CPC Final Report

Client: Northwestern Registrar
Key Client Contact: Dustin Levell
Team Members: Alex Romanenko, Antonio Santana, Mekhi Barnes Adams, Mark Ruiz
Date: June 4, 2024

Table of Contents

Executive Summary	3
Introduction	4
Solution	6
Limitations and Future Developments	12
Conclusion	13
Appendix 1: Background Research	14
Appendix 2: Data and Software Resources	16
Appendix 3: Preprocessing	17
Appendix 4: Modeling and Analysis	21
Appendix 5: File Overview	37
Appendix 6: Cloud-Based Web Server Instructions	38
Works Cited	40

Executive Summary

Northwestern University's Registrar's Office works closely with department heads and faculty to schedule courses. When scheduling courses, the Registrar's Office first collects class information from department heads using the Course Leaf Section Scheduler (CLSS). This data is then processed by an optimization algorithm, which assigns classrooms to classes. However, the accuracy of classroom allocation is severely diminished by unreliable class enrollment forecasts, which are often based on subjective "best-guesses" rather than data-driven predictions. Inaccurate forecasts can lead to last-minute classroom changes, course cancellations, and students being unable to enroll in desired courses. Therefore, improving the accuracy of class enrollment forecasts is essential for streamlining classroom allocation processes and enhancing the overall academic experience.

To address this issue, we focused our project on predicting class enrollments for courses in the McCormick School of Engineering and Applied Sciences. We experimented with several predictive models including linear regression without regularization, linear regression with regularization, and XGBoost models. After comparing the accuracies of the models, we found XGBoost to be the best, as it could capture nonlinear relationships between predictors. Our final solution employs three distinct XGBoost models tailored to small, medium, and large class sizes. This approach enhances the flexibility of our predictors, allowing their relevance to be influenced by anticipated class size categories. We also used latent semantic analysis (LSA) as a natural language-processing technique to extract important concepts from course titles and incorporate those in our XGBoost predictive models. This is particularly helpful for newly offered courses that lack historical data relevant to high-quality predictions. To evaluate the effectiveness of our predictive models, we measured the average difference between our forecasts and actual enrollments, finding it to be 3 for small classes, 8 for medium classes, and 26 for large classes. Interestingly, our models are particularly accurate for small classes, which the Registrar's Office identified as the most challenging to predict. This is particularly noteworthy because unlike core courses, which tend to have more predictable enrollment numbers, small classes often encompass newer or higher-level courses, which exhibit greater variability in enrollment patterns. Thus, our solution provides significant value for the Registrar's Office. It is our hope that this solution will empower department heads to make data-driven enrollment predictions, reducing reliance on guesswork and improving the efficiency of classroom allocations.

Our primary deliverable to our client is an interactive dashboard website that department heads can use to generate class enrollment forecasts. The interactive dashboard includes inputs that can be toggled by the users according to their desired class specifications. Submitted inputs are passed to our predictive models, which output the enrollment forecast for the specified course. Our dashboard also displays expected enrollment variations based on the start time of the class, allowing faculty to more accurately plan to offer courses at times with greatest student interest. Furthermore, the dashboard enables department heads to save and download forecasts to Excel for future reference. This deliverable is ideal for our clients as it can be used by non-technical department heads, without having to run any code. To ensure ease of use, we include explanatory pages on how our model works and frequently asked questions. Along with the dashboard, we provide a copy of this report and all programming files, ensuring transparency and accessibility of our solution.

Introduction

Northwestern University's Registrar's Office oversees student records and coordinates course scheduling procedures. When scheduling courses, department heads and faculty collaborate with the Registrar's Office, proposing timings and student enrollments for department-specific courses. This information is entered into the Course Leaf Section Scheduler (CLSS) system, which uses an optimization algorithm to allocate classrooms based on departmental submissions. It's essential to recognize that while department heads manage scheduling in department-assigned rooms, they depend on the Registrar's Office for assigning general-purpose classrooms. Following course schedule releases, departments have the flexibility to relocate classes to available classrooms better suited for course needs or enrollment numbers. This intricate scheduling workflow is illustrated in Figure 1. As depicted in the diagram, classroom allocations depend on enrollment forecasts, highlighting that inaccuracies in forecasts directly contribute to inefficient classroom assignments.

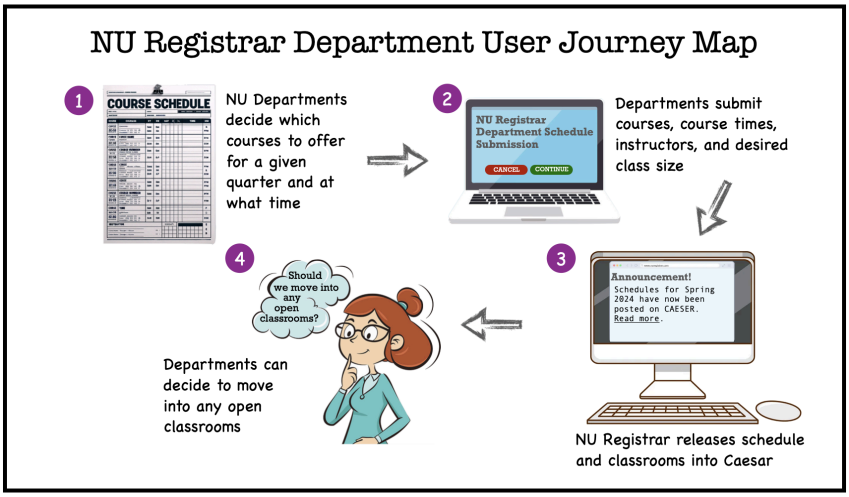

Figure 1: Journey Map For Offering Courses at Northwestern University

The Registrar's Office faces significant challenges in accurately forecasting class enrollment, which adversely impacts classroom allocation across various subjects and sections. Currently, predictions rely exclusively on departmental estimates, which are merely "best guesses" and vary widely in accuracy. These inaccuracies not only strain the Registrar's resources but also negatively impact the academic experience for both students and instructors. Inaccurate predictions can 1) lead to last-minute changes in classroom assignments, 2) cause last-minute cancellations of courses, and 3) prevent students from taking subjects if a class reaches its capacity. Thus, improving the accuracy of class enrollment forecasts is essential for streamlining classroom allocation processes and enhancing the overall academic experience.

With these considerations in mind, we performed background research to understand the types of data that have the most predictive power in forecasting enrollment, as well as applicable modeling techniques. The study by Ma et al. (2021) offers critical insights into the motivations behind university students' course selections, offering personal interest and peer recommendations as important predictive factors. Due to the sensitivity of course evaluation data (CTECs), we were unable to include these factors in our modeling, necessitating alternative approaches. Dianne Lee's research at Harvard University highlights that decision-tree-based models, such as boosted trees and random forests, are particularly effective. This research led us to experiment with XGBoost (Extreme Gradient Boosting) Trees

which combine the strength of multiple decision trees. Simple decision trees are not able to capture some of the more complex trends that could exist in the data but XGBoost uses these initially simple decision trees to sequentially build new trees that work to fix the error and improve the accuracy of the previous trees. Lee also notes that title relevance is a significant predictor for new course offerings, while historical enrollment data is more crucial for continued courses. Gefen et al. (2017) suggest using Latent Semantic Analysis, a statistical technique to represent course titles as a series of concepts. This method constructs a matrix of course titles and term occurrences, converting it into a small number of dimensions or "concepts." Each course receives a score for each concept, which can be used in subsequent modeling to integrate course titles as predictors. We decided to incorporate Latent Semantic Analysis in analyzing course titles, which we found improved our models across the board.

Equipped with a variety of potential modeling techniques, we turned our focus to creating a practical deliverable for the Registrar's Office. The goal was to design a solution that department heads could use directly to forecast class enrollments. After careful consideration of the requirements and user needs, we developed an interactive dashboard tailored to this purpose. This dashboard accepts specific class parameters as input and generates accurate enrollment forecasts based on these inputs. Moreover, it provides insights into expected enrollment variations depending on the start times of the classes. This feature is particularly valuable for department heads, as it allows them to plan course offerings at times that are most likely to attract the desired number of students.

By integrating advanced predictive models into a user-friendly interface, the dashboard ensures that non-technical users can easily navigate and utilize its features without needing to write any code. It empowers department heads to make data-driven decisions, enhancing the efficiency of the course scheduling process.

Solution

Overview

Our recommended solution is a comprehensive dashboard designed to forecast enrollments for McCormick courses. This tool aids McCormick departments in planning and scheduling classes more efficiently by leveraging predictive analytics. The dashboard employs three distinct XGBoost models, using the user's anticipated enrollment input (e.g., 0-30 students) to select the most appropriate model. We chose XGBoost for our modeling technique after an extensive analysis (see Appendix 4) revealed its superior accuracy over other predictive models like linear regressions. The dashboard not only integrates advanced predictive modeling but also offers ease of use for department heads, making enrollment forecasting accessible and efficient.

Results

Our XGBoost models equip department heads with accurate enrollment forecasts. The average difference between our forecasts and actual enrollments is 3 for small classes, 8 for medium classes, and 26 for large classes. Notably, our models excel in predicting small class enrollments, a task the Registrar's Office has identified as particularly challenging. This achievement is significant because, unlike core courses with more predictable enrollment numbers, small classes often include newer or higher-level courses that display greater enrollment variability. Therefore, our solution offers considerable value to the Registrar's Office, especially in forecasting small classes.

To improve usability of our models, we developed an intuitive website for generating enrollment forecasts, eliminating the need for department heads to write any code (see Figure 2). The website features a "How Our Model Works" page (see Figure 3), a "Department FAQs" page (see Figure 4), and a "Dashboard" page. We include examples of our dashboard in Figures 5 and 6, for the courses IEMS 304 and COMP_SCI 214 respectively.

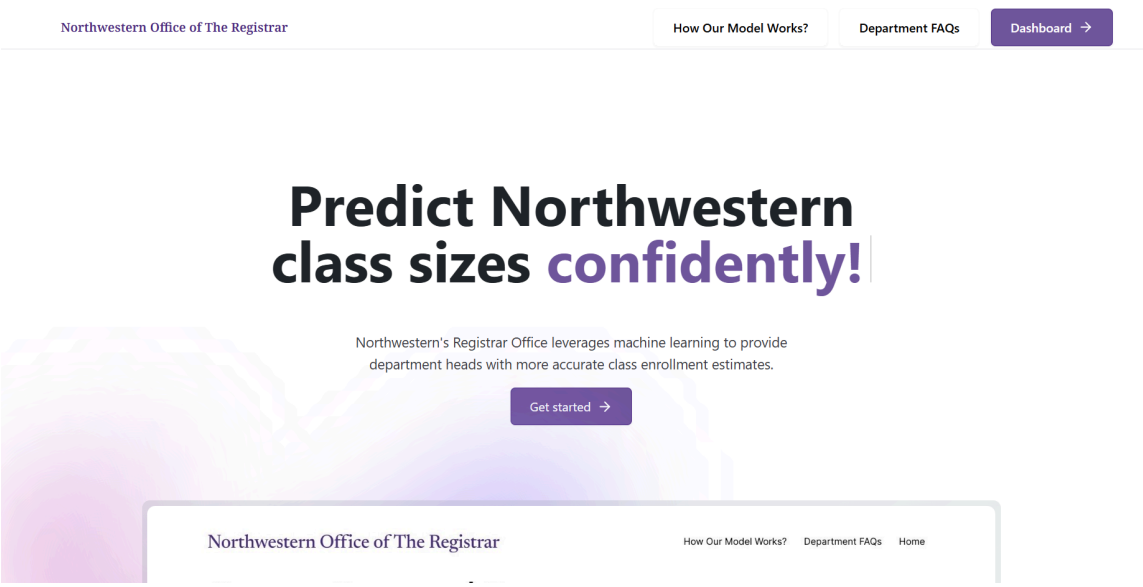

Figure 2: Homepage Featured at https://nuregistrarforecasting.web.app

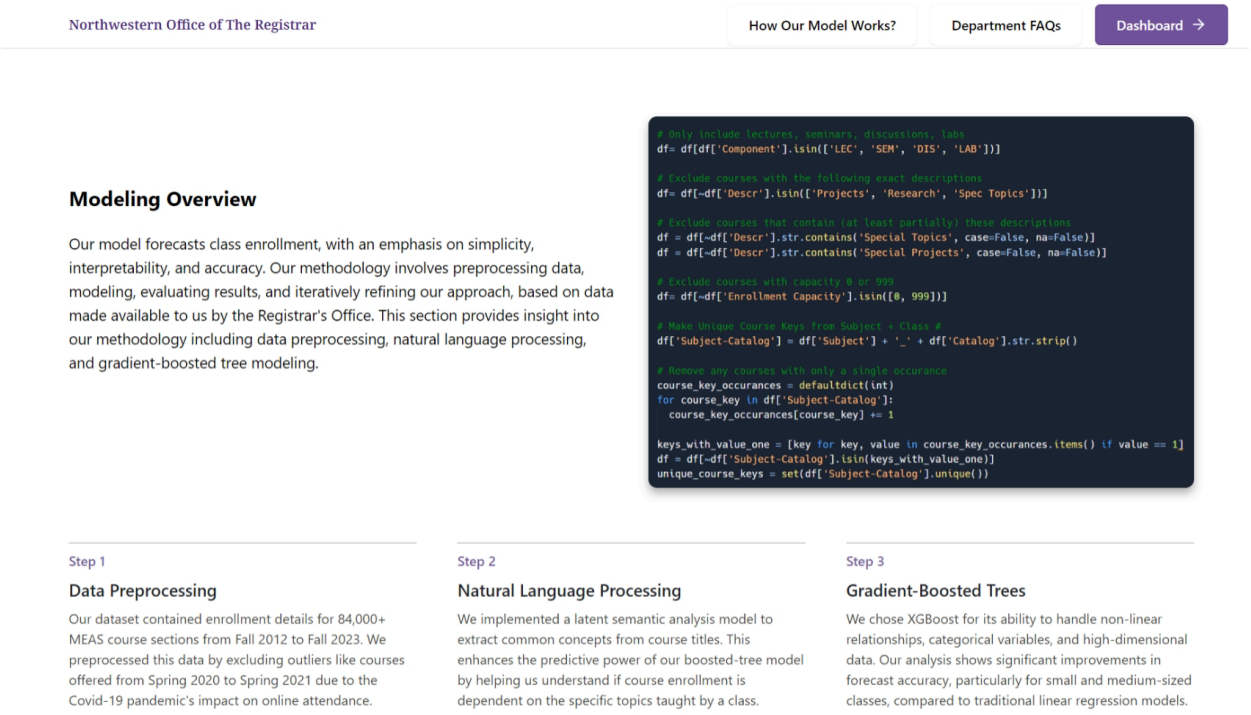

Figure 3: How Our Model Works Page Featured at https://nuregistrarforecasting.web.app/model

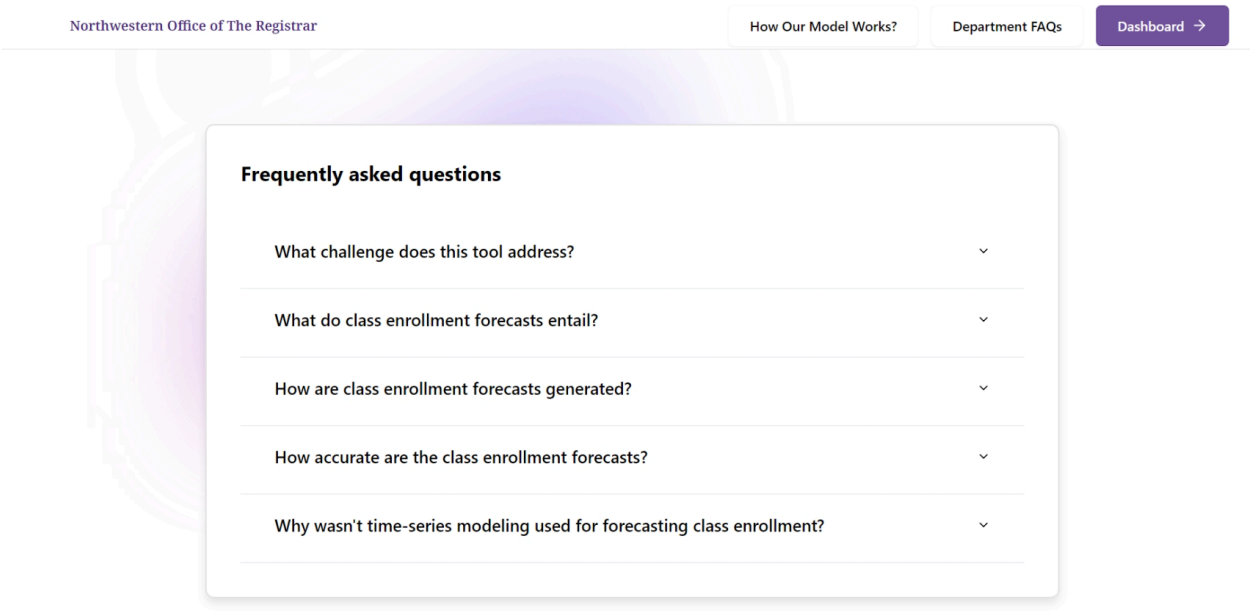

Figure 4: Department FAQs Featured at https://nuregistrarforecasting.web.app/faq

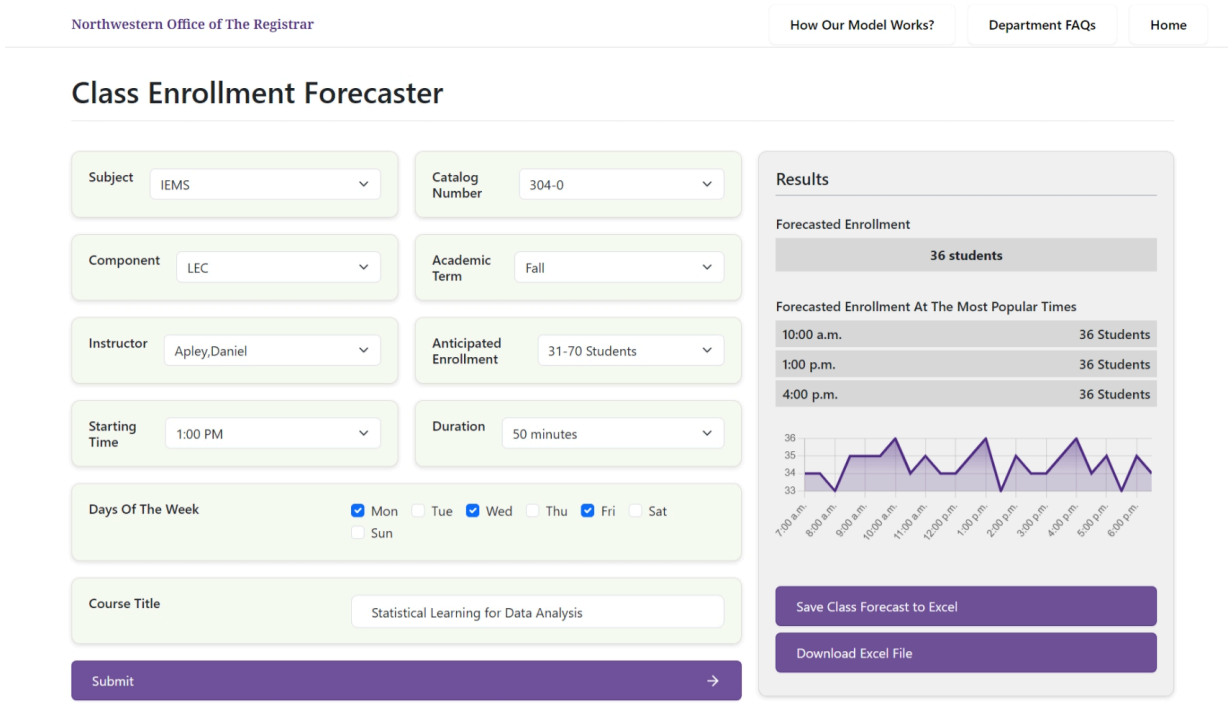

Figure 5: Dashboard For IEMS 304 Featured at https://nuregistrarforecasting.web.app/dashboard

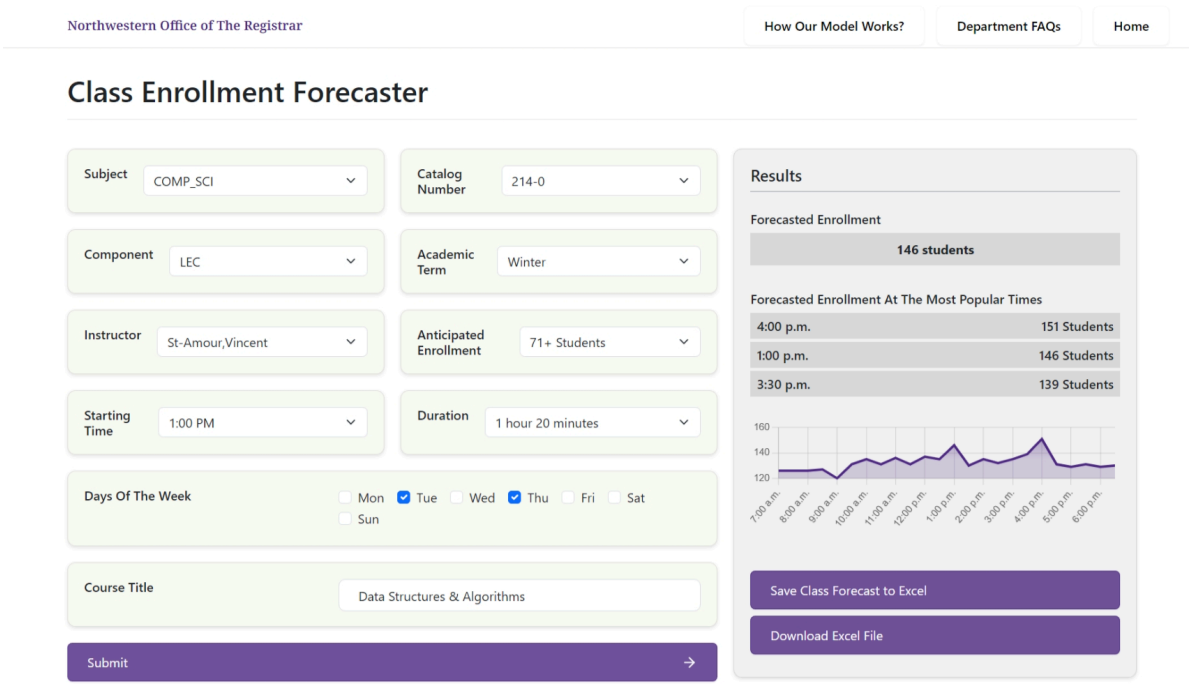

Figure 6: Dashboard For COMP_SCI 214 Featured at https://nuregistrarforecasting.web.app/dashboard

The dashboard features two main components: 1) a set of inputs and 2) a set of outputs. After a user submits inputs, the dashboard processes the data using an XGBoost model to predict the expected class size. Specifically, there are three XGBoost models, each one trained on different class sizes. The anticipated enrollment range provided by the user determines which model is used, ensuring the most accurate predictions. These enrollment forecasts are returned as output for display on the dashboard.

Inputs

The dashboard accepts inputs of three data types: categorical, binary, and textual. Course titles and subjects are particularly valuable in modeling newly offered courses, which pose greater challenges in prediction due to a lack of historical data. They provide context and categorical differentiation that enhance the model's accuracy.

- Categorical:
    - Subject: The specific department or subject area (e.g., IEMS).
    - Catalog Number: The unique course identifier (e.g., 394-0).
    - Component: The type of class component (e.g., LEC for lecture).
    - Academic Term: The term in which the course is offered (e.g., Fall).
    - Instructor: The name of the instructor (e.g., Abrams, Daniel Michael).
    - Anticipated Enrollment: The expected number of students (e.g., 0-40 students).
    - Starting Time: The start time for the class (e.g., 7:00 AM).
    - Duration: The length of the class (e.g., 50 minutes).
- Binary:

    - Days of the Week: The days the class is held (e.g., Mon, Tue, Wed, Thu, Fri, Sat, Sun).
- Textual:
    - Course Title: The title of the course (e.g., Client Project Challenge).

Outputs

The dashboard output includes forecasted enrollment, an enrollment vs. start time plot, and the top three starting times with the largest forecasted enrollment. It also enables department heads to save multiple enrollment forecasts and download them as an Excel file.

- Forecasted Enrollment
    - This value represents the forecasted enrollment for the provided user inputs.
- Enrollment vs. Start Time Line Graph
    - This graph displays how total predicted enrollment varies with different starting times, allowing departments to identify optimal class timings and make data-driven scheduling decisions.
    - This feature was included after an interview by a potential user, Professor Wilson, who identified the need for a tool to aid in planning and scheduling. The ability to predict enrollment patterns and optimal class times provides valuable insights for academic departments, enhancing their scheduling capabilities and ensuring better resource allocation.
- Top 3 Start Times List
    - The dashboard lists the top three recommended start times based on predicted enrollments, for easier visualization of the most popular times.

Data and Software Resources

Our project uses a dataset from Northwestern University's Registrar's Office that includes enrollment information for over 84,000 MEAS course sections. It includes course titles, course instructor, days of the week a course is offered, and several other columns previously discussed in the inputs section. Despite requesting additional data on Course and Teacher Evaluations (CTECs), historical course forecasts, and other factors, we faced limitations accessing this data due to privacy concerns and availability.

To analyze this data and build our machine learning models, we used Python, leveraging libraries such as pandas, NumPy, scikit-learn, XGBoost, SciPy, and Matplotlib. For developing the front-end website and dashboard, we utilized the React framework in JavaScript, which is widely used for creating modern web applications. Additionally, we employed Flask in Python as our local web server to connect our machine learning models to our dashboard. For further details on data and software resources, please refer to Appendix 2.

Updating Our Solution

To keep our solution relevant and effective for future use, it's essential to understand its architecture. Our system functions by training XGBoost models in Python from a provided dataset. These trained models are then stored as pickle files to avoid the need for retraining

every time a forecast is required. Our dashboard functions by taking in user inputs and sending them to our web server through an Application Programming Interface (API). These inputs are then passed into our saved models, and the resulting forecasts are sent back to the dashboard as an API response. Finally, these outputs are presented to the user on the dashboard (see Figure 7). For more information on our file systems and web server, see Appendix 5 and 6.

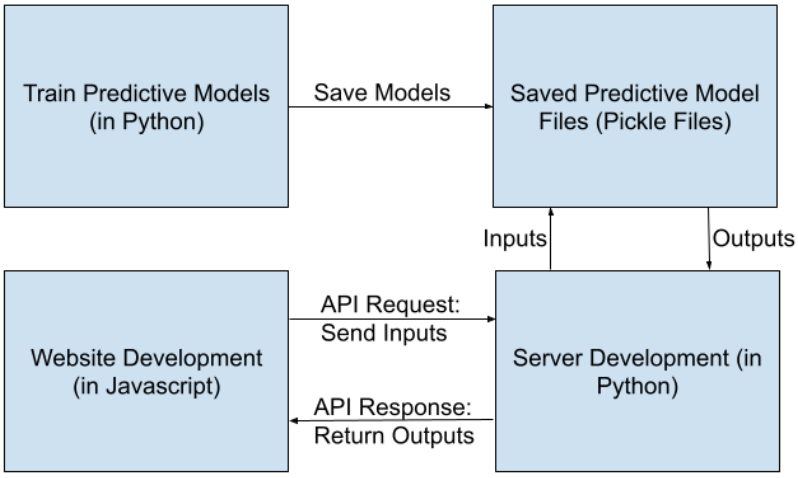

Figure 7: System Architecture that Supports our Dashboard

To ensure the solution remains effective and up-to-date, the predictive models can be retrained with new data. This process can be performed yearly, quarterly, or as our client sees fit, allowing the models to incorporate the latest enrollment trends and patterns. After updating the Excel files with new data, the client can execute the XGBoostTest.py script to retrain the XGBoost models accordingly. Upon completion, new models will be generated for each specified class size category and saved as pickle files. By replacing the previous pickle files with these updated ones in the same directory as the web server, the dashboard forecasts will seamlessly reflect the latest information. This process is designed to be user-friendly, requiring minimal technical expertise from department heads and enabling the Registrar's Office to maintain a current and efficient system over time.

Limitations and Future Developments

Data Availability

Our analysis faced significant limitations due to the unavailability of certain datasets that would have been extremely helpful to train and aid our models. We requested additional data from the NU Registrar, including Course and Teacher Evaluations (CTECs), historical department course forecasts, prerequisite requirements, waitlist information, and counts of declared majors. Unfortunately, these datasets were either too sensitive to be provided or not readily available.

The absence of CTEC data meant our models could not account for factors like professor reviews and course difficulty, which our background research indicated are significant predictors of class enrollment. To overcome the lack of such qualitative data, we implemented Latent Semantic Analysis (LSA), which we used to analyze course titles—allowing us to identify patterns and trends within course descriptions and incorporate these insights into our predictive models.

Future Data Integration

To improve the accuracy of our models, future development should focus on incorporating these missing datasets. Periodic updates to the dataset, such as quarterly or yearly retraining of the models with new enrollment data, will ensure the solution remains current and effective. This continuous data collection and integration will help the model adapt to new trends and changes in enrollment patterns.

Usability Limitations

One limitation users may encounter is when expected enrollment falls between two bin sizes (e.g., around 30 students). In such cases, predictions from our 0-30 and 31-70 models, with the same other inputs, may differ significantly. We advise users to run both models and use their intuition to select the prediction number that makes the most sense. This approach helps mitigate discrepancies and ensures more reliable decision-making.

Additionally, it's important to note that the output should not be the sole factor in forecasting class enrollment, but rather serve as a helpful resource in designing schedules. Users should use the model's predictions as a guide alongside their expertise and judgment to make the best decisions.

Model Expansion

Expanding the model's applicability beyond the McCormick School of Engineering & Applied Sciences to other schools will enhance its utility. This expansion will require retraining the models with school-specific data to capture unique predictors of enrollment within different academic units. This may require recalibrating bin sizes for each model, standardizing data, fine-tuning our LSA to accommodate new course titles, and further experimentation with relevant predictors. By tailoring the models to each school's needs, we can provide more accurate and relevant forecasts across the university.

Conclusion

Our XGBoost models provide the NU Registrar's office with accurate and actionable predictions for class enrollment, greatly enhancing their planning capabilities. Faculty within the McCormick School of Engineering and Applied Sciences (MEAS) can directly access these predictions through our interactive dashboard, which offers the flexibility to adjust parameters and receive tailored enrollment forecasts. This functionality is pivotal in ensuring that faculty can make informed decisions regarding class scheduling. Additionally, the capability to identify popular class times and export data to Excel further supports faculty in optimizing their course offerings.

Accurate enrollment forecasts mitigate several key issues: they prevent last-minute classroom changes, reduce the likelihood of course cancellations, and ensure that students can enroll in the courses they need. Our models demonstrate a high level of accuracy, with average forecast deviations of just 3 students for small classes, 8 for medium classes, and 26 for large classes. This precision is particularly beneficial for predicting small class enrollments, a historically challenging area for the Registrar's Office. Since small classes often include newer or advanced courses with unpredictable enrollment patterns, our solution provides substantial value by offering reliable forecasts where they are needed most.

Our XGBoost models, trained on a comprehensive dataset of courses offered between 2012 and 2023, form the foundation of our enrollment prediction solution. This extensive training period ensures that our models have learned from a wide variety of enrollment patterns, trends, and anomalies, thereby enhancing their initial accuracy. We have also developed a modeling framework that allows for periodic updates, which can be executed quarterly or yearly in alignment with the academic cycles. This means that as new class enrollment data becomes available, it can be seamlessly integrated with the existing historical dataset. By continually updating the dataset, our models can be retrained to incorporate the most recent trends and shifts in enrollment behavior. This retraining process ensures that the models remain relevant and accurate, reflecting the latest student preferences, curriculum changes, and other influencing factors.

We would like to thank Dustin Levell and the staff at NU Registrar's for their consistent assistance throughout the length of this project. Their willingness to answer questions and gather data and other resources were invaluable for the completion of our project. We appreciate them taking the time to meet with us each week and provide honest feedback about the progress and direction of our project.

We would also like to thank our faculty advisor, Imon Banerjee, whose presence and advice were invaluable at each stage of our project. His experience and knowledge regarding modeling, machine learning, and statistics helped us define the scope of our solution and better approach the problem while also being able to guide our group in an innovative direction.

Finally, we would like to send our appreciation and gratitude to our instructors and any unmentioned advisor for this project. Janice Mejia and Mike Watson were unwavering in their support for our project, and we thank them for continuing to push us to innovate and improve the small details. Our TA, Zhonghao Liu was readily available to provide support and answer any small questions that arose, which was a nice pillar to have. We would also like to extend our appreciation to our writing instructor, Kate Scharfenberg who was always willing to provide her high-level insight and feedback in our reports. This project had its obstacles and challenges but the support of our professors and other faculty members made the completion of this solution possible.

Appendix 1: Background Research

Northwestern University's Registrar's Office faces significant challenges in accurately forecasting class enrollment. Enhancing the accuracy of class enrollment forecasts will lead to more effective room assignments, reducing issues of classroom underutilization and overcrowding. To achieve this, it is essential to understand the drivers of demand for college courses and the industry-leading models commonly used for forecasting problems. Our background research aims to support our endeavors of generating more accurate forecasts, by exploring course selection motivations, time-series modeling approaches, and natural language processing techniques.

Understanding the drivers of demand for university courses helps identify possible predictors to include in our forecasting models. The study by Ma et al. (2021) offers critical insights into the motivations behind university students' course selections, identifying personal interest, career utility, grade potential, and peer recommendations as key factors. This framework guides our understanding of course demand and assists in identifying significant predictors for our model. For example, incorporating course evaluation (CTECs) textual data could be important in understanding student's *peer recommendations* for a course. Other data related to student motivations, like *why students are taking the course* and their *interest beforehand*, could similarly prove important for our model's accuracy.

However, incorporating CTEC data in predictive models can be a challenging task due to its complex relationship with class enrollment. Bedard and Kuhn chose to investigate the relationship between course evaluations and class size for courses at the University of California Santa Barbara (UCSB). Bedard and Kuhn's findings reveal a significant nonlinear negative effect of class size on evaluations of instructor effectiveness. However, in the context of our project, our model would consider the opposite relationship: the effect of student evaluations on future class size. Bedard and Kuhn's research emphasizes the complex relationship between course evaluations and class enrollment with both being dependent on the other. CTEC responses would likely be dependent on the current class size, whereas future class enrollment would likely be dependent on previous CTECs. This would diminish the efficacy of linear models, as they would be unable to capture the intricate interdependencies present in our dataset.

One method of incorporating these temporal relationships in our analysis is using a Long-Short Term Memory[1] (LSTM) model. A study conducted by researchers Abbasimehr and Yousefi explores the performance of an LSTM against several other statistical and computational modeling techniques in the context of forecasting. Abbasimehr and Yousefi conclude that Long-Short Term Memory (LSTM) models demonstrated the best performance measures for their experimental demand forecasting task. It is worth noting that an LSTM's unique architecture enables it to leverage both historical and present information when making forecasts. LSTMs have the added benefits of capturing nonlinear patterns in the time-series data and functioning without requiring the data to be stationary[2]. These attributes are particularly relevant for forecasting class enrollment as there are risks of nonlinearities and non-stationarity

[1] LSTMs are a type of neural network that use a hidden state vector to store historical information or "context" for a given time series. More specifically, it uses four interacting gates to manage storage and access of historical information.

[2] A time-series is stationary if the series has constant mean and variance over time. One common method for evaluating stationarity of a time-series is the Augmented-Dickey Fuller Test.

of our data. We are also encouraged by the superior performance when compared to other models such as simple neural networks and k-nearest neighbors for forecasting tasks. Despite these advantages, representing course enrollment data as a time-series is a difficult task. A course may offer multiple course sections during the same quarter and year, meaning there may be multiple observations for a course at a given point in time. Thus, it is nearly impossible to represent course enrollment sections as a time-series, rendering LSTMs an ineffective modeling choice for forecasting course enrollment.

With the difficulties of obtaining this data, as well as the aforementioned challenges of representing its relationships in modeling approaches, it becomes imperative to explore alternative problem-solving strategies. One such avenue is illuminated by Dianne Lee's research at Harvard University, where she conducted a comprehensive study on applying mathematical prediction models to course enrollment estimation. Lee's research developed several predictive models, distinguishing between new and continued courses. Her findings revealed that machine learning models, particularly decision tree models, outperformed human baselines in predicting enrollment for existing courses. Interestingly, the efficacy of the machine learning model heavily depended on the course's historical data availability.

Further differentiating between new and continued courses, Lee demonstrates that the relevance of predictive features varies significantly depending on the nature of the course; for example, title relevance was a significant predictor for new course offerings, while historical enrollment data was more critical for continued courses. This not only helps us pinpoint potential important predictors that we may want to seek more data for, but it is also helpful for us as we decide how best to structure our solution, potentially requiring different estimation approaches for new and continued courses.

While Lee’s recommendations for modeling newly offered courses are invaluable, the utilization of course titles as predictors poses a formidable challenge, requiring advanced methodologies. Gefen et al. (2017) propose employing Latent Semantic Analysis, a statistical technique that can be used to represent course titles as a series of concepts. This method involves constructing a matrix that includes each course title and the occurrences of terms within those titles. Latent Semantic Analysis is then applied to convert the matrix into a small number of dimensions or “concepts”. Each course would receive a score for each concept, indicating the concepts' relevance to the course. By leveraging these scores in subsequent modeling, we can effectively integrate course titles as predictors in our models. This will be invaluable for our forecasts, especially with newly offered courses which lack historical information relevant for predictions.

In conclusion, our research identifies key factors likely to influence class enrollment and underscores the potential of advanced models like Random Forests. This is of critical importance to the Registrar’s Office as enhancing the accuracy of forecasts can significantly improve room assignments, thereby mitigating issues of classroom underutilization and overcrowding. While our research presents promising avenues for improving forecasting accuracy, we also acknowledge the challenges inherent in incorporating complex data such as CTECs. Our research aims to provide actionable insights and methodologies to support the Registrar’s Office in addressing classroom allocation challenges. Through the development and implementation of advanced forecasting models, we strive to contribute to more efficient resource management within Northwestern University, ultimately enhancing the academic experience for students and instructors alike. We hope that our efforts will prevent last-minute changes in classroom assignments and cancellations of courses.

Appendix 2: Data and Software Resources

Data Resources

We have received two datasets, which were provided to us by Northwestern University's Registrar's Office. Our primary dataset includes enrollment information on over 84,000 Mccormick Engineering & Applied Sciences (MEAS) course sections offered from Fall of 2012 to Fall of 2023. Each row within the dataset corresponds to a single section for a course and includes the following relevant information: term, subject, catalog number, title, component[3], enrollment capacity[4], total enrollment, start date, end date, instructor, start time, end time, and binary indicator variables for the days of the week that the course section is offered. Each row also includes registrar-specific data such as term code, section, class number, class type, and course owner, which are irrelevant for our modeling. It's important to emphasize that aside from times and dates, almost all potentially relevant predictors in the dataset are categorical in nature.

Our secondary dataset includes detailed course descriptions for over 1,200 Mccormick Engineering & Applied Sciences courses. Unfortunately, this dataset only contains course descriptions for courses currently being offered, making it unsuitable as a supplement to our primary dataset for forecasting course enrollment.

Software Resources

We used the Python programming language to implement our exploratory data analysis and machine learning models. Python provides useful libraries specific to data analysis and machine learning, which includes pandas, NumPy, scikit-learn, XGBoost, SciPy, and Matplotlib. For front-end development of our dashboard, we used the JavaScript programming language. Specifically, we used the React framework to create reusable components for our application, and we fetched enrollment predictions from our machine learning models by creating API Endpoints on a locally hosted web server. This server was created in Python, leveraging the Flask library to circumvent server expenses. Nonetheless, the option exists to create a serverless lambda function on Amazon Web Services to ensure continuous and efficient handling of API requests.

Our dashboard functions by 1) taking in user input, 2) passing this input to our machine learning models via API, 3) sending a class forecast response to our front-end, and 4) displaying the returned information on our dashboard (see Figure 8).

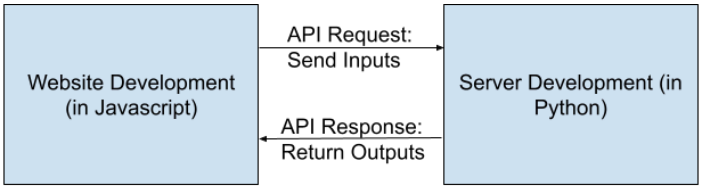

Figure 8: Overview of the Web Server's Interaction With the Dashboard

[3] Components include lectures, discussions, labs, and seminars.

[4] Enrollment capacity is the maximum capacity for a class, as displayed by Caesar. It is determined using anticipated enrollment numbers and maximum enrollment limits requested by instructors.

Appendix 3: Preprocessing

In data preprocessing, both row and column manipulation play critical roles in refining datasets for analysis. Alongside these processes, Latent Semantic Analysis (LSA) serves as a powerful technique for uncovering underlying patterns in textual data. After completing all preprocessing steps and applying Latent Semantic Analysis (LSA), the original dataset (see Figure 9) with dimensions (224451, 25) has been transformed into a new dataset with dimensions (8971, 58).

	Term Code	Term	Subject	Catalog	Descr	Section	Class Nbr	Component	Class Type	Enrollment Capacity	Total Enrollment	Start Date	End Date	Course Owner	Course Career	Instructor	Start Time	End Time	Mon	Tues	Wed	Thurs	Fri	Sat	Sun
0	4560	2014 Fall	MUS_THRY	316-0	Renaissance Counterpoint	20	16442	LEC	Enrollment Section	20	26	2014-09-23	2014-12-06	MUSIC	TGS	Gjerdingen,Robert O	09:30:00	10:50:00	N	Y	N	Y	N	N	N
1	4560	2014 Fall	POLI_SCI	499-0	Independent Study	77	18063	IND	Enrollment Section	10	0	2014-09-23	2014-12-06	WCAS	TGS	Stevens,Jacqueline	12:00:00	12:00:00	N	N	N	N	N	N	N
2	4560	2014 Fall	POLI_SCI	499-0	Independent Study	78	18064	IND	Enrollment Section	10	0	2014-09-23	2014-12-06	WCAS	TGS	Tillery,Alvin Bernard	12:00:00	12:00:00	N	N	N	N	N	N	N
3	4560	2014 Fall	THEATRE	448-0	Studies in Am Theatre & Drama	20	19022	LEC	Enrollment Section	15	5	2014-09-23	2014-12-06	SPCH	TGS	Young Jr,Harvey	14:00:00	16:50:00	N	N	Y	N	N	N	N
4	4560	2014 Fall	ENGLISH	206-0	Read/Wrt Poetry	24	19125	SEM	Enrollment Section	15	12	2014-09-23	2014-12-06	WCAS	UGRD	Kinzie,Mary	11:00:00	12:20:00	N	Y	N	Y	N	N	N

Figure 9: Head of Original Dataset

Column Preprocessing

The data cleaning process included dropping several columns to refine the dataset for better predictive power. It's essential to keep only relevant columns for the model because each column we train on requires a user input. This helps in simplifying the user interface and enhances the overall user experience by eliminating unnecessary inputs. To achieve this goal, our preprocessing of data columns involved a combination of feature engineering, dropping columns, and converting data types.

Feature Engineering

1. Subject-Catalog:
    - Created by combining the Subject (e.g., IEMS) and Catalog (e.g., 225).
    - Justification:
        - Provides an identifier for each course, enhancing predictive power and simplifying user input.
2. Duration:
    - Created by subtracting start and end time.
    - Justification:
        - More convenient user input than end time while also maintaining model performance, simplifies input process.
3. Year:
    - Extracted from Start Date to easily filter data based on years.
    - Justification:
        - Useful for creating train and test sets based on years; <u>not</u> used as input for predictions.

Dropping Columns

1. Start Date & End Date:
    - Dropped as they offer no predictive power; only the academic term matters.
    - Justification:
        - Removed after calculating duration.
        - Simplifies user input and data structure, with comparison between models showing no significant loss in accuracy.
2. Course Owner:
    - Dropped since all courses are McCormick courses.
    - Justification:

        - Redundant information as all entries share the same value (given our focus in McCormick), simplifying the dataset.
3. Course Career:
    - Dropped after testing showed it didn't help models performance.
    - Justification:
        - Irrelevant to model performance, reduces complexity.
4. Class Nbr, Section, Term Code:
    - Dropped because they are clerical and offer no predictive power.
    - Justification:
        - Administrative data that could introduce erroneous relationships in our model, leading to poor performance on testing data.
5. Class Type:
    - Removed after testing showed Enrollment Section versus Non-Enrollment Section type had little predictive power.
    - Justification:
        - Minimizes user input complexity without compromising model accuracy.
6. Catalog:
    - Dropped after creating Subject-Catalog.
    - Justification:
        - The catalog number alone is not a predictor for course enrollment.
        - Including it could introduce erroneous relationships in the model.

Convert Data Types

1. Convert numerical data into categorical types:
    - Only relevant to Start Time and Duration, which we converted from object to numerical data type for preprocessing.
    - Justification:
        - Numerical data could result in misleading relationships for our model.
        - It would imply that later start times and longer classes result in higher enrollment, which is unrealistic and not necessarily true.
2. Convert columns from object to categorical types:
    - Enables efficient handling by XGBoost, which can natively handle categorical data.
    - Justification:
        - Improves processing efficiency and model performance by treating discrete categories appropriately.
        - Avoids the pitfalls of one-hot encoding:
            - Memory limitations
            - Sparse feature issues
        - Leverages XGBoost's optimal partitioning algorithm for better accuracy and resource management

Row Preprocessing

The data cleaning process included substantial preprocessing of the 224,451 unique course entries (rows). Important steps included:

1. Filtering out Irrelevant Data Points
    - Filtered for McCormick courses:
        - Limit the scope to McCormick courses, as suggested by our client.
        - Justification: Focuses analysis on relevant subset.
    - Filtered for lectures, seminars, discussions, and labs:
        - Focus on standard instructional formats.
        - Justification: Ensures consistency in course types.

    - Excluded special topics courses:
        - Additional filtering of courses with generic and uninformative titles.
        - Justification: Removes ambiguity in course content categorization, as there are many different types of special topic courses.
    - Excluded single-offering courses:
        - Focus on regularly offered courses.
        - Justification: Improves model reliability by excluding anomalies.
    - Excluded low-frequency durations:
        - Include class time durations that appear at least 100 times in our dataset.
        - Justification: Eliminates outliers associated with atypical class durations.
2. Handling Missing Data
    - Dropping rows with NaT times:
        - Justification: Maintains data integrity.
3. Handling Potentially Misleading and Erroneous Data
    - Removing early morning and late night classes (7 PM - 6 AM): Exclude unlikely class times.
        - Justification: Eliminates implausible data and unusual scenarios.
    - Removing courses with 0 and 999 capacities: Exclude outliers from our dataset that represent non-existent or irrelevant class sizes.
        - Justification: Removes misleading data.
4. Handling Extraordinary Data and Outliers
    - Removing COVID-19 period courses: Exclude classes from Spring 2020 to Spring 2021 as classes were online during COVID-19, inflating class enrollment numbers.
        - Justification: Avoids distortion from pandemic-related anomalies.
    - Removing classes with 0 enrollment capacity: Further refine data by removing classes that show no student enrollment.
        - Justification: Ensures that only actively attended classes are analyzed.
    - Removing classes with less than 5 students in total enrollment: Focus on classes with significant attendance.
        - Justification: Prioritizes classes with meaningful enrollment data and avoids potentially incorrect total enrollment data.
5. Duplicate Data
    - Removing duplicates: Ensure uniqueness in course entries.
        - Justification: Enhances data accuracy.

LSA Processing

We used Latent Semantic Analysis (LSA) to process course titles, extracting scores for various concepts, with higher scores indicating greater relevance of a concept to a course. By utilizing these scores in subsequent modeling, we can effectively incorporate course titles as predictors in our models (see Appendix 1 for more details).

1. Preprocessing:
    - Convert descriptions to lowercase.
    - Tokenize descriptions.
    - Apply lemmatization.
    - Remove slashes, hyphens, and non UTF-8 characters.
2. Word Removal:
    - Remove stopwords and specific unwanted tokens.
    - Eliminates noise from the data, enhancing LSA interpretability.
3. Replacements:

- Replace custom terms with standardized terms tailored to McCormick and engineering terminology (uses a single term to represent close variations of the same term)
- Replace common bigrams with a single token (e.g., "mechanical engineering analysis" converted into "mechanicalengineering")
    - Preserves meaningful phrases in the analysis.

4. TF-IDF Vectorization:
    - Convert processed text to a TF-IDF matrix
    - Remove rows with all zero values in a TF-IDF matrix (caused by preprocessing out all of the words in 'Desc') to avoid numerical stability issues during SVD.
    - Set min_df to 15 as the refinement process showed that pruning terms appearing less frequently led to more interpretable loading
5. SVD Transformation:
    - Perform singular value decomposition (SVD) to get LSA loadings.
    - Choose the optimal number of components based on explained variance, with 40 components chosen by evaluating the elbow point of the plot in Figure 13 in Appendix 4.
6. Iterate steps 2-7 until LSA loadings are interpretable.
7. Add a course's scores for each of the 40 LSA components to the dataframe (18->58 columns).

Appendix 4: Modeling and Analysis

In our approach to forecasting class enrollment, we strive to implement simple, easy-to-maintain, and highly-interpretable models as requested by the NU Registrar's office. This process involves evaluating the historical accuracy of prior forecasted enrollments by NU departments and comparing these predictions to those of our several regression models, which include linear regression, both with and without regularization, as well as a predictive model using the XGBoost algorithm. By understanding the shortcomings of prior department forecasts and creating multiple predictive models for this task, we establish a robust framework to propose and justify a particular forecasting method for our clients.

Predictive Model Formulation

The objective of these models is to minimize the mean squared error of predicted enrollment and actual enrollment. In less technical terms, this can be interpreted as trying to make our forecasted enrollment as close as possible to actual enrollment, with greater penalty being placed on less accurate forecasts. This metric, as well as our modeling relationships are represented in the following equations:

$$Mean\ Squared\ Error \quad = min \sum_{i \in classes} (Predicted\ Class\ Enrollment_i - Actual\ Class\ Enrollment_i)^2$$

$$Predicted\ Class\ Enrollment = Model(term,\ subject\ catalog,\ subject,\ instructor,\ start\ time,\ duration,\ days\ of\ the\ week,\ course\ topics,\ anticipated\ enrollment\ range)$$

Preprocessing

Our preprocessing methodology involved processing of columns, rows, and textual course titles. For detailed information, please refer to Appendix 3.

Representing Cross-Listed Courses

We addressed the issue of cross-listed courses by assuming that each enrollment section was its own class. This allows individual departments to forecast their sections, with total forecasted enrollments being the sum of each department's sections.

Grouping Courses By Expected Enrollment Size

Based on our exploratory data analysis, we've classified each class section into small, medium, and large sizes. Our findings suggest that departmental predictions are less accurate for smaller courses. Illustrated in our exploratory data analysis (see Figures 10-12), we observe a weaker correlation between enrollment capacity and actual enrollment for smaller courses compared to larger ones. This points towards lower accuracy in enrollment forecasting for smaller classes, reflecting the qualitative trends observed by the Registrar's Office. In the subsequent modeling section, we introduce additional analysis that establishes optimal enrollment ranges for small, medium, and large classes.

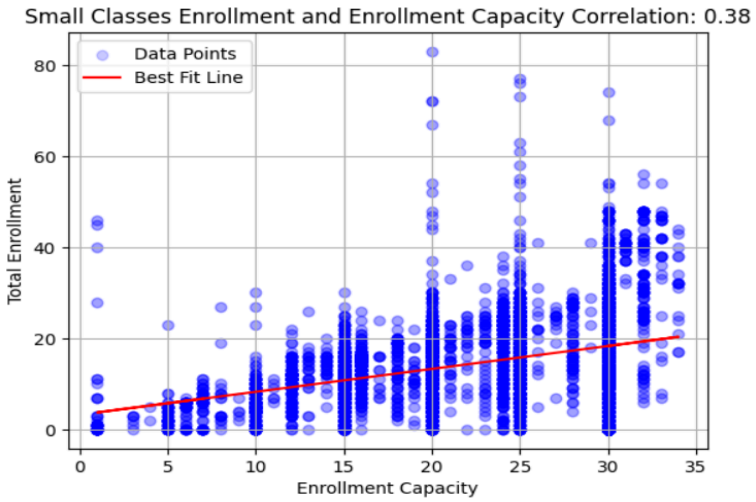

Figure 10: Small Classes Enrollment and Enrollment Capacity Correlation

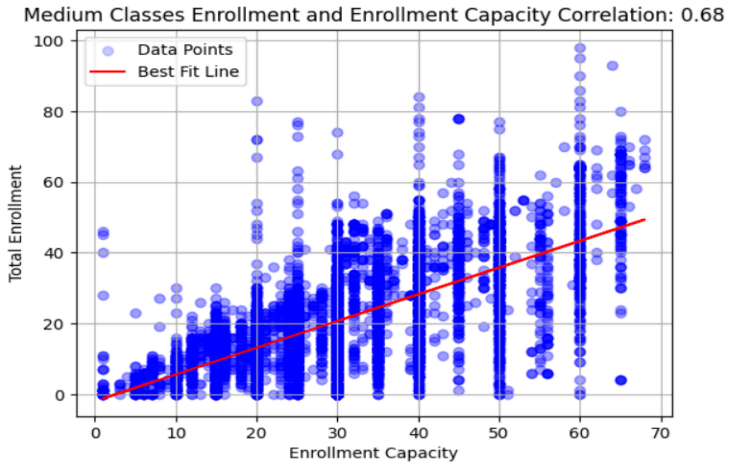

Figure 11: Medium Classes Enrollment and Enrollment Capacity Correlation

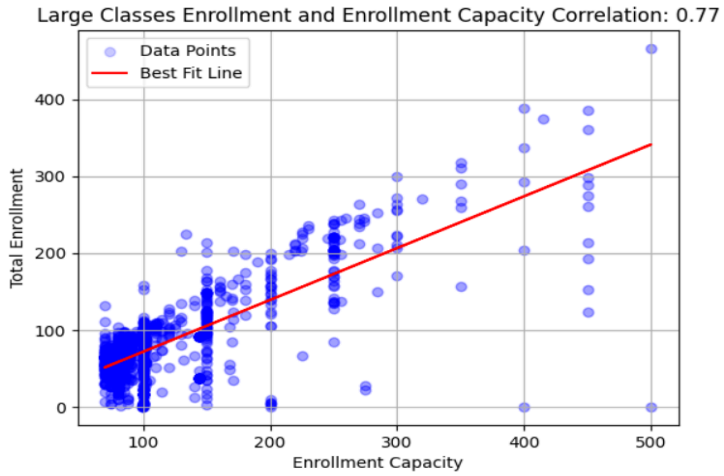

Figure 12: Large Classes Enrollment and Enrollment Capacity Correlation

Modeling

To provide a baseline model for our prediction, our group implemented Linear Regression models. The simplicity of Linear Regression makes it a natural baseline, providing a foundational understanding of relationships within the data without the complexities of more sophisticated models. The incorporation of regularization techniques, through Lasso and Ridge Regression, addresses the potential for overfitting and the inclusion of irrelevant variables, offering a balance between complexity and predictive power. Although Linear Regression models offer simplicity, we expected the models to perform poorly due. This is due to the fact that linear regression models can only capture linear relationships between variables, whereas we suspected that nonlinear relationships were likely to be present.

The shortcomings of a linear regression model guide our subsequent modeling choices. Based on our background research, random forests are effective in forecasting enrollment due to their ensemble method, which allows for handling diverse data types. However, models that can handle a high number of categorical variables, like decision trees, were subject to heavy overfitting on the testing set. Therefore, we opted to use XGBoost for its advanced capabilities in model refinement. XGBoost works by sequentially adding weak learners—typically simple models like decision trees—that individually may not perform well but collectively lead to strong predictive performance. This method uses a gradient descent algorithm to optimize the loss functions, thereby iteratively minimizing errors. Compared to random forests, which build numerous decision trees independently and average their predictions, XGBoost focuses more precisely on correcting previous errors. While both methods use an ensemble approach that can complicate interpretability, neither method is inherently more explainable than the other. Therefore, we choose to employ the method with better performance, which is XGBoost in our case.

Our initial model, trained on the entire dataset, presented deceptively acceptable performance overall. However, a closer analysis of residual plots and accuracy across different class sizes revealed significant shortcomings, particularly for small and medium classes. This discrepancy highlighted the necessity for a tailored approach, as a single model for all class sizes assumes uniform predictor relevance, which proved inaccurate, especially given the department's challenges with accurately forecasting enrollments in smaller classes. Despite a high percentage of predictions falling within a single standard deviation (~31) of the true enrollment, the model performed poorly for small and medium class sizes, highlighting the need for differentiated modeling strategies.

To address these issues, we hypothesized that predictors vary in importance across different class sizes: small, medium, and large. Therefore, we trained distinct models for each size category, based on the premise that each class size exhibits unique characteristics requiring different predictor weights. This method aims to enhance the specificity of forecasts but might reduce the generalizability of each model across other class sizes.

To validate our approach, we cross-validated bin sizes for class categorization, testing various enrollment ranges for small, medium, and large classes. Our selection process for these ranges was carried out in increments of 5, balancing the accuracy and usability of our dashboard interface. The optimal groupings, which yielded the highest validation accuracy, were established as: 0-30 students for small classes, 31-70 students for medium classes, and 71 or more students for large classes. We include a thorough discussion of how we determined these groupings in the results section.

By creating these distinct models, we are implicitly assuming that predictors influence total enrollment differently across the three class sizes. This tailored approach is not only justified by the empirical data from the residual plots but also by the improved accuracy in forecasts across the different class sizes as can be seen in our results section, providing a nuanced understanding and better predictability of course enrollments.

Latent Semantic Analysis

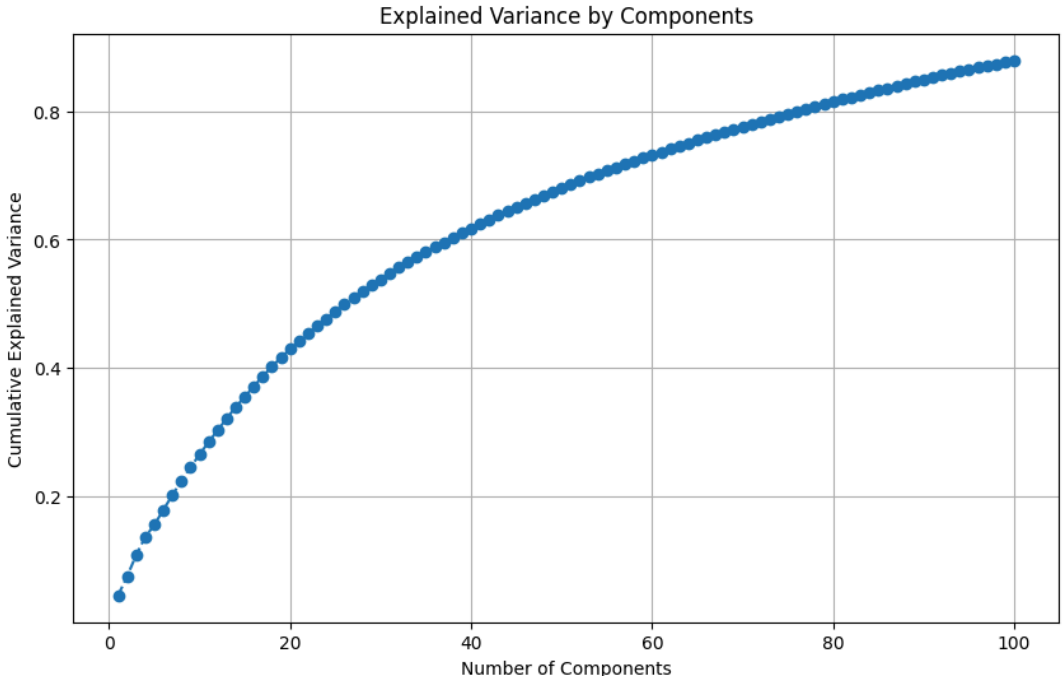

Figure 13: Explained Variance by Number of LSA Components

After identifying the optimal number of components through the elbow method in the explained variance plot (see Figure 13), we configured our Latent Semantic Analysis (LSA) to produce 40 meaningful component loadings. Each component is defined by the top 10 terms that offer deep insights into the underlying themes of course descriptions. For example, terms like “engineeringanalysis,” “honor,” and “experience” contribute to an LSA component reflecting the relevance to “Engineering Analysis” courses, where courses such as EA1, EA2, EA3, and EA4 are expected to score highly on this component.

Each course title in our dataset is represented by scores for these 40 concepts (see Table 1), which quantitatively reflect the relevance of each theme to the course content. These scores serve as features in our predictive model, enhancing its ability to forecast course enrollments by incorporating detailed, thematic understandings of course content. This approach not only improves model accuracy but also provides a richer, more nuanced analysis of course dynamics.

Component	Name	Description
1	Experience_FirstYear_LSA	First Year Seminars
2	Communication_Design_LSA	Design thinking and design communication
3	Topics_PersonalDev_LSA	Topics central to personal development
4	Intro_Design_LSA	Fundamental principles of design
5	Design_Project_LSA	Processes involved in design projects
6	EngAnalysis_LSA	Engineering Analysis Courses
7	DTC_Method_LSA	DTC (Design Thinking & Communication)
8	Computing_Program_LSA	Fundamentals of computing and programming
9	Seminar_Manage_LSA	Discussions focused on management strategies
10	Statistics_Data_LSA	Statistical analysis driven by data
11	Material_Process_LSA	Techniques in materials processing
12	Systems_Physics_LSA	Physics principles in systems engineering
13	Mechanics_Fluids_LSA	Studies related to fluid mechanics
14	Project_Management_LSA	Techniques for managing complex projects
15	Selection_MechEng_LSA	Selection criteria in mechanical engineering
16	Selection_Optimization_LSA	Strategies for optimal selection processes
17	Optimization_Modeling_LSA	Optimization techniques in modeling
18	Analysis_Program_LSA	Analytical methods in programming
19	Probabilities_Program_LSA	Concepts in probabilistic programming
20	Analysis_Human_LSA	Analysis of human factors
21	Data_Structure_LSA	Data Structures Courses
22	Mechanical_Manufacturing_LSA	Processes in mechanical manufacturing
23	Thermodynamic_Env_LSA	Thermodynamics in environmental applications
24	Environmental_Modeling_LSA	Modeling of environmental systems
25	Principles_Entrepreneurial_LSA	Foundational principles of entrepreneurship
26	Discourse_Statistics_LSA	Analysis of statistical discourse
27	Machine_Learning_LSA	Applications of machine learning
28	Modeling_Stochastic_LSA	Techniques in stochastic modeling
29	Fundamentals_Transport_LSA	Basic understanding of transport systems
30	Construction_Manage_LSA	Management of construction projects
31	Develop_Career_LSA	Career development
32	Advance_Communication_LSA	Advanced methods in communication

33	Life_Designing_LSA	Strategies for designing personal life
34	Life_DesigningComm_LSA	Designing community-oriented life strategies
35	Structure_Theoretical_LSA	Concepts in theoretical structures
36	ResponsibleConduct_Train_LSA	Responsible Conduct of Research course
37	Construction_Computing_LSA	Use of computing in construction
38	Advanced_Analytical_LSA	Advanced techniques in analytical methods
39	Management_Transport_LSA	Management of transportation
40	Simulation_Disc_LSA	Simulation

Table 1: Components of LSA Model

Additionally, we performed the following replacements to enhance LSA interpretability:

- Custom Replacements:
    - Standardized terms, replacing 233 abbreviations and variants.
    - E.g. converting 'engg', 'eng', and 'engineerng' to 'engineering'.
    - Reduces data sparsity and enhances semantic similarity across the dataset.
- Bigram/Trigram Replacements:
    - We merged 76 common bigrams and trigrams into single tokens.
    - E.g "engineering analysis" to "engineeringanalysis".
    - Objective is to maintain contextual integrity.

These preprocessing efforts are essential for refining LSA input data, leading to more effective feature extraction and improving the model's predictive accuracy. To evaluate the effectiveness of incorporating LSA concepts, we trained models both with and without these LSA features. This approach allows us to ascertain whether the inclusion of LSA enhances model performance and to quantify the extent of its impact.

Results

Since historical department forecasts for class sizes are unavailable, to assess the performance of our models to a baseline, we must devise a method to assess current forecasting accuracy. To this end, we propose that "enrollment capacity" in CAESAR, which reflects both the limits set by departments and the constraints imposed by available classroom sizes, is correlated with "total enrollment." Consequently, while the "enrollment capacity" metric in our dataset does not directly indicate the forecasting accuracy of NU departments, its correlation with "total enrollment" provides crucial insights into where departments may be encountering predictive challenges. This serves as a practical proxy for evaluating departmental forecast performance, especially in the absence of historical data.

We assess our linear and XGBoost models in the test set using two metrics: root mean squared error (RMSE) and accuracy[5]. As depicted in Figures 14-19, our most accurate forecasts result from using three XGBoost Models, with one model used for each class size grouping (e.g. small, medium, large). These models produced accuracies of over 80% for all class size groupings, which is a remarkable improvement when compared to linear models which produced accuracies of 60-70% across the class size groupings. Our results can be evaluated

[5] Accuracy is determined by the percentage of predictions that fall within one standard deviation of the range for each class size grouping (e.g., small, medium, large).

through a comparison of using a single versus multiple models, comparison of model types, comparison when using LSA, and comparison across class size groupings.

1. *Comparison of A Single Versus Multiple Models*
    a. Comparing accuracy and RMSE when using a single model versus three models, we notice significantly higher accuracy and lower RMSE when using the three models. Our approach involves training distinct models for small, medium, and large classes, based on the assumption that each class size exhibits unique characteristics that require different predictor weights.

2. *Comparison of Models Types*
    a. When using multiple models, one for each class size category, our Linear Models with and without regularization consistently performed worse than our XGBoost Models (see Table 2). This is likely because XGBoost Models can capture non-linearities and are better equipped to handle categorical data.
    b. When "enrollment capacity" is used as a departmental forecast of "ideal classroom size," it underperforms compared to both our XGBoost and Linear models. This highlights the limitations of relying solely on departmental estimates for accurate enrollment forecasting, especially for small and medium-sized classes (see Table 3).

3. *Comparison When Using LSA*
    a. RMSE and accuracy were quite similar when LSA was used and when LSA was not used. However, we anticipate that future analysis would demonstrate using LSA can significantly improve forecasts for newly offered courses. This is because course title would be a far more important predictor than historical data for newly offered courses according to our background research in Appendix 1.

4. *Comparison Across Class Size Groupings*
    a. As expected, accuracy was lowest for predicting small class sizes. This is likely because smaller classes are likely to be electives rather than requirements for majors, making their enrollment less predictable. This supports our exploratory data analysis findings and qualitative trends identified by the Registrar’s office.

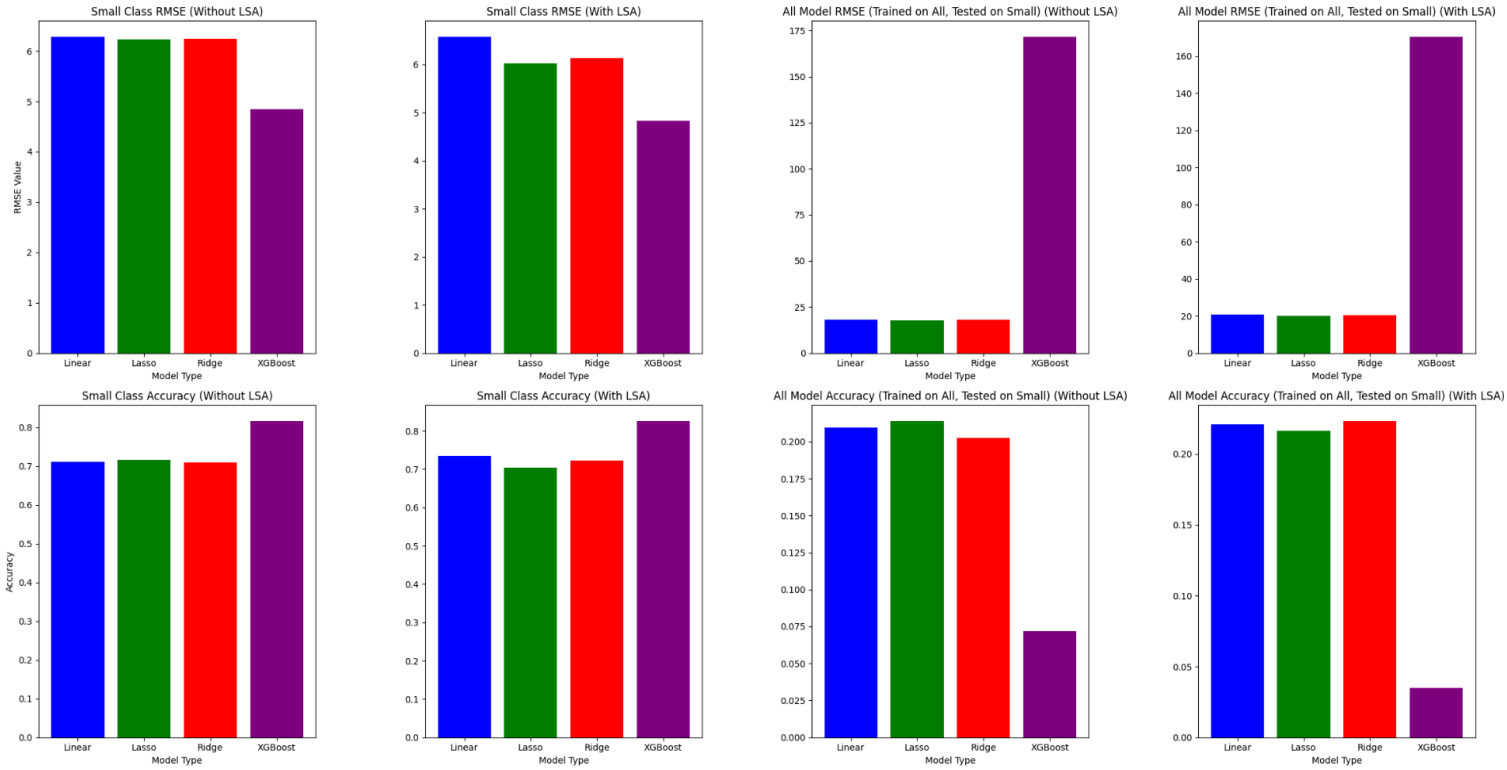

Figure 14: Accuracy Metrics for Total Enrollment <=30

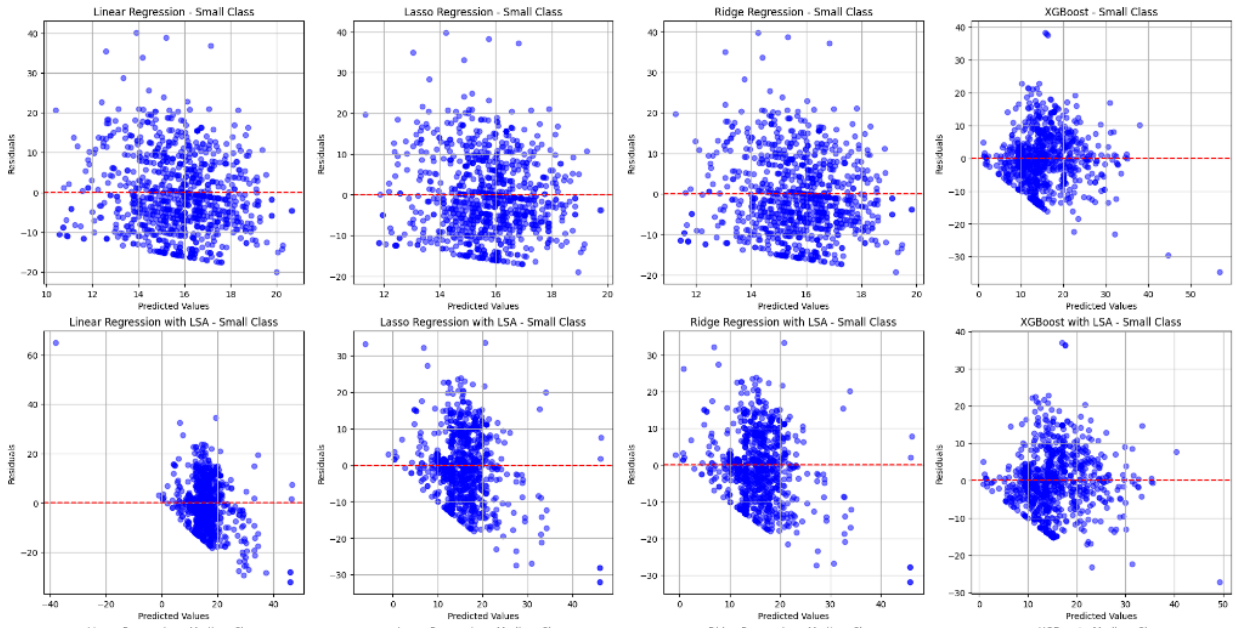

Figure 15: Residual Plots for Total Enrollment <=30

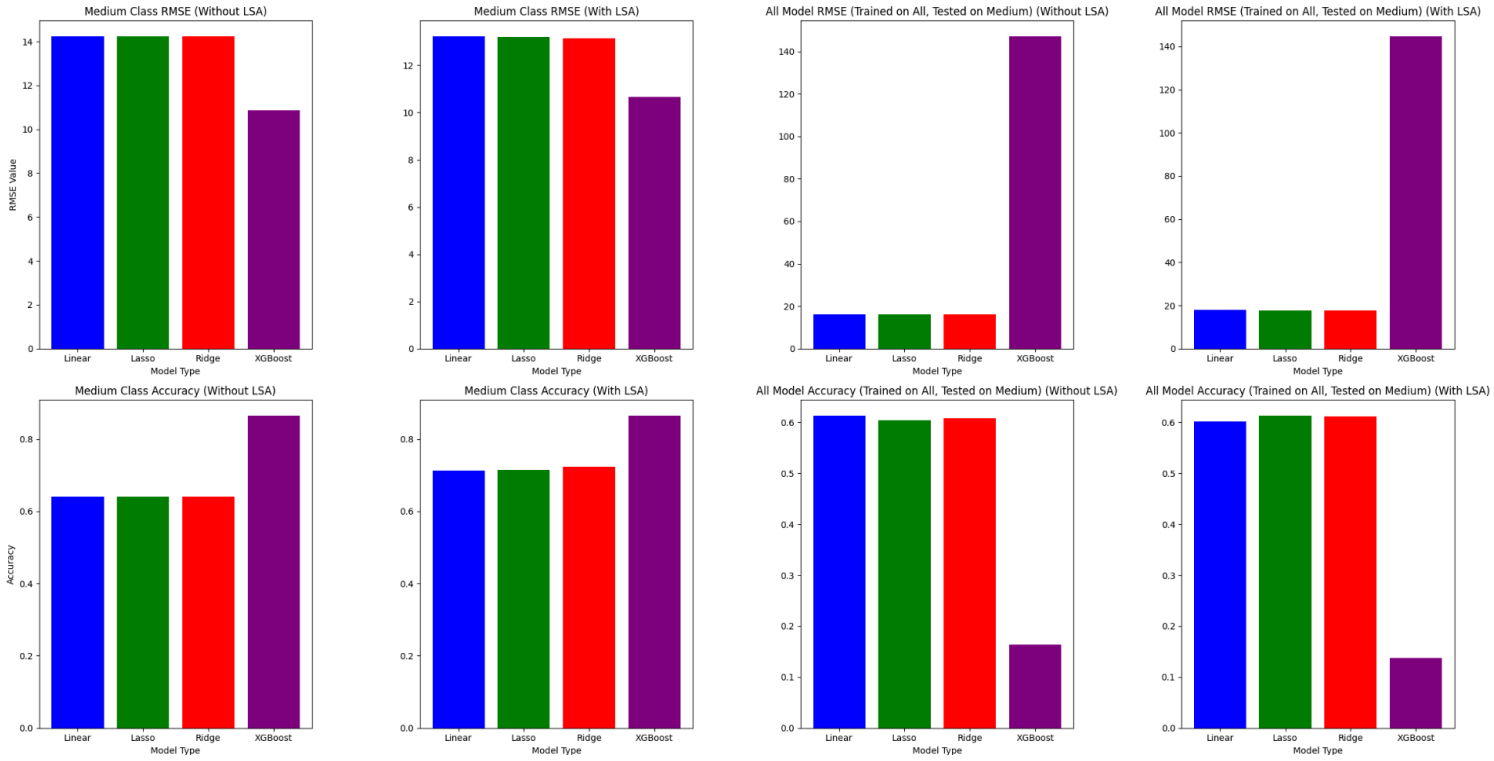

Figure 16: Accuracy Metrics for 31<= Total Enrollment <=70

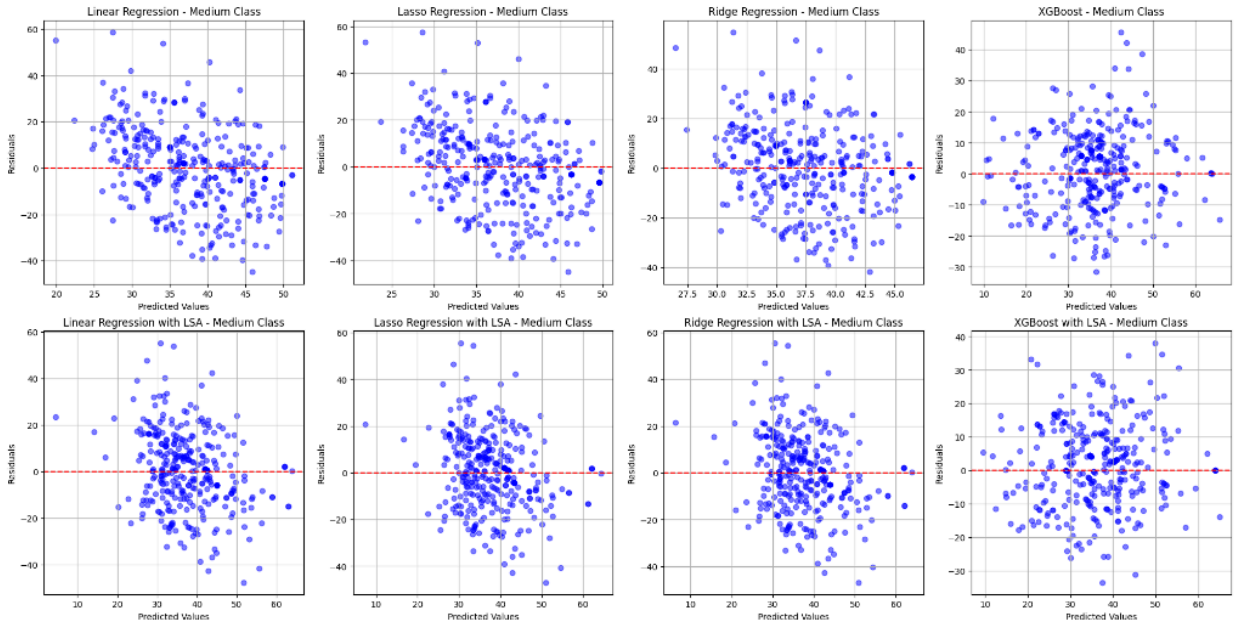

Figure 17: Residual Plots for 31<= Total Enrollment <=70

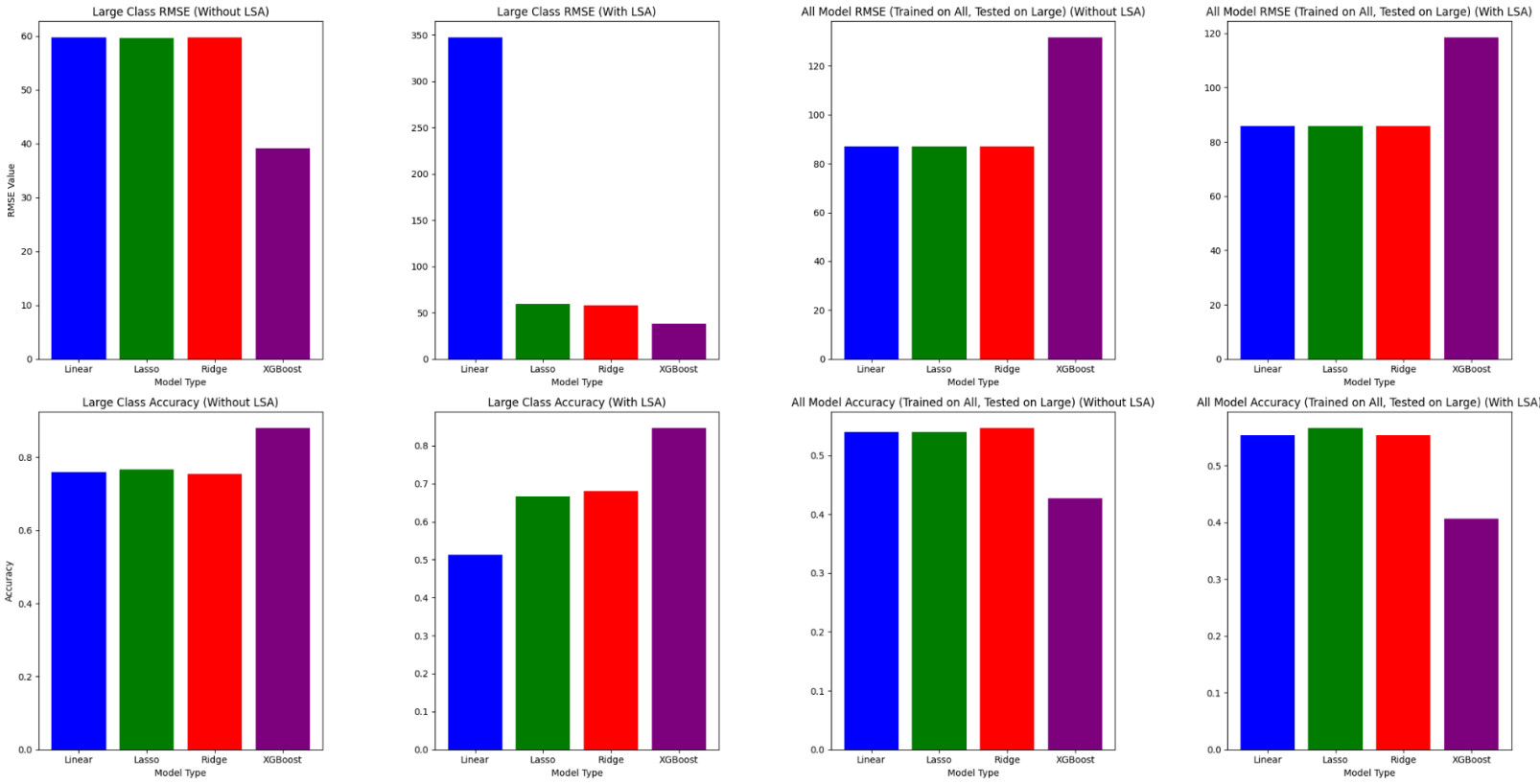

Figure 18: Accuracy Metrics for Total Enrollment >=71

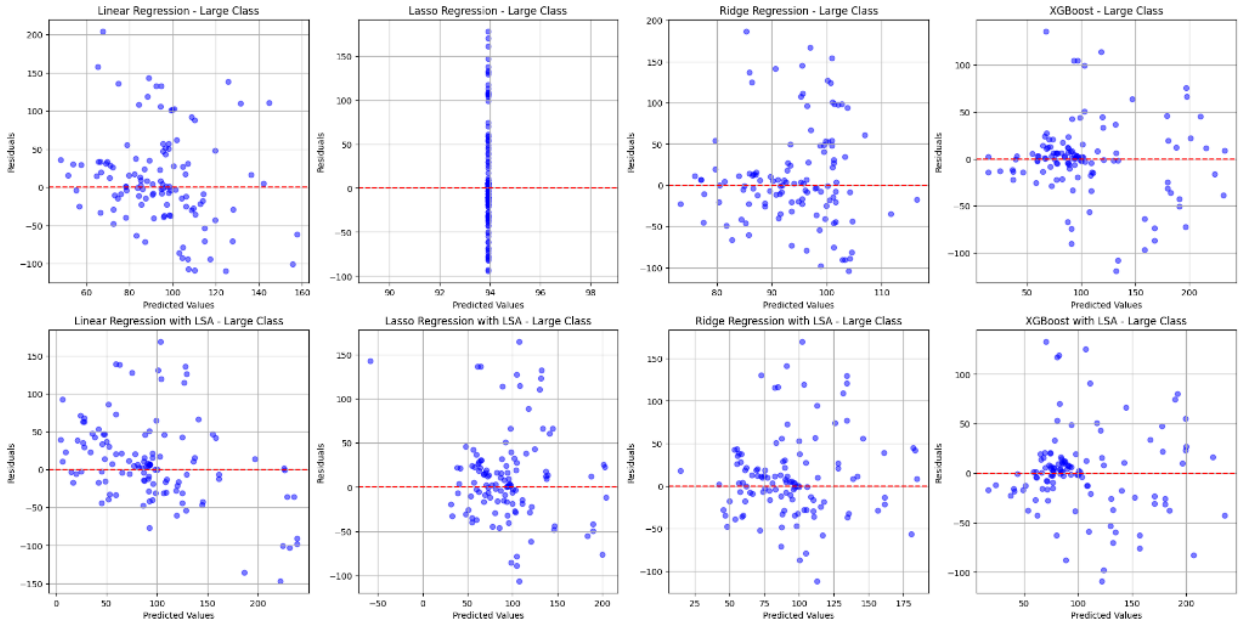

Figure 19: Residual Plots for Total Enrollment >= 71

Optimal Bucket Size Analysis for XGBoost Model
To determine the best bucket size for the XGBoost model, we evaluated the performance metrics across different bin sizes for small, medium, and large classes. We focused primarily on Scaled Test RMSE—defined as RMSE divided by the standard deviation of the bin size—to enable comparison across different class sizes, considering our defined accuracy metric as a secondary evaluation criterion. The analysis and recommendations are based on the data presented in Tables 2-4.

Small Classes
- Scaled Test RMSE:
    - The lowest Scaled Test RMSE for small classes is 0.740398 for the 5-30 bin size (see Table 4).
- Accuracy:
    - The highest accuracy for small classes is 0.848392 for the 5-35 bin size (see Table 4).

While the 5-35 bin size has slightly better accuracy, the 5-30 bin size has the lowest Scaled Test RMSE, indicating it performs better in terms of reducing error relative to other models and splits.

Medium Classes
- Scaled Test RMSE:
    - The lowest Scaled Test RMSE for medium classes is 0.731084 for the 31-70 bin size (see Table 4).
- Accuracy:
    - The highest accuracy for medium classes is 0.864964 for the 31-70 bin size (see Table 4).

The 31-70 bin size has both the lowest Scaled Test RMSE and the highest accuracy, making it the best choice for medium classes.

Large Classes
- Scaled Test RMSE:
    - The lowest Scaled Test RMSE for large classes is 0.647087 for the 81+ bin size (see Table 4).
    - The 71+ bin offers the second best scaled RMSE (0.661) among the options.
- Accuracy:
    - The highest accuracy for large classes is 0.878261 for the 81+ bin size (see Table 4).
    - The 71+ bin has the worst accuracy amongst the three splits.

The 81+ bin size has both the lowest Scaled Test RMSE and the highest accuracy, making it the best choice for large classes.

Optimal Model Split
Based on the Scaled Test RMSE and accuracy metrics, the optimal model split consists of:
- Small Classes:
    - 5-30 bin size
- Medium Classes:
    - 31-70 bin size
- Large Classes:
    - 71+ bin size

Justification

- The 5-30 bin size for small classes has the lowest Scaled Test RMSE, which is crucial given the client's biggest issue with small classes.
- The 31-70 bin size for medium classes shows the best performance in terms of both Scaled Test RMSE and accuracy.
- The 71+ bin is chosen to maintain consistency with the 5-30 and 31-70 bins used for small and medium classes, which are a higher priority for the client. Its Scaled RMSE is close to that of the 81+ bin, ensuring the model remains effective while providing exhaustive coverage across all class sizes.

Class Size	Model	Scaled Test RMSE	Accuracy
Small (5-30)	XGBoost	**0.740**	**0.826**
Small (5-30)	Enrollment Capacity	1.105	0.705
Small (5-30)	Lasso	0.925	0.705
Small (5-30)	Ridge	0.940	0.723
Small (5-30)	Linear	1.009	0.735
Medium (31-70)	XGBoost	**0.731**	**0.865**
Medium (31-70)	Enrollment Capacity	1.051	0.608
Medium (31-70)	Lasso	0.905	0.715
Medium (31-70)	Ridge	0.901	0.723
Medium (31-70)	Linear	0.907	0.641
Large (81+)	XGBoost	**0.647**	**0.878**
Large (81+)	Enrollment Capacity	0.944	0.739
Large (81+)	Lasso	1.031	0.667
Large (81+)	Ridge	1.008	0.680
Large (81+)	Linear	6.042	0.513

Table 2: Optimal Split Model Comparison

Model	Model Size	Bin Sizes	Test RMSE	Scaled Test RMSE	Accuracy
Enrollment Capacity	Small	5-30	7.201	**1.105**	**0.705**
Enrollment Capacity	Small	5-35	10.136	1.231	0.639
Enrollment Capacity	Small	5-40	10.545	1.212	0.627
Enrollment Capacity	Medium	31-70	15.338	1.051	0.608
Enrollment Capacity	Medium	36-75	16.518	**1.044**	0.679
Enrollment Capacity	Medium	41-80	17.374	1.055	**0.681**
Enrollment Capacity	Large	71+	51.983	**0.904**	**0.753**
Enrollment Capacity	Large	76+	57.198	0.958	0.725
Enrollment Capacity	Large	81+	58.060	0.944	0.739

Table 3: RMSE and Accuracy Enrollment Capacity

Model	Model Size	Bin Sizes	Test RMSE	Scaled Test RMSE	Accuracy	Standard Deviation
XGBoost	Small	5-30	4.827	**0.740**	0.826	6.519
XGBoost	Small	5-35	6.293	0.765	**0.848**	7.895
XGBoost	Small	5-40	6.498	0.747	0.836	8.704
XGBoost	Medium	31-70	10.664	**0.731**	**0.865**	14.587
XGBoost	Medium	36-75	12.141	0.767	0.814	15.788
XGBoost	Medium	41-80	12.572	0.764	0.827	16.464
XGBoost	Large	71+	38.008	0.661	0.847	57.523
XGBoost	Large	76+	41.865	0.701	0.875	59.874
XGBoost	Large	81+	39.788	**0.647**	**0.878**	61.489

Table 4: RMSE Top Models and Baseline

Feature Importance

The feature importance plots (see Figure 20) illustrate how different predictors weigh in on enrollment forecasting for classes of varying sizes—small, medium, and large. Here's a breakdown:

1. Small Class Size Model:
    - The 'Instructor' feature shows the highest importance, suggesting that who teaches the class significantly influences enrollment in small classes. This is followed by the 'Subject-Catalog' and 'Start Time,' indicating that course specifics and scheduling are crucial determinants of enrollment numbers in smaller classes. Features like 'Machine_Learning_LSA' and 'Topics_PersonalDev_LSA' also appear but with lesser importance, highlighting some influence of course content themes on enrollment.
2. Medium Class Size Model:
    - In this model, 'Instructor' remains a significant predictor, albeit with a slight reduction in its relative importance compared to the small class size model. The 'Subject-Catalog' and 'Start Time' are also important, similar to the small class model. Notably, 'Experience_FirstYear_LSA' and 'EngAnalysis_LSA' are prominent LSA features, suggesting that course themes related to first-year experiences and engineering analysis are particularly relevant for medium-sized classes.
3. Large Class Size Model:
    - For large classes, the 'Instructor' feature still ranks highest, but the difference in importance across other features like 'Subject-Catalog' and 'Start Time' is less pronounced than in smaller classes. LSA features such as 'Experience_FirstYear_LSA' and 'Computing_Program_LSA' show relevance,

indicating that specific themes in course content can influence enrollment in larger classes as well.

In all models, the F-score indicates the frequency and relevance of each feature in the model; a higher F-score signifies a stronger influence of the feature on the model's predictions. The consistent appearance of the 'Instructor' across all class sizes underscores the instructor's pivotal role in influencing student enrollment decisions, while the varying importance of LSA features across different class sizes highlights how course content relevance may differ based on class size.

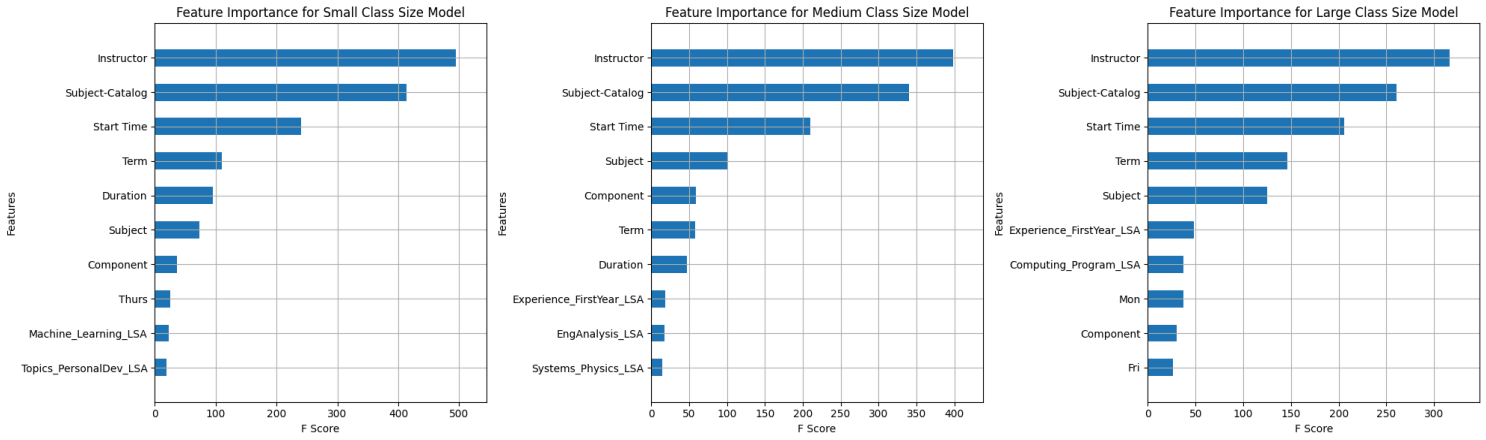

Figure 20: Feature Importance Plots for Final XGBoost Models

Improvements

Our improvements in enrollment forecasting focus significantly on addressing the challenges of model interpretability, particularly for presenting results on the dashboard. XGBoost, while offering state-of-the-art performance, is inherently complex, making it difficult to transparently explain its predictions to non-technical users, who may misinterpret attempts to explain the significance of each input in the model. Based on our background research, we considered other models such as decision trees, linear models, and neural networks. However, these alternative models were either infeasible due to violations of model assumptions (like LSTM), too complex to implement within a 10-week course period, suffered from the same lack of explainability as XGBoost, or performed too poorly to justify over other models. For these reasons and its superior performance in forecasting classes with fewer than 30 students—a critical requirement from our client—we selected XGBoost as our predictive model.

We opted not to include complex explainability features such as SHAP values on the dashboard due to the potential for confusion and misinterpretation. Although these methods are understandable for those with the appropriate technical background, we found that they could lead users without this expertise to draw incorrect conclusions about the model's predictions. We also opted not to include feature importance f-scores on the dashboard as they can be misleading; for example, a highly ranked LSA concept feature might seem influential but actually contribute nothing to a specific prediction if it receives no loading score. Similarly, categories like course catalog combinations not encountered during training default to a particular value in XGBoost, which could misleadingly suggest their importance and, in fact, bias the prediction, reflecting an imputation not truly affecting the outcome.

To aid user interpretation without delving into complex model dynamics, future iterations of this project could incorporate visual benchmarks such as line plots of historical enrollment data by quarter. This approach would provide users with intuitive, comparative insights without requiring a deep understanding of the underlying model.

Another critical area of improvement is data enhancement. Incorporating nuanced data, such as CTEC responses reflecting student perceptions of course desirability and workload, could significantly boost our model's predictive accuracy. Additionally, historical departmental forecasting data provides a more accurate benchmark than mere enrollment capacities. Ensuring data integrity, by addressing issues like duplicate entries and incomplete records, is crucial for maintaining model reliability.

Lastly, alternative modeling approaches could be used to potentially improve results. A clustering model could categorize inputs from the testing set into distinct clusters. The most appropriate forecasting model, trained on clusters of data, would then be selected based on these input clusters, rather than relying on arbitrary size categories. This approach could lead to more finely tuned predictions.

While our current models are tailored to McCormick's data, they may not generalize well to other schools due to distinct factors influencing course enrollment. Developing models specific to each school's unique context would likely yield better results, acknowledging that different schools may have different key drivers of enrollment.

Appendix 5: File Overview

Python files
The Python files are used for data analysis, predictive modeling, and server development. The most important files to modify for future use are:

- XGBoostTest.ipynb: Contains code to test linear and XGBoost models with different bin sizes, allowing for experimentation with model configurations.
- RunServer.py: Links models to the front end, enabling the server to run and serve predictions to the user interface.
- small_model.pkl: Serialized XGBoost model for small classes.
- medium_model.pkl: Serialized XGBoost model for medium classes.
- large_model.pkl: Serialized XGBoost model for large classes.
- mapping_dicts.pkl: Contains mapping dictionaries for categorical variables used in the models.
- svd_model_optimal.pkl: Serialized optimal Singular Value Decomposition (SVD) model for dimensionality reduction as part of LSA.
- tfidf_vectorizer.pkl: Serialized TF-IDF vectorizer for transforming text data into numerical features.

Javascript files
The Javascript files are used for front-end development. The most important files to modify for future use can be found in the src folder. These files are:

- Router.jsx: Manages the application's routing. It defines the paths to different pages and components.
- HomePage.jsx: Displays the main landing page of the application.
- ModelPage.jsx: Displays the page used to explain how our model works.
- FaqPage.jsx: Displays the page that provides answers to frequently asked questions.
- Dashboard.jsx: Collects user inputs, sends them to the server, and displays the resulting forecasts and data visualizations.
- TypeWriter.jsx: Implements the typewriter effect on the HomePage.
- Navbar.jsx: Defines the navigation bar, allowing users to move between different sections of the app easily.
- AreaChart.jsx: Renders an area chart that is displayed on the dashboard.

Appendix 6: Cloud-Based Web Server Instructions

These instructions are designed to guide the implementation of a cloud-based web server. The web server is responsible for taking in user input from our dashboard, generating enrollment forecasts from these inputs, and sending the forecasts back to the dashboard. An API (Application Programming Interface) facilitates this interaction by defining the rules and protocols for communication between the server and the dashboard (see Figure 21). Thus, the server plays a crucial role in our software architecture, linking our dashboard to our predictive models.

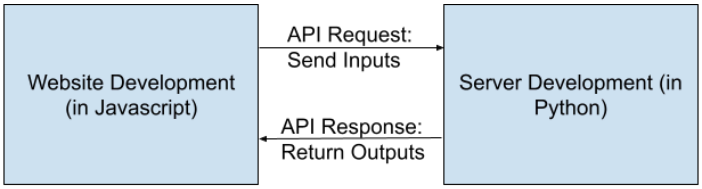

Figure 21: Overview of a Web Server's Interaction With a Website

Local Server Implementation
For demonstration purposes, a local web server was created using the Flask library in Python (see RunServer.py). This server operates only when running on your local machine. It was chosen for its simplicity and lack of server-side costs. However, for continuous operation, a cloud-based solution is recommended.

Cloud-Based Server Implementation
A more robust solution involves using Amazon Web Services (AWS) Lambda Functions. While this incurs some costs, they are considered to be minimal. We recommend requesting assistance from Professor Joe Hummel if you desire a cloud-based server implementation. However, we are providing a general overview of instructions below to create a cloud-based server on AWS, ensuring 24/7 support for the website's forecasting capabilities.

Overview of Instructions
1. Set Up an AWS Account
2. Create and sign in to your AWS account.
3. Search for "Lambda" in the AWS Management Console.
4. Click "Create function," enter a function name, select Python as the runtime, and click "Create function."
5. Upload your Python script and .pkl files in the function code section. Modify the Python script to include a lambda_handler function.
6. Navigate to API Gateway in the AWS Management Console.
7. Click "Create API" and select "HTTP API."
8. Configure routes to link the API Gateway with your Lambda function.
9. Deploy the API to make it publicly accessible.
10. Connect the Dashboard to the API by updating the javascript code in the Dashboard.jsx file of the src folder.
11. Test the workflow by running npm start in the terminal and interacting with the website (you may need to install node package manager and then type npm install to install all relevant dependencies).

12. When everything is working, type npm run build and then firebase deploy. This will deploy all front-end code and javascript files to the cloud (Firebase). You may need to connect a firebase account in the terminal before successful deployment.

Works Cited

Abbasimehr, H., Shabani, M., & Yousefi, M. (2020). An optimized model using LSTM network for demand forecasting. *Computers & industrial engineering*, *143*, 106435

Bedard, K., Kuhn, P. (2008). Where class size matters: Class size and student ratings of instructor effectiveness. *Economics of Education Review*, *Volume 27, Issue 3*, 253-265, https://doi.org/10.1016/j.econedurev.2006.08.007.

Gefen, D., Endicott, J. E., Fresneda, J. E., Miller, J., & Larsen, K. R. (2017). A Guide to Text Analysis with Latent Semantic Analysis in R with Annotated Code: Studying Online Reviews and the Stack Exchange Community. Communications of the Association for Information Systems, 41, pp-pp. https://doi.org/10.17705/1CAIS.04121

Lee, Dianne. 2020. A Classy Affair: Modeling Course Enrollment Prediction. Bachelor's thesis, Harvard College.

Ma, B., Lu, M., Taniguchi, Y., & Konomi, S. (2021, November 27). Investigating course choice motivations in university environments - smart learning environments. SpringerOpen. https://slejournal.springeropen.com/articles/10.1186/s40561-021-00177-4#Sec26

Shaha, A. P., Singamsetti, M. S., Tripathy, B. K., Srivastava, G., Bilal, M., & Nkenyereye, L. (2020). Performance prediction and interpretation of a refuse plastic fuel fired boiler. Ieee Access, 8, 117467-117482.