Classify

Period of Performance: October, 2023 – May, 2024

Technical Point of Contact: Kazi Ruslan Rahman

Electrical Engineering

School of Engineering

San Francisco State University

1600 Holloway Ave. San Francisco, CA 94132

Email: rahmankaziruslan@gmail.com

Advisor: Hao Jiang, Professor

School of Engineering

San Francisco State University

1600 Holloway Ave. San Francisco, CA 94132

Phone: (415) 338-6379; Fax: (415) 338-0525

Email: jianghao@sfsu.edu

Table of Contents

Abstract

What is the project?

Classify is a program that will create a catalog of classes that is optimized to fit into students' and professors' schedules. The primary goal of this project is to develop a user-friendly scheduling system tailored for academic departments, emphasizing the needs of the Engineering Department. It aims to overcome challenges in semester class scheduling due to the diverse needs and course requirements of a large student body. Classify will minimize the amount of time departments spend on creating schedules that both fit within professors' and students' availability.

What is the approach?

The approach is to create a program with simple and intuitive UI/UX that allows any administrator to create an entire catalog of classes easily. The front-end of the project will be quick and responsive. User experience of the application will be of priority ensuring every user can easily use the application. The system is designed to accept inputs such as the sequence of classes and designated teaching faculty. The approach we will use is requirements, system architecture, and system algorithms.

For the requirements, we will first understand specific needs and functionalities such as organizing course time, prerequisites, classes already taken, requirements, faculty available, and more. Then for system architecture, we will use a three-tier structure with a web-based front end (React), a backend in Python (Django), and a Postgresql database. For the system algorithm, we are going to incorporate algorithm models and follow simple if-else if, else logic to create optimal outcomes and use simple supervised learning to improve outcomes.

Why is your approach unique?

Our approach is unique because we are using an algorithm to optimize the catalog of classes so students' and professors' needs are met. The algorithm will be able to make changes to the schedule it creates when different conditions are inputted, this will lead to an application that is efficient and easily scalable. Changing something within the input should be easy and lead to an output (updated schedule) that is created/updated quickly.

Why is it interesting/important?

This project is important because it reduces the time that administrators would spend manually creating the schedule. The motivation behind Classify stems from the observed challenges in semester class scheduling, particularly given the diverse needs of a large student body, each with unique prerequisites and course requirements. The system is designed to accept inputs such as the sequence of classes, designated teaching faculty, current semester schedules, and specific class timings. We aim to

utilize student schedule data to train an algorithm, enabling it to discern patterns and commonalities. This data-driven approach will assist in predicting and formulating optimal class schedules for subsequent semesters. By doing so, the objective is not just to create schedules, but to optimize them in a way that minimizes conflicts and maximizes course availability.

In addition to student schedule optimization, Classify will consider professor availability, ensuring that the generated schedules are practical and feasible from an instructor's perspective as well.

In essence, It's scheduling with modern data-driven techniques, aiming for a more efficient and conflict-free academic scheduling process. Through this project, we hope to offer a tangible solution to an enduring challenge in academic settings, demonstrating the potential of integrating engineering principles with real-world applications.

Section 1. Introduction

Section 1.1 Project Goal

A simple intuitive website application that allows administrators to create the entire catalog of classes that are both optimized for students and professors. The primary goal of this project is to develop a user-friendly scheduling system tailored for academic departments, with an emphasis on the needs of the Engineering Department. It aims to overcome challenges in semester class scheduling due to the diverse needs and course requirements of a large student body.

Section 1.2 Significance

The importance of the simple intuitive website application is that it will help administrators create schedules for students, saving them time having to do redundant tasks. Administrators can focus on other things instead. Lastly, the simple website aims to overcome challenges in semester class scheduling due to the diverse needs and course requirements of a large student body.

This project will also be a great learning experience, in not only web development but algorithms, frameworks, and team development (working in a team).

Section 2. Background

Section 2.1 Prior Arts in Research/Commercial Development

There have been other attempts to solve the issues with scheduling courses for students and instructors optimally. In 2022[1] a group of researchers tackled the issues revolving around the constraints within

[1] ^1 Yu Chen, Mahmonir Bayanati, Maryam Ebrahimi, Sadaf Khalijian, "A Novel Optimization Approach for Educational Class Scheduling with considering the Students and Teachers' Preferences," *Discrete

scheduling courses optimally for students and instructors by proposing a metaheuristic algorithm, genetic. The goal of their approach is to maximize the amount of units students can take. The structure of the genetic algorithm is similar to the concepts of evolution, where it will take many parameters and output all the possibilities from it and from there the strongest and most compatible combination will be prioritized. According to the results, the method was able to satisfy all the constraints, including student and instructor preferences.

In 2021[2], a group of researchers from the Department of Computer Science Engineering at Maharaja Surajmal Institute of Technology (GGSIPU) in India, combat the issues with timetable scheduling in academic institutions. Their approach is to use the graph coloring technique due to its ability to solve complex relationships between objects. The way color graphing functions is that the graph is a collection of vertices and edges that connect them. In the context of assigning color, color graphing works by assigning colors to vertices of a graph in such a way that no two connected vertices share the same color.

Our approach is more suitable and practical as we will be implementing a front-end function as well. Our approach will not involve miscellaneous parameters such as class size, course capacity, or number of available rooms. Rather, we will focus on more major constraints that are students' engineering pathway requirements to successfully graduate.

Section 2.2: Prior Experience

I am a senior in electrical engineering and I have taken ENGR213 but I have experience with machine learning as I took Andrew Ng's Machine Learning courses. I am also taking Hardware for Machine Learning. I have experience in Python as well and different libraries such as NumPy, Pandas, Keras, and Tensorflow.

Section 2.3 Budget

How much is the cost? And how do you pay for the project?

The cost will primarily cover software and services costs, with funding sourced from the students and the engineering department. Cost will be minimal as we are making the whole website online. Using any hosting service like AWS or google cloud is minimal for the amount of data we will be putting through our program. AWS is able to process 50 gigabytes of data at a cost of around 2 dollars a month. Prices

---

Dynamics in Nature and Society*, vol. 2022, Article ID 5505631, 11 pages, 2022, https://doi.org/10.1155/2022/5505631.

[2] ^1 P. Nandal, A. Satyawali, D. Sachdeva, and A. S. Tomar, "Graph Coloring based Scheduling Algorithm to automatically generate College Course Timetable," 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2021, pp. 210-214, doi: 10.1109/Confluence51648.2021.9377151.

seem to correlate depending on the amount of usage. MongoDB is very minimal cost as well, advertising a price of 10 cents per 1 million reads.

Section 3. Approach/Design

Overview of the entire diagram or flow chart

User Interface - A simple intuitive user interface that is easy to use, making it easy for administrators to interact with the website. Used Figma to create a prototype of how our application would look like.

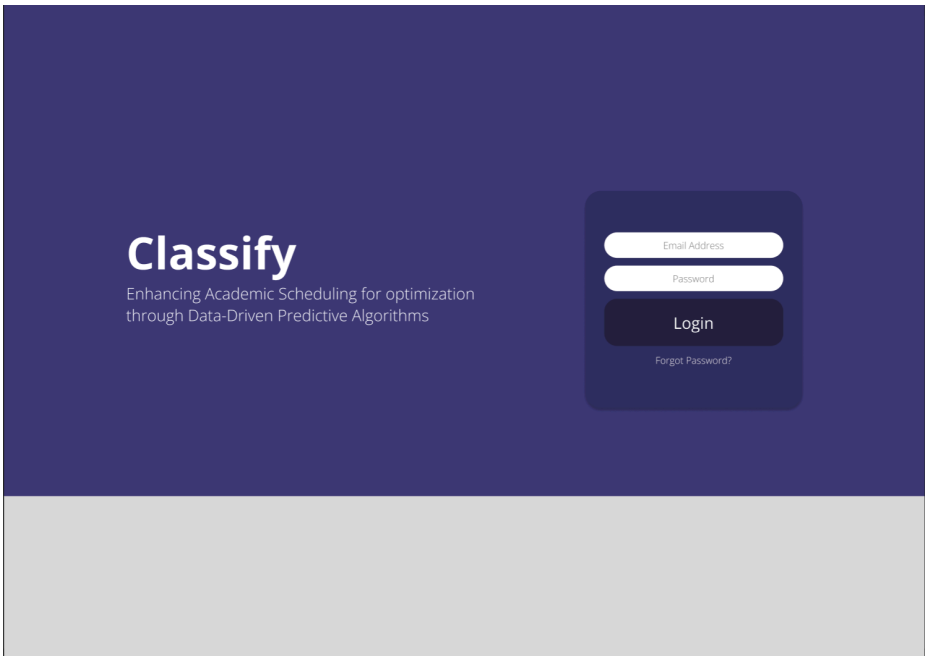


Figure 1: Used Figma to create simple login page for application

Responsive - Program will be optimized to run as responsive as possible. Leverage React framework to create a quick responsive website.

Algorithms - Use algorithms to optimize scheduling for classes. Creating schedules that are optimized for professors and students. We will be using decision tree learning to be able to create an output which is optimized based on our inputs.

Front-end- This would be the aspect that a user will interact with, the front-end would communicate seamlessly with the back-end.

Back-end- This is where all the algorithm aspects of the project would go, as well as, the code that makes the data inputted go to our database/server. The data will come from the front-end which then gets put

into our Postgresql database, once that data is received the algorithm can take the input and begin to work.

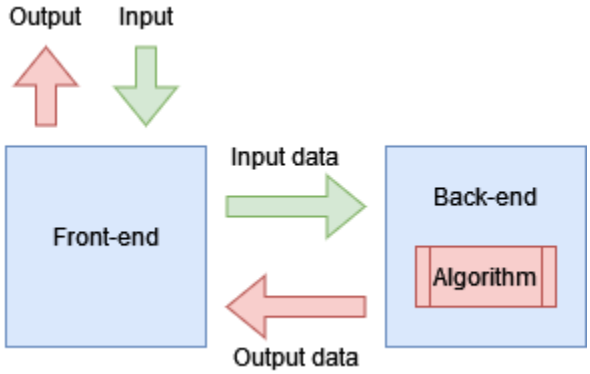


Figure 2: The flowchart of the process of the program

Section 3.1 Design Objective/Specification 1

Section 3.1.1 Introduction

To create the front-end of our program we must implement a user interface that allows the user to upload their input. Using React to capture this data to then send it to the back-end via a REST API. This API should be capable of receiving spreadsheet data. Once the back-end has received the data, the back-end utilizing python will be able to parse through the spreadsheet. After this is accomplished the algorithm can then process and analyze the data.

Postgresql will then store the processed/optimized data. Our database should be designed to efficiently store and retrieve this data. Once this process is complete, the back-end should be able to send this information back to the front-end. Throughout these processes our back-end should be able to send responses to the front-end to show that the processes are in progress or completed.

Coming back to the front-end, the UI should be able to display the results. This information should be shown within the application.

Section 3.1.2 Approach

Our approach will begin with creating a user-friendly interface. Ensuring we have a UI/UX that allows anyone to use our application. To do this we will use React to build a dynamic interface. Implementing a way for users to upload their files is important, a section in the front-end will be created for users to simply upload their spreadsheets. To do this we must ensure we have the correct processes in the program. We must code a section in the back-end that is able to parse data which will then lead to the data to be processed by the algorithm.

All of the program will be coded in collaboration, utilizing GitHub to ensure we are able to collaborate. Taking this approach will ensure a program that is easily debugged and updated.

Section 3.1.3 Anticipated Pitfall and Mitigation Plan

Anticipated pitfalls would be that our program is incapable of sending data from one “stage” to another. This can be easily mitigated by prioritizing on getting this process to work.
Another pitfall could be that our algorithm process is very slow to get results. Going from one stack to the next could potentially slow down our program and result in users having to wait an extended amount of time to get their results.

Processing large files through our program can lead to slow performance and crashes. To fix this issue we put a max on the file size that can be uploaded, or implement some form of file compression. Parsing different spreadsheet formats can also lead to inconsistent parsing. To stop this we can implement libraries like Pandas in python which handles various spreadsheet formats efficiently.

Since our program will be storing our processed data on Postgresql we must implement efficient indexing to speed up queries, this will mitigate the issue of users having to wait an extended amount of time waiting for their results

Section 3.1.4 Anticipated Results

The result of this design objective will be a user-friendly, efficient, and scalable application that will be capable of optimizing the scheduling of an entire class catalog in seconds.

Our program should be able to handle and process any size within the limits of the program spreadsheet that the user will input, ensuring the output is received by the user as quickly as possible. Optimizing each step that the data will take in our program. Continuous collaboration will ensure a working website application.

Section 3.2 Design Objective/Specification 2

Section 3.2.1 Introduction

To create our algorithm we will focus on integrating simple if-else logic alongside supervised learning algorithms to improve the outcomes of our inputs. The objective is to create a responsive system that is capable of making decisions based on user inputs and learned patterns. Decision trees are an example of supervised learning that we can utilize to optimize our inputs. Decision trees are capable of handling both numerical and categorical data, this makes it a perfect algorithm to process data from a spreadsheet. The if-else logic will be able to handle any straightforward decision making processes, while the Decision trees will be used for more complex scenarios where pattern recognition and prediction will be necessary.

Section 3.2.2 Approach

Our approach will involve the implementation of if-else logic to manage and process the basic user input scenarios. This logic will be programmed to handle predictable tasks ensuring quick responses for standard operations.

The more complex decision making will be done by integrating a supervised learning algorithm like Decision trees. This model will be trained on a dataset relevant to our application goal, ensuring our algorithm is capable of making informed decisions based on past data.

To ensure the seamless transition from if-else logic to machine learning based prediction, clear criteria that determine when a task is simple or complex must be implemented. Creating a threshold that determines if the task is too complex for the if-else logic and would then need machine learning processing. These thresholds can be based on factors like complexity of data patterns, variation of input, or lack of information.

Section 3.2.3 Anticipated Pitfall and Mitigation Plan

The main pitfall we will face is handling which aspect of our algorithm should handle the specific task. While the if-else logic should handle easier tasks and the Decision trees handle the more complex task. This process can lead to inefficient or wrong decisions being made.To mitigate this, we must ensure we conduct testing to define the threshold that will ensure the tasks are suitable for either if-else logic or decision trees.

The other obvious pitfall is the accuracy that our model will be able to produce. To ensure our model is able to produce accurate decisions and correct outputs we must train it on high-quality diverse data. As well as, making sure the decision trees are able to adapt easily to inputs that vary.

Section 3.2.4 Anticipated Results

Utilizing a simple if-else logic will ensure that the easy tasks are done correctly and efficiently. While the if-else logic will handle the routine task, the supervised learning (decision trees) will handle the more complex decision that will require intelligent solutions. This dual approach should be able to enhance the application performance. Ensuring it is able to process both routine tasks and complex tasks. We expect our model to only improve over time, as we utilize the Decision trees more it should get better over time at making decisions.

Section 3.3 Design Objective/Specification 3

Section 3.3.1 Introduction

Creating a user interface that is easily usable is key. When creating any sort of application whether it's a website or not, the user experience is a priority. UI/UX of a website will lead a user base that will use your application more. The goal is to create a website that is easily usable by anyone. Making sure users are able to take advantage of any feature that the website provides.

Section 3.3.2 Approach

To be able to create a website with a great UI/UX we must first begin by creating a prototype of our website. Figma is a powerful prototyping tool where we can accomplish this. Creating a wireframe of

how our program would flow through each page will ensure a working product in the end that is both usable and efficient. Figure 1 previously showed a login page that was prototype in Figma. Each page that would be found on our website will be prototyped as well. Doing this will make sure users of our program will enjoy not only using our services but enjoy using the website itself.

Section 3.3.3 Anticipated Pitfall and Mitigation Plan

The biggest pitfall in UI/UX is creating a program that is too complex for users to be able to utilize the application to its full potential. To ensure this does not happen wireframes of the entire flow of our program will be created. It will show what will happen if one button is clicked or what page a button will take the user too. We can also conduct user testing to ensure different kinds of users are all able to easily use the application.

Section 3.3.4 Anticipated Results

Creating an application that is not only powerful in the back-end but elegant in the front-end is the goal. Our website application should be accessible and usable by anyone. Making UI/UX a top priority will ensure a website that is optimized to be used by anyone.

Section 4. Results and Discussions

Data Collecting

To begin this project we created two datasets which consisted of fall and spring 2023. This historical data was used to train the Decision Tree algorithm. We were very careful to standardize the way both these datasets were created. The main parameters that were focused on were the course, unique course number, time, and day. Extra details that were taken into consideration were course instructors, number of units, and the binary boolean of whether the course is required for a specific engineering major.

UI/Ux

Creating a user interface/user experience that was easily accessible to everyone was a major goal of our project. To ensure we created a website that is easily accessible, each page was prototyped in Figma. Utilizing Figma to create the layout/wireframe of our website allowed us to create a user-friendly website that is both straightforward and responsive.

Users can either create an account or log into their previously made account. The account information is stored in the PostgreSQL database. The account password is hashed for privacy and security. Once users successfully log in or register they can now access the website. On the home page is a table showcasing the team with photos. Although the goal was to create a more intuitive front-end that acts as a tool rather than a showcase, due to time constraints this was the optimal choice that allowed the end-user to easily understand what the purpose of the tool isWhile the user does not have direct ability to update their input, the input can be updated through accessing it in the back-end which in turn updates the predicted sub-optimized future schedule.

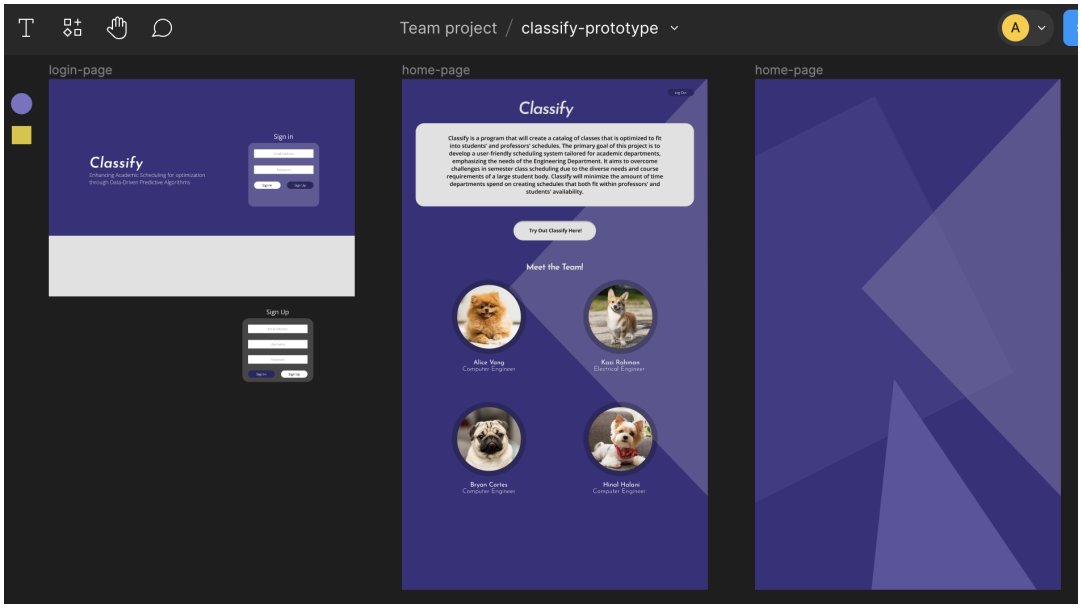


Figure 3: Figma prototypes

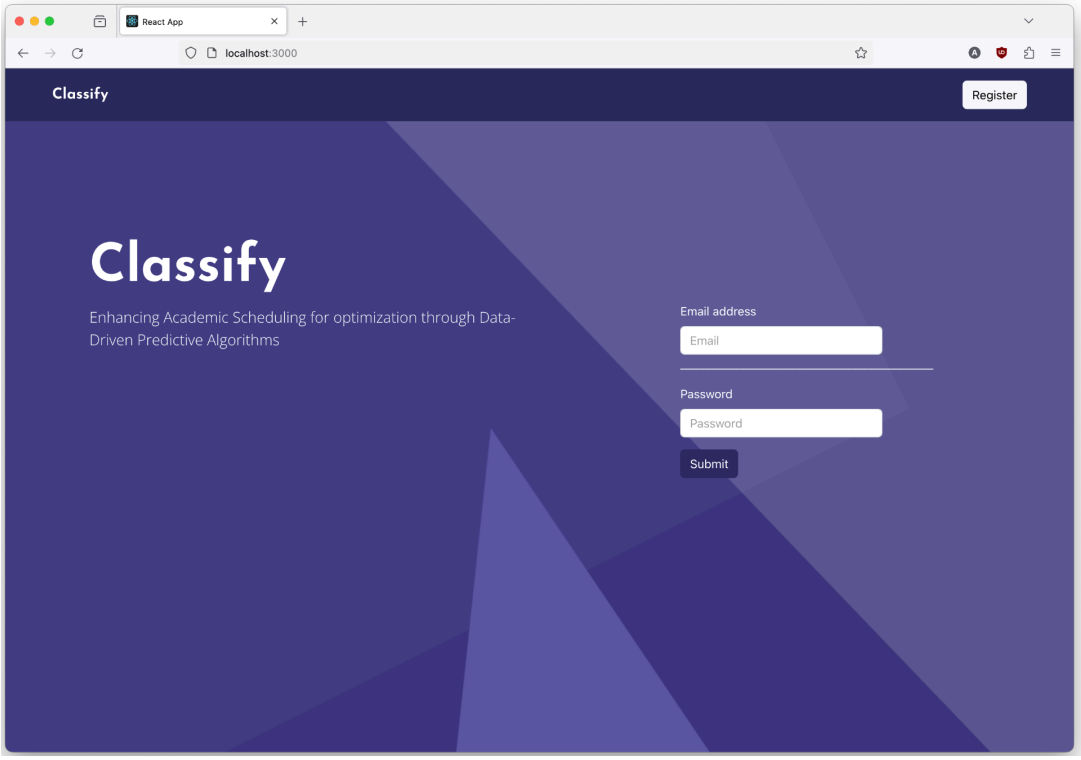


Figure 4: Login page

Front-End

The front-end was developed using React to enable us to use their intuitive functions. Our front-end is a single-page application that allows our end-user to use it easily. Since the program is specifically intended for administration use, we created a simple login page as shown in Figure 4. The

end-user will be able to toggle between registering and logging in depending on whether they are making a new user or using a current user. After logging in, the first thing that the end-user is introduced to is the primary concept summary about the project.

The end-user then goes through a series of buttons that act like an execute switch to run parts of the machine learning script to clean, analyze, predict, and evaluate. The first thing the end-user must do is select a CSV file that contains the data from previous semesters. As shown in Figure 5, we have to input a file. If we do not give the program a file, an error will flag and direct the user to input a valid CSV file before moving on. For demonstration purposes, we went ahead and chose the Fall 2023 semester as an input file. The program will then run the data analysis and output visual charts to better understand what time and day are prioritized for these courses offered in the semester.

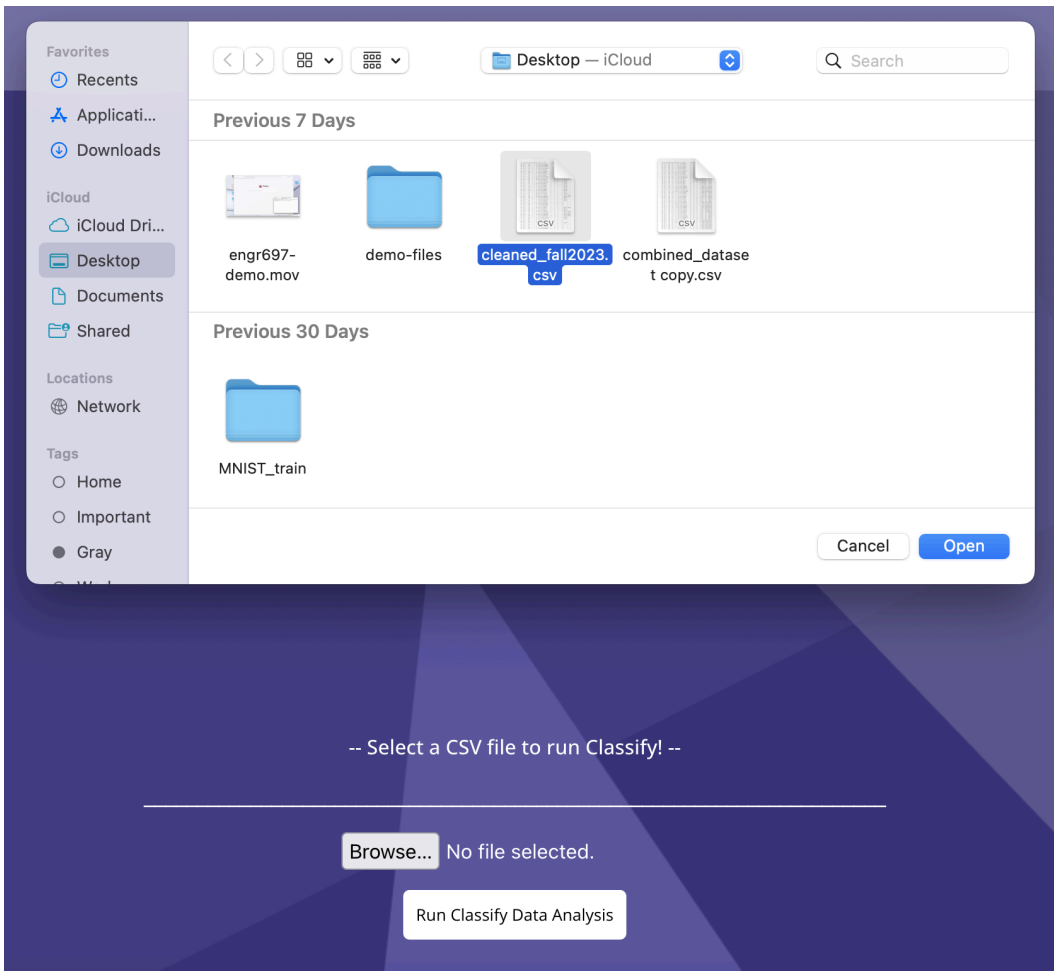


Figure 5: User Input

After running the analysis, the end-user will continue to execute each part of the script until it reaches the end and outputs the evaluation results that summarize the efficiency of the script. Since React is a user-friendly technology, it allows the creation of necessary functions to be straightforward while keeping the flexibility from JavaScript and the extra tools from React's Bootstrap.

For the prediction execution, the URL will be fetched from the back-end. The CSV file will then be sent through another function to parse it properly. The data is parsed by splitting the lines, extracting the

headers, and mapping out the data based on placement. After it is parsed, JavaScript is used to properly display it. For every URL that is fetched, whenever there are bugs, errors are thrown in the proper areas to further comment on the why and where the problem is. The console will output logs of information to make sure that everything is streamlined.

Back-End

Django and Postgres were used for the back-end. For the Django to communicate with React we had to install Axios from React so that the HTTP requests were able to translate into the REST framework that way Django was able to understand and output back an HTTP response to which Axios would also translate back so React could properly perform an output. Django and React are two different Technologies, so for them to be able to communicate, we installed CORS (cross-origin resource sharing) headers to allow cross-origins. It will allow different domains to access Django since the localhost port for React is 3000 and Django's port is typically 8000.

When an HTTP request is sent from React, it first goes into URLS, and then depending on what was sent, views will execute and return the specific function. For the machine learning portion, various functions were connected to the machine learning script. It ran specific portions to do the data analysis, predictions, and evaluations. As for the login and register function, when an HTTP request is sent from React it goes through the serializer to authenticate the user based on the email and password given. From there the user will be saved as a model in models and it will be saved into the Postgres database. Whenever a user tries to log in it will create a new session ID in React and save the cookies of the user's activity. After logging out the session ID will expire and no longer be active.

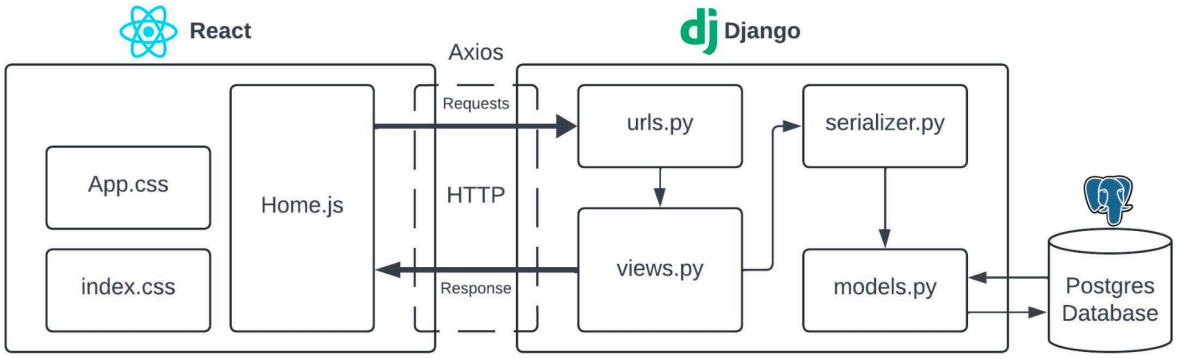


Figure 6 - Flowchart of the frontend and backend

As shown in Figure 6, when React sends an HTTP request and fetches the information from views, it will be converted into JSON Because it is integrated seamlessly with JavaScript, which React uses. Hence why, from the Django REST framework, our front-end also has the same formatting. The images of the data and graphs were saved as a PNG. These images were saved in the path of our front-end to allow our front-end to have access to them and output them properly. Every time the script is run, it updates and replaces the images and data based on the data inputted and the script manipulated.

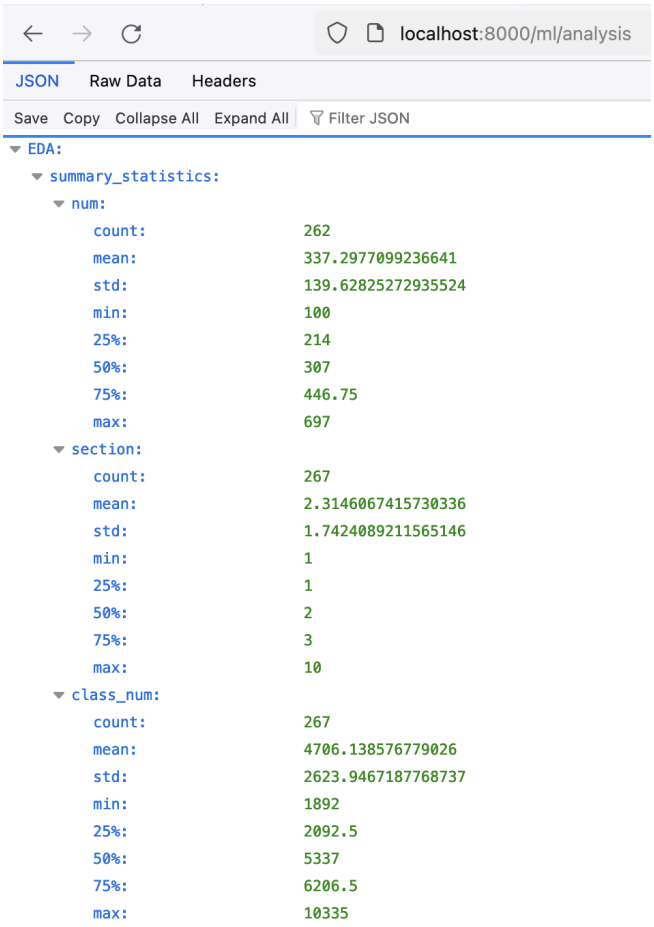
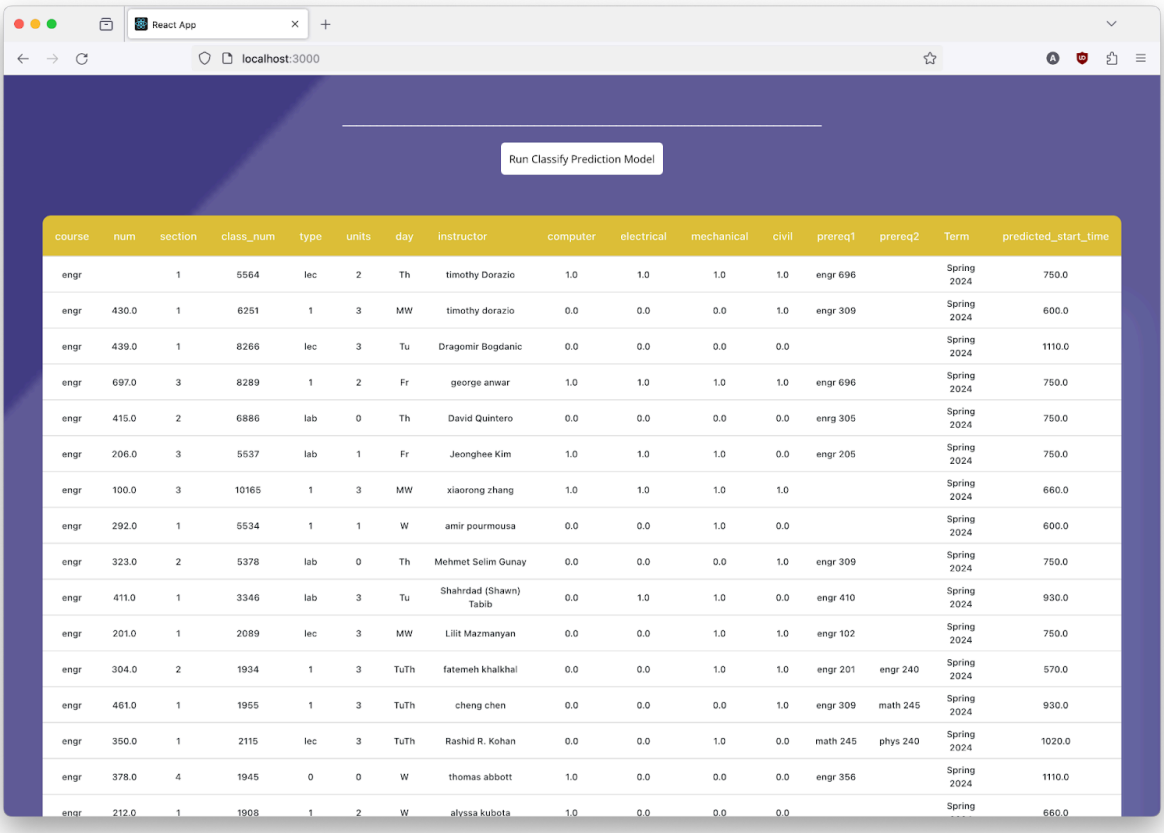


Figure 7 - Django REST Framework of analysis



course	num	section	class_num	type	units	day	instructor	computer	electrical	mechanical	civil	prereq1	prereq2	Term	predicted_start_time
engr		1	5564	lec	2	Th	timothy Dorazio	1.0	1.0	1.0	1.0	engr 696		Spring 2024	750.0
engr	430.0	1	6251	1	3	MW	timothy dorazio	0.0	0.0	0.0	1.0	engr 309		Spring 2024	600.0
engr	439.0	1	8266	lec	3	Tu	Dragomir Bogdanic	0.0	0.0	0.0	0.0			Spring 2024	1110.0
engr	697.0	3	8289	1	2	Fr	george anwar	1.0	1.0	1.0	1.0	engr 696		Spring 2024	750.0
engr	415.0	2	6886	lab	0	Th	David Quintero	0.0	0.0	0.0	0.0	enrg 305		Spring 2024	750.0
engr	206.0	3	5537	lab	1	Fr	Jeonghee Kim	1.0	1.0	1.0	0.0	engr 205		Spring 2024	750.0
engr	100.0	3	10165	1	3	MW	xiaorong zhang	1.0	1.0	1.0	1.0			Spring 2024	660.0
engr	292.0	1	5534	1	1	W	amir pourmousa	0.0	0.0	1.0	0.0			Spring 2024	600.0
engr	323.0	2	5378	lab	0	Th	Mehmet Selim Gunay	0.0	0.0	0.0	1.0	engr 309		Spring 2024	750.0
engr	411.0	1	3346	lab	3	Tu	Shahrdad (Shawn) Tabib	0.0	1.0	1.0	0.0	engr 410		Spring 2024	930.0
engr	201.0	1	2089	lec	3	MW	Lilit Mazmanyan	0.0	0.0	1.0	1.0	engr 102		Spring 2024	750.0
engr	304.0	2	1934	1	3	TuTh	fatemeh khalkhal	0.0	0.0	1.0	1.0	engr 201	engr 240	Spring 2024	570.0
engr	461.0	1	1955	1	3	TuTh	cheng chen	0.0	0.0	0.0	1.0	engr 309	math 245	Spring 2024	930.0
engr	350.0	1	2115	lec	3	TuTh	Rashid R. Kohan	0.0	0.0	1.0	0.0	math 245	phys 240	Spring 2024	1020.0
engr	378.0	4	1945	0	0	W	thomas abbott	1.0	0.0	0.0	0.0	engr 356		Spring 2024	1110.0
engr	212.0	1	1908	1	2	W	alyssa kubota	1.0	0.0	0.0	0.0			Spring	660.0

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	course	num	section	class_num	type	units	day	begin	end	instructor	computer	electrical	mechanica	civil
2	engr	100	3	10165	1	3	MW	1100	1150	xiaorong zhang	1	1	1	1
3	engr	100	4	10166	2	0	M	1600	1740	xiaorong zhang	1	1	1	1
4	engr	100	7	10169	1	3	TuTh	1000	1050	lilit mazmanyan	1	1	1	1
5	engr	100	8	10170	2	0	Th	1200	1340	lilit mazmanyan	1	1	1	1
6	engr	100	9	10171	1	3	TuTh	1100	1150	jenna wong	1	1	1	1
7	engr	100	10	10172	2	0	Th	1600	1740	jenna wong	1	1	1	1
8	engr	101	1	1892	0	1	MW	1830	2115	kawai law	0	0	1	1
9	engr	101	2	1893	0	1	Tu	1830	2115	jun jian liang	0	0	1	1
10	engr	101	3	1894	0	1	W	1830	2115	kawai law	0	0	1	1
11	engr	101	4	8617	0	1	MW	1830	2115	yuanhong chen	0	0	1	1
12	engr	102	1	1895	1	3	MW	1230	1345	lilit mazmanyan	0	0	1	1
13	engr	102	2	1896	1	3	MW	1400	1515	natalia igu	0	0	1	1
14	engr	103	1	9960	0	1	M	1530	1815	jonathan song	0	0	1	0
15	engr	103	2	9958	0	1	Tu	1530	1815	jonathan song	0	0	1	0
16	engr	200	1	1900	1	3	MW	1000	1050	natalia igu	0	0	1	1
17	engr	200	2	1901	0	0	Tu	930	1215	natalia igu	0	0	1	1
18	engr	200	3	10131	1	3	Th	1000	1050	natalia igu	0	0	1	1
19	engr	200	4	10133	0	0	Tu	1230	1515	natalia igu	0	0	1	1
20	engr	200	5	10134	1	3	MW	1000	1050	natalia igu	0	0	1	1
21	engr	200	6	10135	0	0	Th	930	1215	natalia igu	0	0	1	1
22	engr	201	1	1902	1	3	MW	1230	1345	natalia igu	0	0	1	1
23	engr	203	1	1903	1	3	MW	1100	1215	m hajiaboli	0	0	1	0
24	engr	205	1	1904	1	3	MWF	900	950	jeonghee kim	1	1	1	1
25	engr	205	2	1905	1	3	MWF	1000	1050	jeonghee kim	1	1	1	1
26	engr	206	1	1906	0	1	Tu	1230	1515	jonathan song	1	1	1	0
27	engr	206	2	1907	0	1	Th	1530	1815	jeonghee kim	1	1	1	0
28	engr	206	3	5469	0	1	F	1230	1515	jeonghee kim	1	1	1	0
29	engr	212	1	1908	1	2	W	1100	1150	alyssa kubota	1	0	0	0

Machine Learning/ Schedule Optimization

To create the future predicted schedule we employed a Decision Tree regressor. Our objective was to predict the optimal start times of classes, this would then lead to a reduction of class scheduling conflicts.

Merging Datasets

To begin the machine learning process we combined the Fall and Spring datasets. These datasets included the set of features that provided a foundation for the machine learning process. Features such as course number, section, class number, type, units, day, start and end times, instructor, and prerequisites. An important aspect of our program was to ensure that any irregularities within the datasets were cleaned. This would help prevent any disruption in the machine-learning process. Merging the datasets allowed the dataset to go from one step of the program to the next much easier. Streamlining this process allowed for a unified analysis and modeling of the datasets. To ensure the differentiation of the different terms this merged dataset had an extra Term column to distinguish between each term.

Data Cleaning

Data cleaning was essential in ensuring both data quality and consistency. Standardizing time formats, converting categorical data to appropriate types, and handling any inconsistent or missing values. Standardizing time formats was very particularly important for accurate time-based analysis. Converting the begin time column into a numerical format (minutes after midnight) was essential for the computations in the predictive model. This conversion facilitated the comparison of class times and the calculation of the conflicts within the schedule. This specific aspect of the program was imperative since any inconsistencies with the time attribute could lead to wrong numbers in many aspects of the program.

Exploratory Data Analysis (EDA)

EDA was performed to understand the distribution and characteristics of the data. This analysis provided insights into the data, such as the distribution of classes across different days, start times, and instructor workloads. These insights were crucial for identifying patterns and potential issues in the scheduling data.

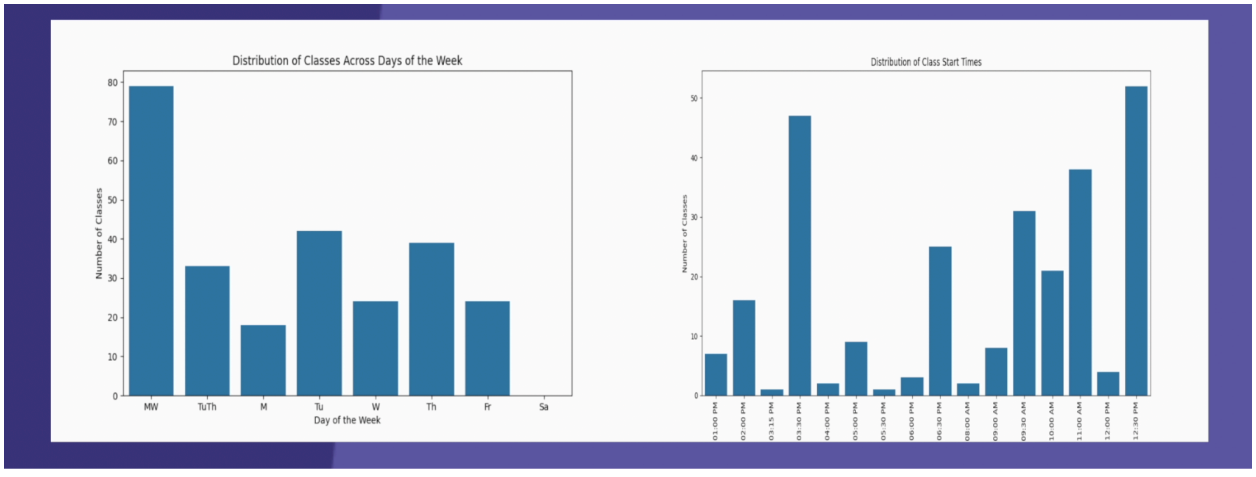


Figure 8: Distribution of classes

Conflict Analysis

Identifying scheduling conflicts was important in the process of creating an algorithm that would reduce the amount of conflicts. Functions were developed to calculate conflicts based on overlapping class times. These functions generated conflict scores for the Spring, Fall, and combined datasets, providing a quantitative measure of scheduling conflicts.

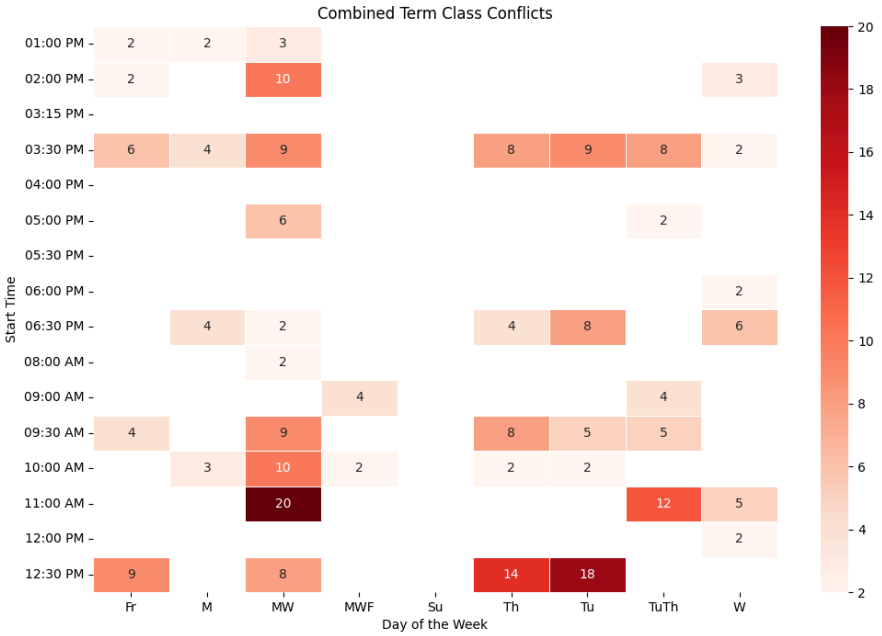


Figure 9: Heatmap of conflicts within combined dataset

Feature Engineering

Processing and validating the prerequisite data features to ensure they would be processed accurately within the model was a crucial step to incorporate the course dependencies into the scheduling model. This enabled the algorithm to take into account the prerequisites between each course.

Predictive Modeling

To implement the Decision Tree Regressor a comprehensive pipeline was created. This pipeline consists of data processing steps like imputation for missing values and one-hot encoding for categorical variables. The first step in the actual machine learning process was splitting the data sets into training and testing sets. Then after training the model the algorithm was able to make predictions. This pipeline process ensured that the model was capable of handling data inconsistencies and produce accurate predictions.

Output Generation

Finally, the predicted sub-optimized schedule was able to be generated. Alongside the predicted schedule an updated heatmap of the scheduling conflicts was generated. This heat map provided a visual representation of the sub-optimized schedule. This in turn allowed us to verify that the amount of conflicts was indeed reduced quantifying the improvement that the algorithm was able to achieve.

Our detailed approach not only streamlined the scheduling process but also provided valuable insights into current scheduling patterns, potential conflicts, and areas for improvement. By incorporating a Decision Tree Regressor, thorough data preprocessing, and conflict analysis, we developed a robust scheduling model ready for further enhancements and real-world application.

Section 5. Self Assessment

Lessons and Improvement

Throughout the project we identified several areas for improvement and learned valuable lessons. One of our early challenges was ensuring that the datasets had as little errors or inconsistencies. Some of these mistakes seeped in and multiple stages of the program were being affected. Many times we had to locate a tiny mistake in a very long spreadsheet of information. These mistakes directly affected the performance of our entire program, especially the effectiveness of our model.

In the process of creating our project one of the first blaring issues we had was the lack of data. This lack of data pertains to the fact that we do not know key factors in creating future schedules. These key factors being professors availability, student and professors preferences, classroom availability, and so much more data that would be crucial in improving our predictive model.

While our outcome produced a sub-optimized schedule with less conflicts than previous semesters, the lack of data bottlenecked our Decision Tree regressor. To combat this one algorithm that we looked into but decided to abandon since the outputs were worse was a Random Forest. This algorithm was briefly looked into since it is so similar to a decision tree. In the end however, the Decision Tree Regressor was capable of accomplishing a sub-optimized schedule.

Section 6. Conclusions

In the end we were able to successfully develop a scheduling optimization system that is all accessible through a user friendly website. This website allows users to sign in or log in providing both a secure and personal interaction with the platform. Users are able to easily upload their input schedule files and interact with various features of their inputs. Enabling them to manage and analyze their data. The website showcases the capabilities of our machine learning algorithm, providing a clear and clean view of the sub-optimized future schedule.

Utilizing decision trees proved to be an effective choice in optimizing start times of classes and reducing scheduling conflicts. Through detailed preprocessing, data cleaning, exploratory data analysis,

and feature engineering, our machine learning model is capable of producing a schedule with reduced conflicts.

A major challenge which limited the performance of our model was lack of extensive data on professor availability, student preference, classroom availability, etc. These limitations highlight the importance of collecting as much data as possible to improve the scheduling process within the machine learning model. Incorporating these critical datasets would significantly improve the models ability to produce a schedule that is optimized for both professors and students.

Overall, this project demonstrates the potential of combining machine learning with a user-centric web platform to tackle complex scheduling problems.