/(TP+FP)
	Negative	FN	TN		TN/(FN+TN)
		Sensitivity or Recall	Specificity	Accuracy=(TP+TN)/(P+N)	
		TP/(TP+FN)	TN/(FP+TN)		

Figure 3.20: *Confusion Matrix [3]*

- **True Positive(TP):** When observed data is "True Positive" when actual class and predicted class, both are TRUE.
- **True Negative(TN):** When observed data is "True Negative" when actual class and predicted class, both are Negative.
- **False Positive(FP):** When observed data is "False Positive" when actual is "False" predicted class is Positive.
- **False Negative(FN):** When observed data is "False Negative" when actual is "False" predicted class is Negative.

2. **Accuracy [24]:** It gives the measure of accurately predicted data.

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

3. **Precision [17]:** Precision is a measure that describes the proportion of dataset that is predicted as "Positive" is actually "Positive".

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

4. **Recall or Sensitivity [17]:** Recall is a measure that describes the proportion of dataset that is predicted as "Positive" but is actually "Negative". This is also

known as True Positive Rate.

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

5. **Specificity [6]:** Specificity is a measure that describes the proportion of dataset that is predicted as "Negative" divided by total number of negatives. This is also known as True Negative Rate.

$$Specificity = \frac{TN}{TN + FN}$$

6. **False Positive Rate [6]:** Specificity is a measure that describes the proportion of dataset that is predicted as "False Positives" divided by the total number of negatives.

$$FalsePositiveRate = \frac{FP}{TN + FP} = 100 - Specificity$$

7. **F1 score:** The F1 score is the harmonic mean of Precision and Recall [17]. It is not appropriate to take arithmetic mean of Precision and Recall e.g., in a case where precision is 2% and recall is 98%. Mean will give value 50% which indicates the model must be good. But actually, the model is really bad as it is predicting 98% wrong values. Hence, the F1 score is calculated by evaluating the value of the harmonic mean.

$$F1Score = \frac{2 * Precision * Recall}{(Precision + Recall)}$$

8. **Area under ROC curve (AUC) [23]:** In a Receiver Operating Characteristic (ROC) curve, the "True Positive Rate" (sensitivity 4) is plotted against the

"False Positive Rate" for different cut-off Points. For example, a model classifies between two classes A or B based on a cutoff point equal to 0.5. It can also be interpreted as if $Y(x)$ (the test data point) value of the classification method is great that equal to 0.5 then classify test sample into class A else class B.

Figure 3.21 shows that a model is considered to be performing worse when the ROC curve is below the random classifier line (in red). A model performance gets better as TPR is closer to 1 for low FPRs (and even as FPR increases, TPR remains close to 1), hence maximizing the Area Under the Curve (AUC) . The metric that is used to measure the performance using ROC curve is AUC. The AUC value ranges between 0 and 1. The higher the AUC the better the performance of the model.

- 90-1 = excellent (A)
- .80-.90 = good (B)
- .70-.80 = fair (C)
- .60-.70 = poor (D)
- .50-.60 = fail (F)

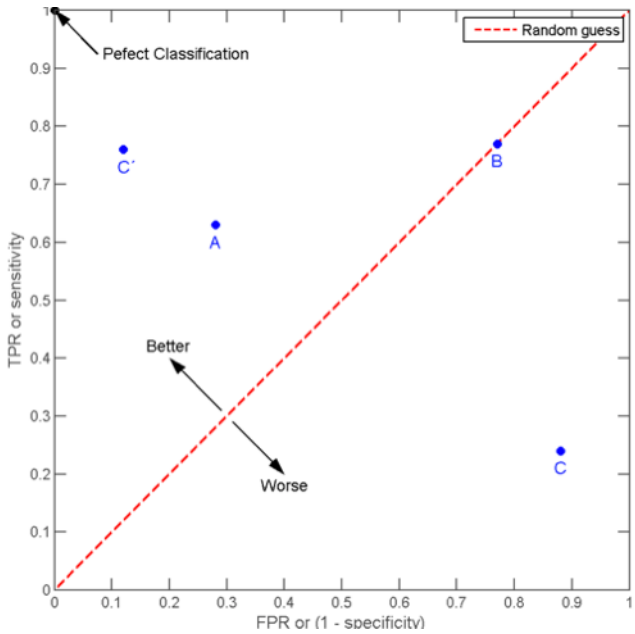


Figure 3.21: *Example of area under ROC curve [53]: The red line represents the ROC for a random classifier, i.e., 50% chance of classifying correctly. When the line curves following the (Better) arrow, the AUC increases, indicating better classifier performance.*

Chapter 4

Evaluation and Results

This Chapter covers the experiments, results and subsequent evaluation of the methodologies used in our research. First we will discuss Classification efforts and then Recommendation work.

4.1 Classification

For the Classification problem we focused on using kNN and ANN methodologies.

4.1.1 Selecting Best k Value for kNN

In kNN, we aim to find the optimal number of neighbors that produce better classification results. Figure 4.1 represents a plot of Accuracy (black), F1-score (red), Precision (blue) and Recall (green) for varying values of k, where k represents the number of k nearest neighbors to be considered for classification. This graph helps us identify the optimum value of k that can be used for classification. In this case, experiments showed that the optimal value was when k was 2 or 3. The performance of the model degrades drastically after the value of k exceeded 8. This is likely a result of many

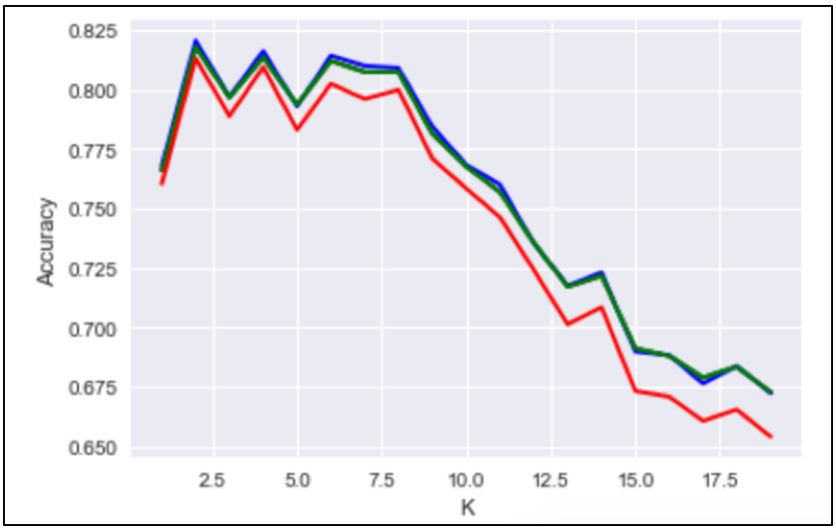


Figure 4.1: *Representation of classification metrics for kNN*

subtle differences in the various features that identify one observation, which made it difficult for the model to capture similarities (close distances) between observations.

4.1.2 Selecting Best Combination of Hidden Layers and Number of Neurons for ANN

In terms of ANN, the focus of our experiments was to determine a neural network configuration that yields best classification results. We varied number of hidden layers and hidden nodes. Figure 4.2 illustrates results from various experiment performed to obtain optimum values of hidden layers and nodes in hidden layers for ANN. Each plot represents the change in accuracy as the number of hidden layers was increased with different number of neurons. These trends fluctuate a lot, particularly for small number of neurons (e.g., 5 and 10), so results are unclear and hard to interpret; there is no clear relation between the accuracy and the number of hidden layers, a variation that gives unstable accuracy. When the number of neurons was set to 15 and 20, the plots seem to be more stable particularly beyond 20 hidden layers. Still, the accuracy

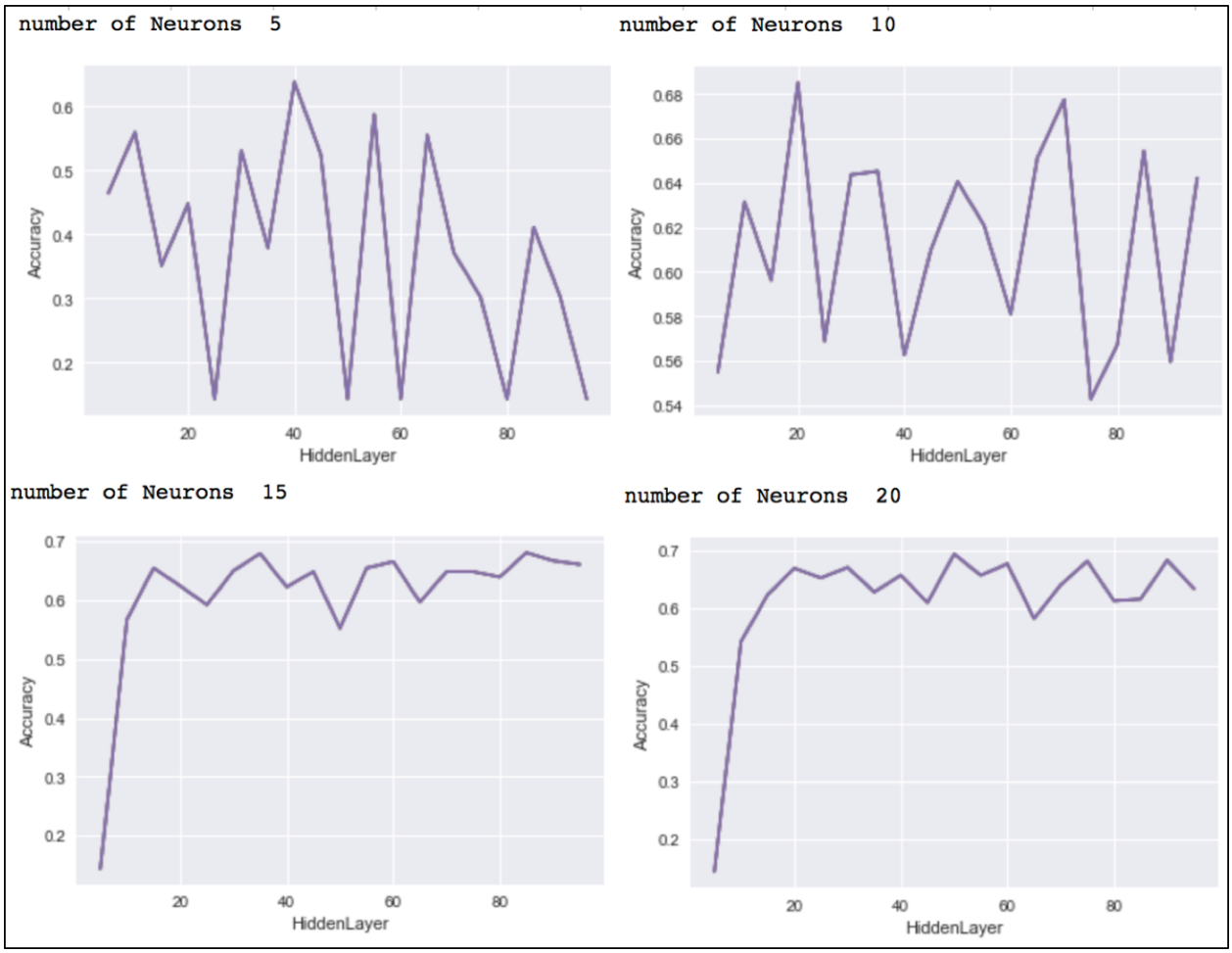


Figure 4.2: *Accuracy of our ANN before min-max scaling, varying the number of neurons and increasing the number of hidden layers.*

of the classifier is around 70%.

To stabilize the spikes in the plot, min-max scaling is performed (discussed in Chapter 3 under Data Transformation). Figure 4.3 shows the plots after the scaling, displaying a better relationship between accuracy and hidden layers. The optimum value of hidden layers is 50 and number of nodes is 10.

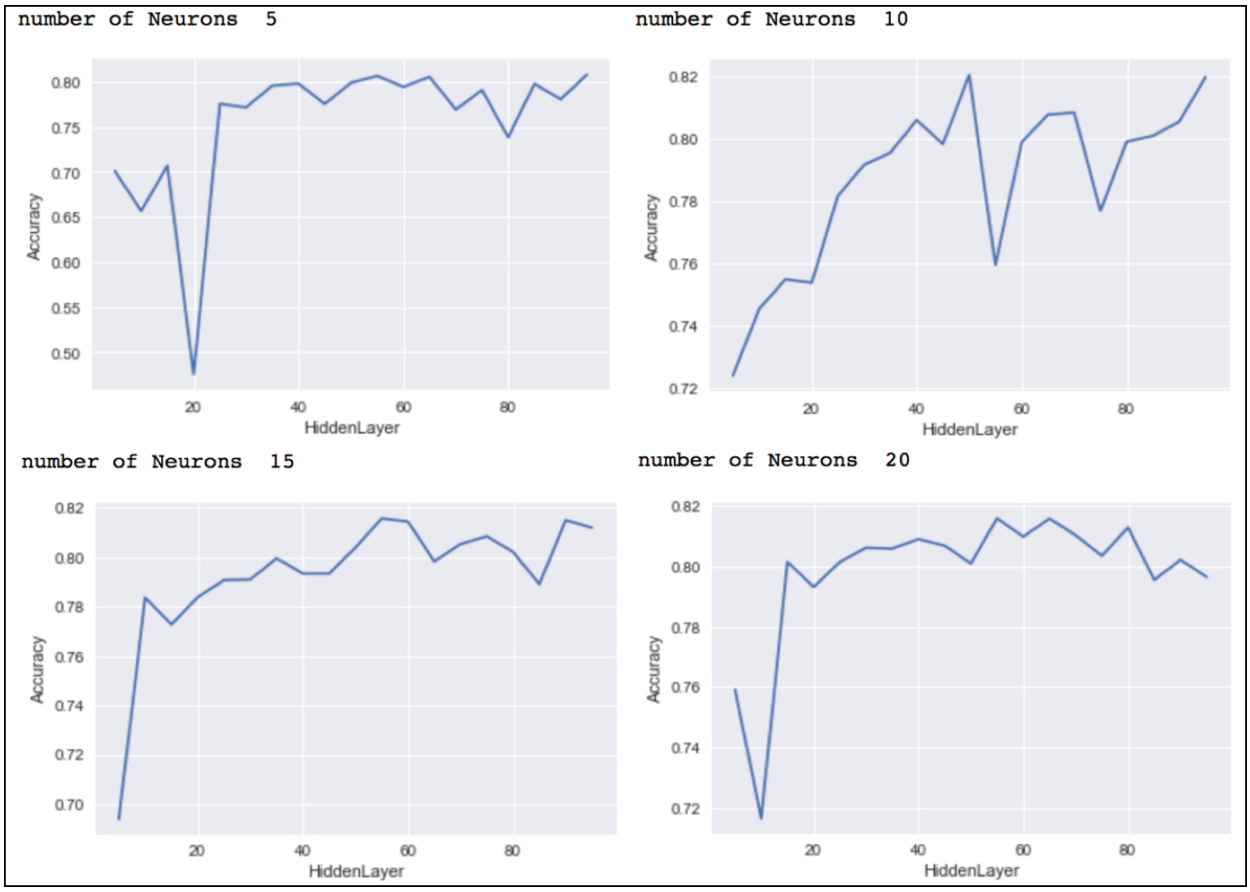


Figure 4.3: *Accuracy of our ANN after min-max scaling, varying number of neurons and increasing number of hidden layers.*

4.1.3 Experiments and Results

Several experiments were conducted using the available dataset. The goal was to try three different classification strategies Logistical Regression, kNN and ANN to see which one can provide a suitable model for our problem of classifying study plans. The classes we are considering correspond to each of the possible rankings -1, and 1 through 5, that way a plan will be categorized as incorrect, or how "good" it is.

In this section we present the outcomes of each classifying strategy using confusion matrices and ROC and Precision and Recall curves. The confusion matrices show the

		Confusion Matrix - Neural Network						
		Predicted Ratings						
		-1	1	2	3	4	5	Recall
ACTUAL	-1	94	0	0	0	0	0	100.00
	1	0	107	0	0	0	0	100.00
	2	0	7	105	0	0	0	93.75
	3	0	2	26	85	11	1	68.00
	4	0	1	2	24	50	32	45.87
	5	0	0	0	0	11	96	89.72
	Precision	100.00	91.45	78.95	77.98	69.44	74.42	82.11

Figure 4.4: *Confusion Matrix for Artificial Neural Network*

breakdown of True Positive rates per class, while the ROC and Precision and Recall curves will help us determine overall performance of each classifier for each class.

Classification Using ANN

Figure 4.4 illustrates the resulting Confusion Matrix for the ANN model used in our experiments. Interesting information can be drawn from this table. First of all, note that the highest values are associated with class -1, which is the category we have use to denote study plans that are incorrect. Values for class 1 are also high. This implies that plans rated as incorrect (unusable) or very poor, are rather easy to identify. When we move to classes corresponding to rankings of 3, 4, 5, we see the classifier performance decrease, particularly for class 4. This result is intuitive because, a ranking of 4 can be very close to either a ranking of 3 or a ranking of 5, i.e., very small changes to a plan can make a human decide to rank a plan a whole point higher or lower. The overall classification accuracy achieved for this model is 82.11%.

We now look at ROC curve in Figure 4.5. Compared to other models, the trends

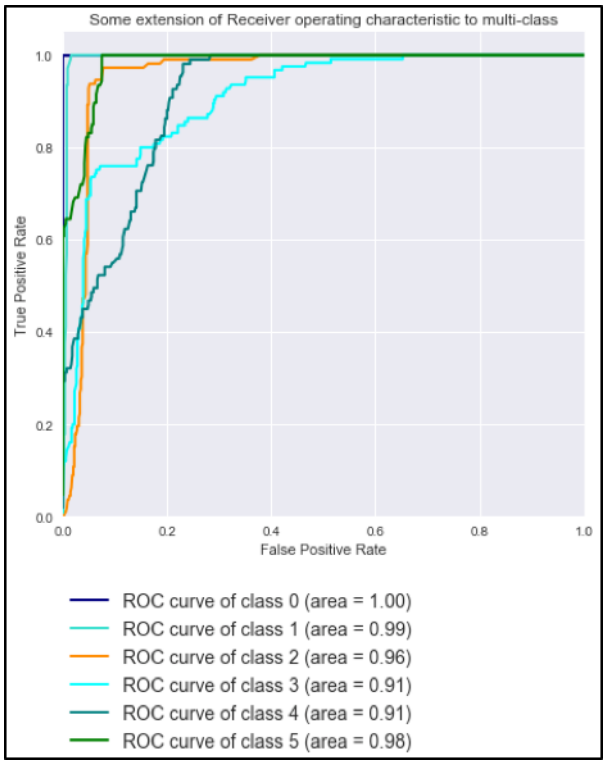


Figure 4.5: *ROC curve for ANN*

for ANN indicate that this model performed better than the other models. The AUC is above .9 for all the classes. This curve also highlights individual class performance in more detail. This model has performed well for class with Rating -1,1,5 and performed reasonably for class with ratings 2,3,4.

Classification Using Logistic Regression

Figure 4.6 represents the Confusion Matrix for Logistic Regression model. The confusion matrix shows that this model did not perform well for any class besides the ranking of -1. However, it also shows a similar trend as the one for ANN, where the poorest performance is for classes 3, 4, 5. Accuracy for this model is 58.06%.

Regarding ROC trends, it can be observed from figure 4.7 that Logistic regression performed well for plans with ratings in (-1,1,5), it performed fairly for plans with

		Confusion Matrix - Logistic Regression						
		Predicted Ratings						
		-1	1	2	3	4	5	Recall
ACTUAL	-1	92	0	0	0	1	1	97.87
	1	0	93	3	2	3	6	86.92
	2	0	33	62	2	11	4	55.36
	3	0	20	36	22	22	22	18.03
	4	1	3	14	16	32	43	29.36
	5	0	14	3	4	9	77	71.96
	Precision	98.92	57.06	52.54	47.83	41.03	50.33	58.06

Figure 4.6: *Confusion Matrix for Logistic Regression*

ratings (2,4) and performed poorly for plans with rating of 3. These plots are very useful for visually comparing the relative performance of the classes. In our experiment, this plot indicates that model did not perform well for classes 2,3,4 but fairly for classes 1,5 and best for the class -1.

Classification using kNN

We now discuss results for the kNN strategy. Figure 4.8 represents the Confusion Matrix for kNN model. Similar to the other models, predictions for classes -1, 1 are the best. Worst prediction is for class 4 followed by classes 2, 3 and 5. In this case there is a more noticeable difference between classes 4 and 5, suggesting that with this strategy it might be easier to correctly identify plans ranked as 5 compared to those ranked as 4 (hence, this model is able to better capture similarities across plans in class 5). The accuracy for this model is 81.80%.

Looking at the trends from ROC curve, Figure 4.9 suggests that kNN performed better than the Logistic Regression model and similar to ANN in a multi-class classi-

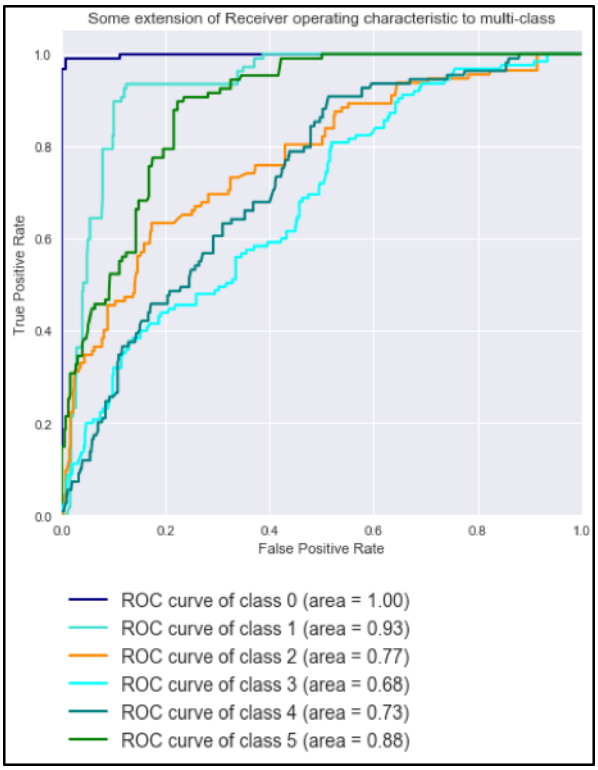


Figure 4.7: *ROC curve for Logistic Regression*

fication problem. The AUC is above 0.9 for classes with rankings -1,1,2,5 and above 0.8 for classes with ratings 3,4. We have observed similar behaviors as the other two classification strategies.

4.2 Recommendation System

In this section we discuss the experiments and results related to the recommendation system strategy for choosing best suitable study plans. The main strategy for recommending plans was to use a similarity based approach, i.e., based on the euclidean distances between observations – those samples that are more similar, will have the shortest distances between them. For each data point (observation), the multi-dimensional distance from every other point was calculated to identify clusters

Confusion Matrix - K-Nearest Neighbor								
		Predicted Ratings						
		-1	1	2	3	4	5	Recall
ACTUAL	-1	94	0	0	0	0	0	100.00
	1	0	107	0	0	0	0	100.00
	2	0	7	105	0	0	0	93.75
	3	1	1	30	83	7	3	66.40
	4	0	0	4	28	63	14	57.80
	5	0	0	0	0	24	83	77.57
	Precision	98.95	93.04	75.54	74.77	67.02	83.00	81.80

Figure 4.8: *Confusion Matrix for kNN*

of observations that are close together (interpreted as similar). Clusters represent sets of training samples similar to each other based on the values of input features/metrics. The premise is that points located close in space would fall closer together in space, so it would be easier for a recommendation system to find which cluster a new point (new plan) belongs to based on its relative position to every other previously identified cluster.

To depict the results from recommendation experiments, we discuss the use of Elbow Method and associated plots [7] as well as Silhouette plots to identify clustering [48] as discussed in section 3.4.2.

The Elbow Method is used with K-means clustering to find out appropriate value for K, i.e., optimum number of clusters for our model. Figure 4.10 is a typical trend used for the Elbow Method using our dataset. The plot represents sum of squared distances between data points and the closest cluster center while changing values of K (refer section 3.4.2). This plot is helpful for finding appropriate number of clusters in a dataset. The trend in this figure lacks the "elbow", which is a pronounced bend in the lower-left corner of the plot (towards the origin); this indicates that these results

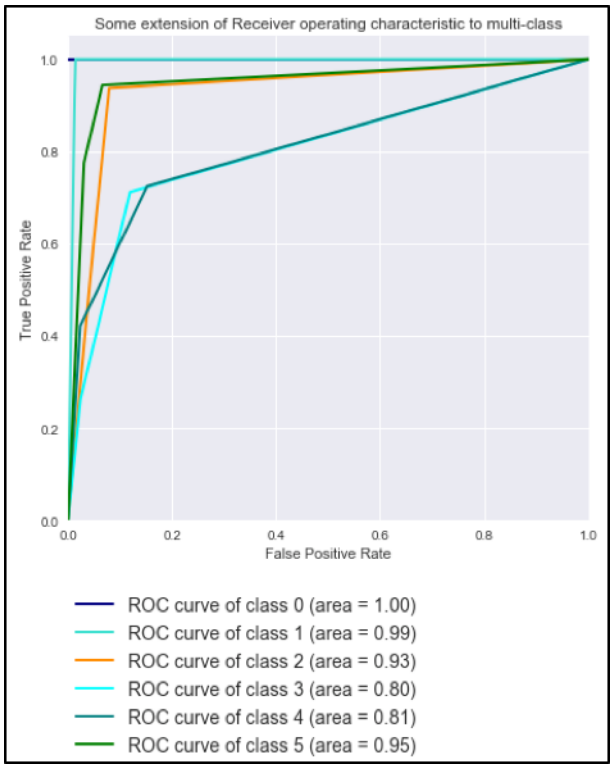


Figure 4.9: *ROC curve for kNN*

do not display any appropriate value for K. The "elbow" represents a point with a sharp variance which is an ideal cluster value. The assumption is that the dataset has a majority of features that have values ranging between 0 and 1. It would not be possible for K-means algorithm to identify clusters with appropriate distance between their mean values.

The Silhouette analysis (also discussed in section 3.4.2) is another method used to measure how close each point in one cluster is to the points in another cluster. Silhouette coefficients uses values between -1 and 1, where:

- 1 indicates that points are far away from neighboring clusters
- 0 indicates that points are close to decision boundaries
- -1 represents that a point may be assigned to an incorrect cluster

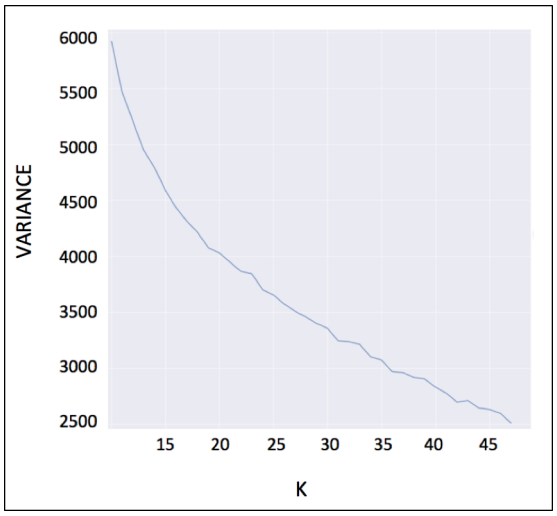


Figure 4.10: *Elbow method to find out an appropriate number of clusters. The x-axis represents a change in variance [18] with changing number of K values that represents y-axis.*

The experiment was carried out for K values (clusters) between 1 and 45. An initial silhouette analysis is presented in Figure 4.11, using cluster value score of 24. We can see from these figures that for all the clusters, the average Silhouette score is closed to 0 which indicates that existing clusters are very close to each other. Additionally, we can observe that the value of several clusters is less than 0 which means data points may have been assigned to wrong clusters. The *silhouette coefficient* is below 0.2, which indicates that these are not good results because the clusters may be confused with each other.

To try improve clustering performance, and hence recommendation, we performed another experiment where we associated a specific weight to each feature according to its relevance (discussed in Section 3.4). The weights based on relevance were different for each feature. For example, an input preference for taking "courses in summer quarter" has less relevance compared to feature "Student type" (enrollment).

The idea behind assigning different weights to different feature values is that it should help increase the difference between clusters, which is the goal of our recom-

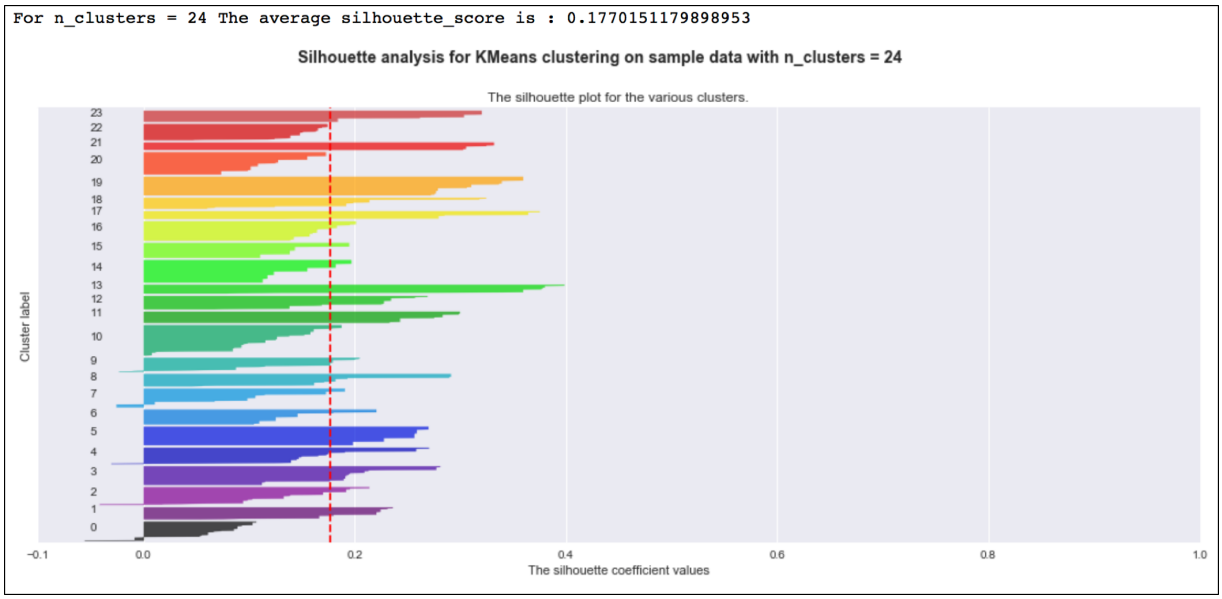


Figure 4.11: *Silhouette Analysis for clusters $K = 20$ and $K = 24$. The x-axis represents silhouette coefficient value between -1 to 1 and y-axis represents labels for clusters.*

mendation system. A recommendation engine should be able to identify similar inputs without confusing them as identical.

Figure 4.12 represents the elbow method plot [7] using K-means clustering with the weighted features. This figure depicts a pronounced elbow curve at $K = 12$, and after $K = 20$, the trend becomes close to a flat line, showing that the ideal value for K (number of clusters) is a range somewhere between 12 and 20.

We now perform a silhouette analysis on the weighted features for K-means clustering. This analysis shows that a good number of clusters in our dataset is $K = 24$, for which the average silhouette coefficient is 0.96 (see Figure 4.13), which indicates that the data points are very well clustered using 24 groups (within each group, observations are closely related, but different) [47]. This is a reasonable strategy for recommendation since it allows for plans to be recommended based on their similarity to specific clusters.

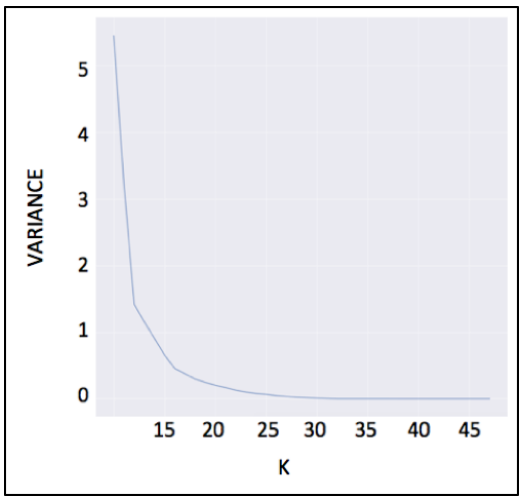


Figure 4.12: *Elbow plot showing a pronounced "elbow" trend. The x-axis represents a change in variance [18] with changing number of K values that represents y-axis.*

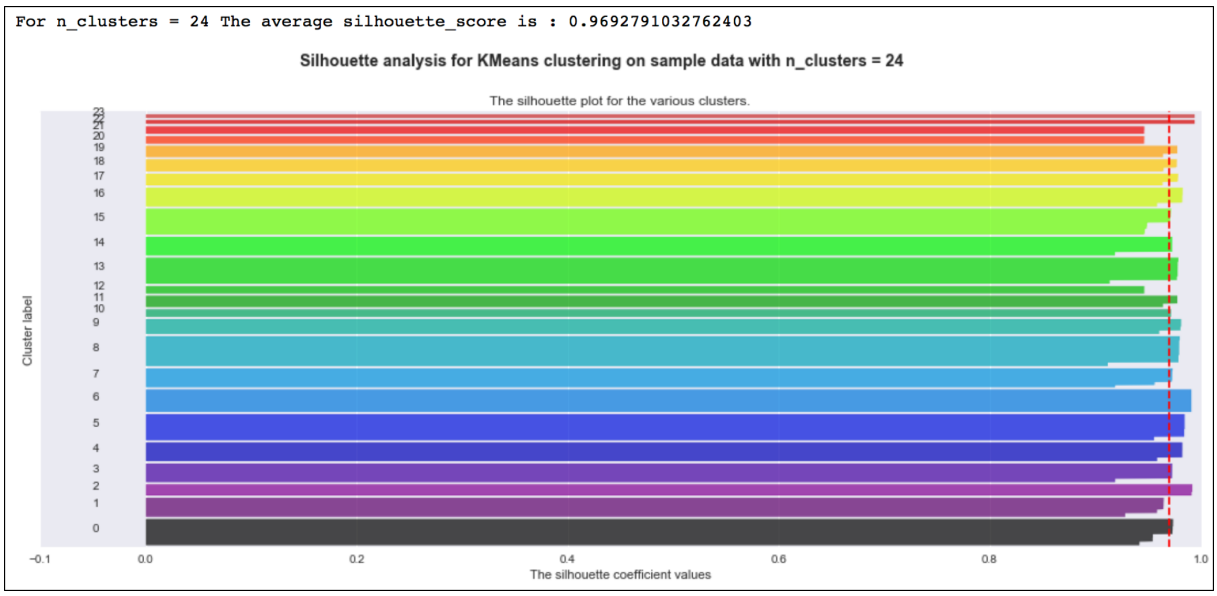


Figure 4.13: *Silhouette Analysis for cluster $K = 24$. The x-axis represents silhouette coefficient value between -1 to 1 and y-axis represents labels for clusters*

Chapter 5

Conclusions and Future Work

In this section, we describe our research achievements so far. We then discuss limitations and challenges encountered during our work. Finally, present possible improvements for future work.

5.1 Conclusions

Based on the experiments discussed in Chapter 4, we can observe some common trends across the different strategies.

Feature Analysis Takeaways

In addition, we found these interesting highlights from the feature analysis process:

- Feature selection techniques: were helpful in improving the accuracy of classifiers, as well as authenticity of results. The first indicator of feature issues was identified from an initial heat map figure, which showed suspicious correlations between some of the features. Some initial features that were either "painfully" redundant, noisy or statistically useless were removed during the feature selection process.

- Univariate analysis: an interesting takeaway from the univariate feature analysis is that when the feature "Student Enrollment Type" is "Full Time", it makes it more likely for a plan to be ranked as "Good". While this was not intuitive from a theoretical perspective, we have confirmed this with faculty/advisor users, who asserted that making plans for Full Time students is easier, while Part Time students are usually difficult and ambiguous.

- Bivariate analysis: an intuitive correlation was confirmed, students who work part time, tend to prefer taking classes later in the day, and are usually Part Time students. However, as mentioned before, we don't have much accurate data for Part Time students, so while intuitive, we should carefully weigh this conclusion as it may be biased based on the plans we used for training.

Classification Takeaways

These results provide information regarding both about the classifiers and our dataset. Table 5.1 summarizes our experimental results across classifiers in terms of accuracy.

Some of the highlights from the our experiments are as follows:

- Ability to classify incorrect and poor plans: The fact that our classifying experiments showed that we can easily categorize an incorrect plan (ranking -1) can be very useful as it shows that ML approach could be used for discarding erroneous plans without having to run exhaustive testing (which can be expensive). It is also easy to classify a poor ("Bad") plan, e.g., with ranking of 1 or 2).

- Ability to classify correct plans: On the other hand, classifying "better" plans with rankings 3, 4, 5 is more challenging. This might be a result of the specific dataset that we have used for training. Since these classes are too close to each other it might take a considerable amount of additional examples to distinguish

Table 5.1: Classification Summary (Metrics were described in Section 4.1.3)

Classification Metrics	**Artificial Neural Network**	**Logistic Regression**	**K Nearest Neighbor**
Accuracy	82.11	58.06	81.80
F1 Score for Rating -1	100.00	98.40	99.47
F1 Score for Rating 1	95.54	68.89	96.40
F1 Score for Rating 2	85.71	53.91	83.67
F1 Score for Rating 3	72.65	26.19	70.34
F1 Score for Rating 4	55.25	34.22	62.07
F1 Score for Rating 5	81.36	59.23	80.19

between these categories. It may also indicate that a better grouping for plan classification could be used instead of the 5-star ranking, for instance, use instead a 3-star raking or simply "Incorrect", "Bad" and "Good". Generally speaking, even a classification strategy with accuracy of 70% might provide plans that are a good starting point for a faculty/advisor or student to build a suitable study plan instead of starting from scratch.

- Computation time: In terms of computational complexity, the most expensive strategies are ANN and kNN, however, they are also the best performing. Hence, there is a trade off between computation time and classifying accuracy that should be taken into account. As a problem scales up in size, kNN tends to become unmanageable in terms of computation time. It is worth mentioning that none of the tools used in our experiments are particularly optimized for High Performance Computing behavior, and the hardware we have used are personal laptops, so it is

plausible to achieve better time-perfomance with appropriate code optimization and/or hardware.

Recommendation Strategy Takeaways

- Implementation of Elbow method and Silhouette analysis yielded an optimum value of number of clusters that could be used for reasonably efficient practical recommendation. Both strategies used performed well and provided a similar range of values for clusters, confirming the intuition behind some of our assumptions and understanding of the data. This also shows that our strategy for recommendation is reasonable and can be implemented and integrated into a larger recommendation framework. From the two methods explored, the application of Silhouette method produced more reliable results as compared to the Elbow method. Both of these methods have similar computational complexity and could be integrated into the larger system's software framework.

5.2 Limitations and Challenges

The majority of the limitations and challenges in this research are related to our dataset. Our starting dataset had very limited number of samples and it was also quite noisy and with many "impurities". The data consisted of less-than-a-hundred manually-created study plans, using an archaic tool, and as such, these plans had a lot of human errors that we had to manually process to make suitable for our tools. As we started working on this research we had to conduct several information sessions with EvCC faculty to understand the data and how to best sanitize it. Besides cleaning the data, our biggest challenge was to devise a strategy to artificially (yet realistically) grow the dataset so that we could appropriately apply ML approaches and

train Classification and Recommendation models. The artificially generated data is based on rigorous data variance. For instance, for each metric, we determined a range of acceptable values for each specific class. Random Numbers were generated within these ranges. The logic to determine the ranges was supervised by faculty advisor from EvCC to ensure data authenticity.

Other challenging aspects related to the data are related to the type of information contained in the initial sample, for instance, there is no information regarding students who are not starting with intro-level courses (i.e., no plans were provided for students who did not have to take initial classes in a sequence like Math), and hence, their plans would be a lot shorter. In addition, to protect student privacy, EvCC did not provide any personal student information like their GPAs or course evaluation information. Study of these GPAs in relation to core courses could be very useful to help analyze which courses are harder – an algorithm can learn from this kind of data to not recommend certain combinations of courses that increase the risk of students failing or dropping out.

5.3 Future Work

From a big-picture approach, an immediate need is to fully integrate the Recommendation Module (ML approach) with the rest of the VAA software system (UI and database). This will enable the generation and storage of more user-created study plans, and hence, more data for training and tuning the ML recommendation engine. In this integration, we should also consider a Testing Module component that checks validity of recommended study plans (i.e., checks for critical errors like matching basic preferences requirements).

In terms of ML strategies research, sanitizing existing data and growing the dataset

are imperative for further tuning and exploration. The creation and/or automatic-generation of more data involves not only adding more observations but also new types of features that can provide additional information and possibly improve classification results. Some examples of future features that will be considered are: different starting points (e.g., taking into consideration placement into courses that are not entry level), distribution of core courses in quarters of interest (e.g., first quarters should always have a high density of core courses), breaks in relevant course sequences (e.g., math sequences are long and it is recommended that is taken back-to-back in consecutive quarters). Some of these future metrics will help improve the knowledge of whether a recommended plan is "Good" or "Bad".

Overall, talking about additional data, we should further validate our strategy for artificial data generation. This will also allow for testing scalability and extensibility of the proposed strategies using larger amounts of data.

Bibliography

[1] https://www.researchgate.net/figure/Sample-of-a-feed-forward-neural-network_fig1_234055177. 2018.

[2] *3.1. Cross-validation: evaluating estimator performance — scikit-learn 0.19.1 documentation*. http://scikit-learn.org/stable/modules/cross_validation.html. (Accessed on 05/31/2018).

[3] *7 Important Model Evaluation Error Metrics Everyone should know*. https://www.analyticsvidhya.com/blog/2016/02/7-important-model-evaluation-error-metrics/. (Accessed on 05/28/2018).

[4] *An introduction to seaborn*. https://seaborn.pydata.org/introduction.html. 2018.

[5] *Artificial Neural Networks Currently Used*. https://analyticsindiamag.com/6-types-of-artificial-neural-networks-currently-being-used-in-todays-technology/. 2018.

[6] *Basic evaluation measures from the confusion matrix*. https://classeval.wordpress.com/introduction/basic-evaluation-measures/. 2018.

[7] Purnima Bholowalia and Arvind Kumar. "EBK-means: A clustering technique based on elbow method and k-means in WSN". In: *International Journal of Computer Applications* 105.9 (2014).

[8] Horst Bischof, Werner Schneider, and Axel J Pinz. "Multispectral classification of Landsat-images using neural networks". In: *IEEE transactions on Geoscience and Remote Sensing* 30.3 (1992), pp. 482–490.

[9] Peter Brucker. "The job-shop problem: Old and new challenges". In: *Proc. of the 3rd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA)*. 2007, pp. 15–22.

[10] Gates Bryant. "Driving toward a Degree: The Evolution of Planning and Advising in Higher Education". In: *Tyton Partners paper, August* 28 (2015).

[11] Maico Cassel and F Lima. "Evaluating one-hot encoding finite state machines for SEU reliability in SRAM-based FPGAs". In: *On-Line Testing Symposium, 2006. IOLTS 2006. 12th IEEE International*. IEEE. 2006, 6–pp.

[12] J. Chapmann. *Machine Learning: Fundamental Algorithms for Supervised and Unsupervised Learning With Real-world Applications*. CreateSpace Independent Publishing Platform, 2017. ISBN: 9781548307752. URL: https://books.google.com/books?id=4Oc9tAEACAAJ.

[13] *Chapter 9 Recommendation Systems*. http://infolab.stanford.edu/~ullman/mmds/ch9.pdf. 2018.

[14] Heng-Tze Cheng et al. "Wide & deep learning for recommender systems". In: *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM. 2016, pp. 7–10.

[15] *Class Imbalance Problem*. http://www.chioka.in/class-imbalance-problem/. 2018.

[16] Edward Grady Coffman and John L Bruno. *Computer and job-shop scheduling theory*. John Wiley & Sons, 1976.

[17] Jesse Davis and Mark Goadrich. "The relationship between Precision-Recall and ROC curves". In: *Proceedings of the 23rd international conference on Machine learning*. ACM. 2006, pp. 233–240.

[18] *Elbow method (clustering) - Wikipedia*. https://en.wikipedia.org/wiki/Elbow_method_(clustering). (Accessed on 05/28/2018).

[19] *Finding help — scikit-learn 0.19.1 documentation*. http://scikit-learn.org/stable/tutorial/statistical_inference/finding_help.html. (Accessed on 05/28/2018).

[20] Ian Goodfellow et al. *Deep learning*. Vol. 1. MIT press Cambridge, 2016.

[21] *Guided Pathways*. https://www.sbctc.edu/colleges-staff/programs-services/student-success-center/guided-pathways.aspx. 2018.

[22] Martin T Hagan, Howard B Demuth, Mark H Beale, et al. *Neural network design*. Vol. 20. Pws Pub. Boston, 1996.

[23] James A Hanley and Barbara J McNeil. "The meaning and use of the area under a receiver operating characteristic (ROC) curve." In: *Radiology* 143.1 (1982), pp. 29–36.

[24] AM Hay. "The derivation of global estimates from a confusion matrix". In: *International Journal of Remote Sensing* 9.8 (1988), pp. 1395–1398.

[25] Georgef Hepner et al. "Artificial neural network classification using a minimal training set- Comparison to conventional supervised classification". In: *Photogrammetric Engineering and Remote Sensing* 56.4 (1990), pp. 469–473.

[26] *Hobsons Starfish Retention Solutions*. https://www.starfishsolutions.com/. 2015.

[27] *imblearn.over_sampling.RandomOverSampler*. http://contrib.scikit-learn.org/imbalanced-learn/stable/generated/imblearn.over_sampling.RandomOverSampler.htm. 2018.

[28] Gareth James et al. *An introduction to statistical learning*. Vol. 112. Springer, 2013.

[29] *K-MEANS*. http://scikit-learn.org/stable/modules/clustering.html. 2018.

[30] *Logistic Regression for Machine Learning*. https://machinelearningmastery.com/logistic-regression-for-machine-learning/. 2018.

[31] *Missing data - Wikipedia*. https://en.wikipedia.org/wiki/Missing_data. (Accessed on 05/28/2018).

[32] *ML Language Shoot out*. https://cdn-images-1.medium.com/max/1600/0*BZG-twC8uvwtnH91.jpg. 2018.

[33] Nasser M Nasrabadi. "Pattern recognition and machine learning". In: *Journal of electronic imaging* 16.4 (2007), p. 049901.

[34] Daniel A Newman. "Missing data: Five practical guidelines". In: *Organizational Research Methods* 17.4 (2014), pp. 372–411.

[35] Guobin Ou and Yi Lu Murphey. "Multi-class pattern classification using neural networks". In: *Pattern Recognition* 40.1 (2007), pp. 4–18.

[36] *Outlier*. https://en.wikipedia.org/wiki/Outlier. 2018.

[37] Sinan Ozdemir and Divya Susarla. *Feature Engineering Made Easy: Identify unique features from your dataset in order to build powerful machine learning systems*. Packt Publishing, 2018. URL: https://www.amazon.com/Feature-Engineering-Made-Easy-Identify-ebook/dp/B077N6MK5W?SubscriptionId=

0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=B077N6MK5W.

[38] Justin D Paola and Robert A Schowengerdt. "A detailed comparison of back-propagation neural network and maximum-likelihood classifiers for urban land use classification". In: *IEEE Transactions on Geoscience and remote sensing* 33.4 (1995), pp. 981–996.

[39] Jennie Pearce and Simon Ferrier. "Evaluating the predictive performance of habitat models developed using logistic regression". In: *Ecological modelling* 133.3 (2000), pp. 225–245.

[40] *Peoplesoft Degree Audit*. https://www.sbctc.edu/resources/documents/colleges-staff/it-support/sms/smsoverview.pdf. 2012.

[41] *Preprocessing data*. http://scikit-learn.org/stable/modules/preprocessing.html. 2018.

[42] William F Punch III et al. "Further Research on Feature Selection and Classification Using Genetic Algorithms." In: *ICGA*. 1993, pp. 557–564.

[43] *Python :: Anaconda Cloud*. https://anaconda.org/anaconda/python. (Accessed on 05/28/2018).

[44] Oliver C Robinson. "Sampling in interview-based qualitative research: A theoretical and practical guide". In: *Qualitative Research in Psychology* 11.1 (2014), pp. 25–41.

[45] Toby Segaran. *Programming collective intelligence: building smart web 2.0 applications.* " O'Reilly Media, Inc.", 2007.

[46] *Selecting good features – Part IV: stability selection, RFE and everything side by side — Diving into data*. https://blog.datadive.net/selecting-good-features-part-iv-stability-selection-rfe-and-everything-side-by-side/. (Accessed on 05/31/2018).

[47] *Selecting the number of clusters with silhouette analysis on KMeans clustering — scikit-learn 0.19.1 documentation*. http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html. (Accessed on 05/28/2018).

[48] *Silhouette (clustering) - Wikipedia*. https://en.wikipedia.org/wiki/Silhouette_(clustering). (Accessed on 05/28/2018).

[49] *Statistics How To*. http://www.statisticshowto.com/univariate/. 2018.

[50] Florian Strub and Jeremie Mary. "Collaborative filtering with stacked denoising autoencoders and sparse inputs". In: *NIPS workshop on machine learning for eCommerce*. 2015.

[51] N Suguna and K Thanushkodi. "An improved k-nearest neighbor classification using genetic algorithm". In: *International Journal of Computer Science Issues* 7.2 (2010), pp. 18–21.

[52] *Supervised learning*. https://en.wikipedia.org/wiki/Supervised_learning. 2018.

[53] *The Area Under an ROC Curve*. http://gim.unmc.edu/dxtests/roc3.htm. 2018.

[54] Oliver Theobald. *Machine Learning for Absolute Beginners: A Plain English Introduction*. Independently published, 2017. ISBN: 152095140X. URL: https://www.amazon.com/Machine-Learning-Absolute-Beginners-Introduction/

dp/152095140X?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=152095140X.

[55] *Top 15 Python Libraries for Data Science in 2017*. https://medium.com/activewizards-machine-learning-company/top-15-python-libraries-for-data-science-in-in-2017-ab61b4f9b4a7. 2018.

[56] Paul J Werbos. "Backpropagation through time: what it does and how to do it". In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560.

[57] Stephen Gang Wu et al. "A leaf recognition algorithm for plant classification using probabilistic neural network". In: *Signal Processing and Information Technology, 2007 IEEE International Symposium on*. IEEE. 2007, pp. 11–16.

[58] Ting-Fan Wu, Chih-Jen Lin, and Ruby C Weng. "Probability estimates for multi-class classification by pairwise coupling". In: *Journal of Machine Learning Research* 5.Aug (2004), pp. 975–1005.

[59] Takeshi Yamada and Ryohei Nakano. "Job shop scheduling". In: *IEE control Engineering series* (1997), pp. 134–134.

[60] Jihong Yan and Jay Lee. "Degradation assessment and fault modes classification using logistic regression". In: *Journal of manufacturing Science and Engineering* 127.4 (2005), pp. 912–914.

[61] Alice Zheng and Amanda Casari. *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. O'Reilly Media, 2018. ISBN: 1491953241. URL: https://www.amazon.com/Feature-Engineering-Machine-Learning-Principles/dp/1491953241?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=1491953241.

[62] Tianqi Zhou, Lina Chen, and Jian Shen. “Movie Recommendation System Employing the User-Based CF in Cloud Computing”. In: *Computational Science and Engineering (CSE) and Embedded and Ubiquitous Computing (EUC), 2017 IEEE International Conference on*. Vol. 2. IEEE. 2017, pp. 46–50.