Independent Study CS499

Art Grichine

December 12, 2015

Introduction

Every semester, The Computer Science department at California State University, Fullerton faces oversubscribed courses. This problem affects both the students and the staff in the department.

For the students of Cal State Fullerton, oversubscribed courses lead to uncertainty during registration, inability to fill required course load, and even delay in graduation. Sometimes additional courses are added or a student is chosen from a wait list. However, for every student that is selected from the waitlist, countless others are unable to take the necessary class. I have personally experienced this problem and have been at the mercy of the department to make room for me.

Currently, course demand is based on the department's intuition. This leads to difficult decisions regarding what courses the department should offer and how many sections to offer per course. Furthermore, these decisions must be made by the department several months before the semester to ensure staff availability thereby leading to difficulties with last minute additions due to demand.

By applying machine learning principles (e .g. regression, neural networks), Dr. Wortman and I believe that it should be possible to predict the demand for each course offered by the department.

Review of Literature

While researching the enrollment forecasting topic, it came as a surprise that there was not much literature covering this topic. Most research in this domain focuses on enrollment in the higher-education system rather than dealing with the individual department. Two articles were found from 1973 and 1975. However, a more recent paper from 2005 by Kraft and Jarvis [2] addresses these papers. Kraft and Jarvis attempt to solve the enrollment forecasting problem through a seat release system which attempts to predict the number of seats needed for a given section. They identify student characteristics that are significant in predicting course enrollment such as first-time vs. transfer student and pass/fail rates of courses. Data from the previous three years is used as a historical estimation and verification for the parameters in the seat release model. The model output reports the minimum, mean, and maximum predicted values for an individual course. The authors argued that their model proved to be accurate and robust.

More recently, a lecture from University of Missouri-Columbia in 2010 by Felts and Ehlert [1] presented an approach which considers a ratio of student demand for a course to the total students in the available population. They use the historical calculated student demand correlation value and apply it to the new student population. The output is the new predicted student demand.

Approaches

There were multiple approaches considered for this project. Two approaches dominated early discussions regarding the project before the model discussed in this paper was chosen. A graphic model offered a macro approach to the data and variables considered in this research. Alternatively, a micro view was considered, studying individual student records and deciding the likely courses that student would take.

Graphic Model

The department of Computer Science at Cal State Fullerton offers it's students a list of all courses needed to graduate with a Bachelor of Science in Computer Science. Along with the list of courses, a prerequisite tree, seen in Figure 1 is also provided demonstrating how courses interact with one another.

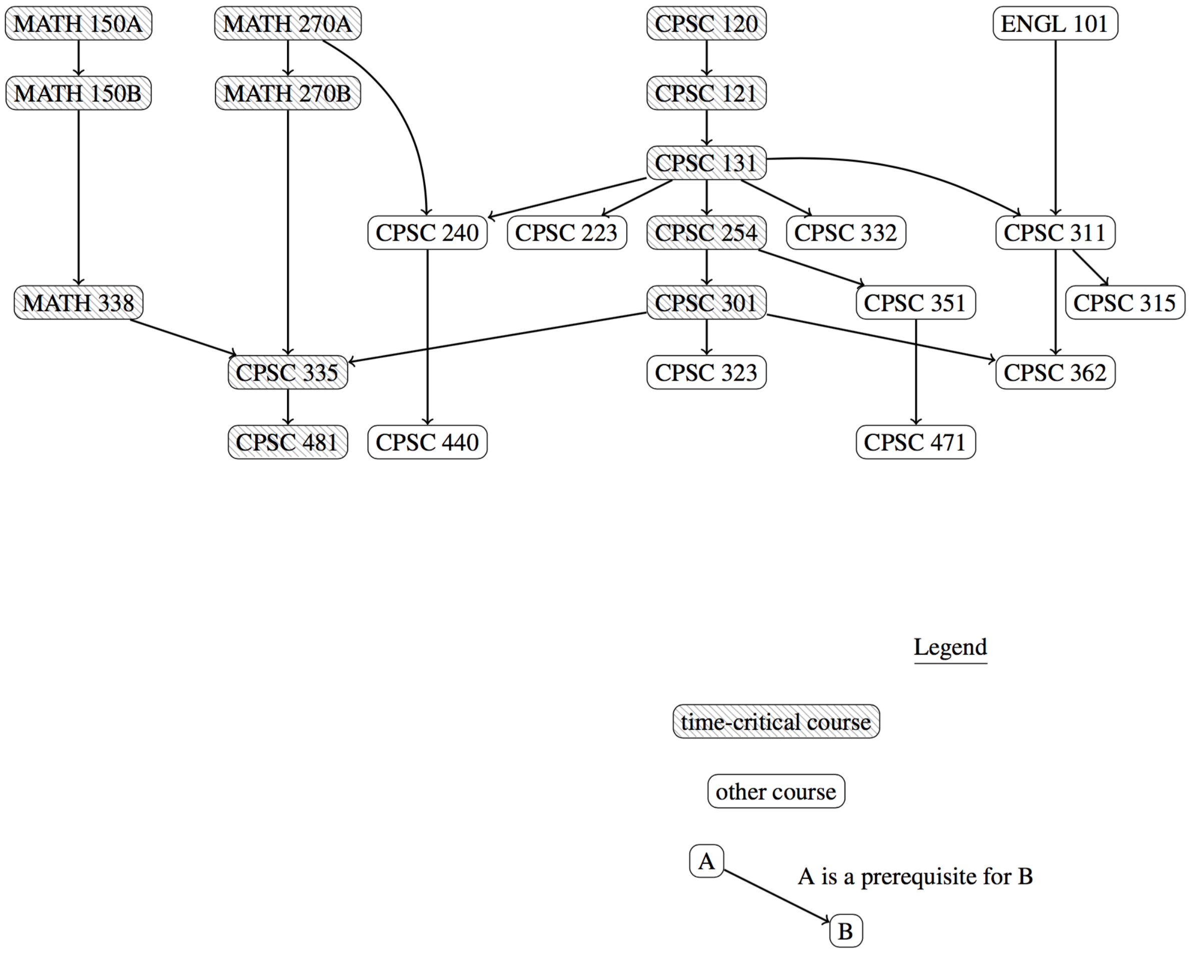


Figure 1: Major Prerequisite Tree

The graphic structure of the tree suggests that perhaps a similar structure can be used to forecast student enrollment. A Bayesian network was considered for the implementation of this project. A Bayesian network uses the conditional dependency between nodes and applies probability to determine how variables (e.g. students) propagate through the model. A Bayesian network's edges must form a directed acyclic graph (DAG) [3]. In the case of the project, nodes would represent the courses a student can take and the progression of students from course to course would be computed based on the probability that a student is to take one course after taking another.

As a more concrete example, lets consider the major prerequisite tree in Figure 1. Here, we see that there is a DAG structure. Therefore, we may be able to apply a Bayesian network. Consider the propagation of students from CPSC 121 to CPSC 131. There are 4 possibilities for a student enrolled in CPSC 121. A student may fail the course thereby having to retake the course. This would mean that the student would propagate from a CPSC 121 node to another CPSC 121 node. Another outcome could be that the student passes the course and moves on to CPSC 131. The model may also experience some attrition due to a student taking a course and not taking any other courses the following semester. These three outcomes can be trained and fit the Bayesian network model.

However, the DAG structure does becomes problematic when we propagate past the CPSC 131 node. This is because in a directed model, once a student moves from the CPSC 131 node to it's child nodes, assuming the student doesn't take all possible child nodes, the student can only propagate downwards in the graph, omitting other courses not connected to the path chosen. This may be more evident in an example. Once again looking at Figure 1, consider that a student passes the CPSC 131 course and moves on to take CPSC 240, CPSC 223, and CPSC 332. In a DAG, once the student propagates to these nodes, they may not go back up to the CPSC 131 node and propagate downward again to take CPSC 254 and CPSC 311. In a Bayesian network, we can compute the probability that the student takes any or all of these courses, however we cannot consider when a student may go back and take other courses they may not have chosen.

To fix the problem we experience in our Bayesian network, we would have to compute all possible sets of courses a student may take as they travel down each path. Here, the problem becomes the ability to process and interpret data with a branching factor that is factorial. Therefore, this option was reconsidered and other options were discussed.

Individual Student Approach

Another model that was considered was to examine each individual student record and determine what courses they were likely to take based on what courses they have taken. The limitations with this approach were not computational. Due to the obvious privacy concerns, student data would need to be void of any student information that could reveal the students identity. Also, student data would need to be gathered which means that forms would need to be submitted to the institutional review board (IRB) to approve research of human subjects. The aggregation of student data and the approval from IRB lead to this approach being reconsidered though it may be worth pursuing when time permits.

Methods

Course enrollment models not using department intuition consider regression analysis to determine course demand. However, regression models may have large error when department policies change regarding curriculum hours and courses required. Also, the student progression through the courses can vary significantly due to individual professor grading. The model used for this study considers historical course grades to predict the amount of students that will pass the course and continue and course demand which is gathered through a questionnaire.

Data

Student data was generated through a Python script called `Dataset_Generator.py`. The dataset generator used the libraries Pandas, Numpy, and MatPlotLib to generate, manipulate, and store data to a .csv file which is then read and used by other programs. This dataset generator was created due to the inability to timely gather needed data for this project. Instead, a Python dictionary of courses was created to hold all undergraduate courses offered by the department of Computer Science at Cal State Fullerton. The main program `Course_Progression.py` calls methods belonging to the `Dataset_Generator.py` program to generate data for its computation. The functions `handle_grades()` and `handle_demand()` are called from the main menu of `Course_Progression.py`. Depending on the user's menu selection, the `Dataset_Generator.py` methods are used to generate the appropriate data. The user also has the option of loading their own data into the program.

Demand Data

The `handle_demand()` function in `Course_Progression.py` asks the user whether they would like to load demand data or if they would like to generate a dataset. Loading data is a trivial action in which the Pandas `read_csv()` method is used and a Pandas data frame containing the file contents is returned to the main menu. The generate demand option calls the `generate_demand()` method from the `Dataset_Generator.py` program. This method uses the default value of `students=1000` and creates a Pandas data frame of current courses and wanted courses for 1000 individual student. This is done to emulate a questionnaire which would ask students which courses they are currently enrolled in and which courses they would like to take next semester.

As the Pandas data frame for student demand is being created, list Python's list comprehension is used to generate individual samples of which courses a student is currently taking and what they would like to take. In the list comprehension, the `generate_samples(students)` function is called to generate individual samples. The `generate_samples(students)` function iterates through the range of the amount of samples needed (in this case `students=1000j` and appends a random sample of courses from the `cs_courses` dictionary into a list of lists (a list of 1000 students each with their own preference). The Numpy `random.choice()` method is used to vary the amounts of courses a student is taking. This method takes a parameter `p` which can be set to a probability distribution for the values chosen. Here, we would like to select between one to six courses for each student (simulating a student course load). However, in most cases it is likely that the student is taking between three to five courses and not one or six courses. Thus, using `np.random.choice()` we can set a higher probability at choosing values of three to five and less at choosing the values of one and six. Once the choice of how many courses to sample from `cs_courses` is returned. The `random.sample()` function is called to choose a random sample of the `cs_courses` dictionary. The samples are then returned to the data frame of current courses and wanted courses in the `generate_demand()` method and the data frame is saved to a demand_data.csv file and returned to the caller.

Grade Data

Just as with demand data, the grade data must be loaded or generated for this program. When the user selects from the main menu the option generate or load student demand data, the `handle_grades()` function is called and the user is prompted to select if they would like to load or generate grade distribution data for their run. If the user chooses to load grades from their own dataset, they will be asked how many grade datasets they would like to load. The user will then enter the names for the .csv file in the working directory one by one until the grades are successfully loaded. However, if the user would like to generate grade distribution data, the user is asked how many datasets would they like to generate and then the users input is passed to the `generate_grades()` method from `Dataset_Generator.py` to generate and load the necessary data frame.

The `generate_grades()` method from `Dataset_Generator.py` program takes one parameter which contains the amount of records needed. It returns a list of data frames which contain grade distributions for each simulated semester. Each individual data frame contains all courses offered by the undergraduate program at the Computer Science Department at Cal State Fullerton. A loop which iterates over the range of records needed calls the `assign_grades()` function which returns a grade distribution for all courses. This result is then appended to the list of data frames and the file is saved to the working directory for record. Once the loop is complete, the list of data frames is returned to the caller.

The `assign_grades()` function creates a random Gaussian (normal) distribution by calling Numpy's `np.random.normal()` method. This method takes two parameters, mean or μ and standard deviation or σ. The mean grade value of $\mu = 75$ and the standard deviation value of $\sigma = 10$ was used to generate the course grades. However, with course grade distribution information this value can be trained to more adequately resemble each individual course distribution. In Figure 2, we see a histogram of the grade distribution generated for an individual course plotted with a probability density function (PDF) line to demonstrate ideal distribution. These grade values are recorded for a given course and then saved. This process is repeated until all courses have grades. Note that each course has its own unique grade distribution as would be seen in practice. A data frame created from the distribution and then returned to the texttgenerate_grades() method. The data frame is saved for record and the loop continues. Once the needed amount of records is created the list of data frames is returned to the caller.

Forecast

To forecast data, the user must select the Forecast Enrollment option in the main menu. At least two files must be loaded before this option is possible. The demand data as well as the grade distribution data must be in loaded using the menu before the forecast enrollment option will work. If this option is selected before the necessary data is loaded, the user is asked to go back and load or generate the necessary data. When the necessary data is

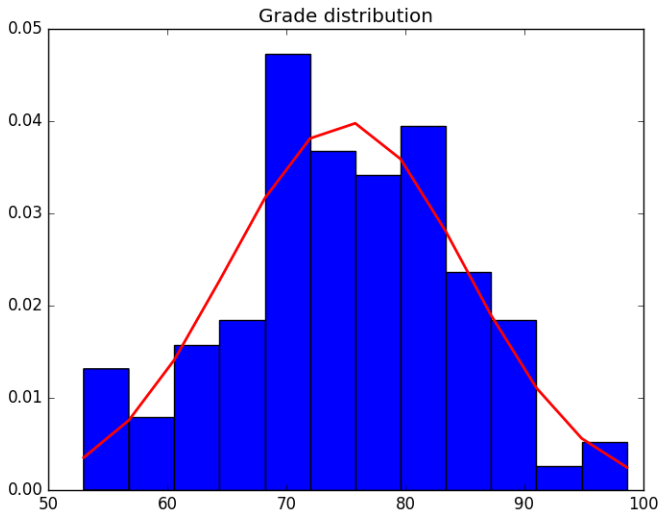


Figure 2: Grade Distribution For A Course

loaded, the `forecast_enrollment()` function is called and the forecast is computed. When the computation is complete, the result is saved to the working directory as forecast.csv and the head and tail of the data frame is shown to the user on the command line. The user is then directed back to the main menu where they may run another simulation or exit the program.

Forecast Enrollment

When the `forecast_enrollement()` function is called by the main menu, it takes the list of data frames containing the grade distributions for each course and a data frame which contains the current student courses and demand. The function then calls the `grade_progression()` function which takes the list of grade distributions to process the data.

In the `grade_progression()` function, we must extract the amount of passing grades for each course as well as the total amount of grades assigned. The list of data frames containing the grade distributions for each simulated semester is enumerated through to extract the sum or passing grades (C- or better) and the sum of total grades assigned to each course. These records are aggregated into one data frame which contains all courses offered as well as the passing, total, and success ratio for each course. The success ratio is computed by

$$\frac{\sum \text{Passing Grades}}{\sum \text{Total Assigned Grades}} = \text{Success Ratio.}$$

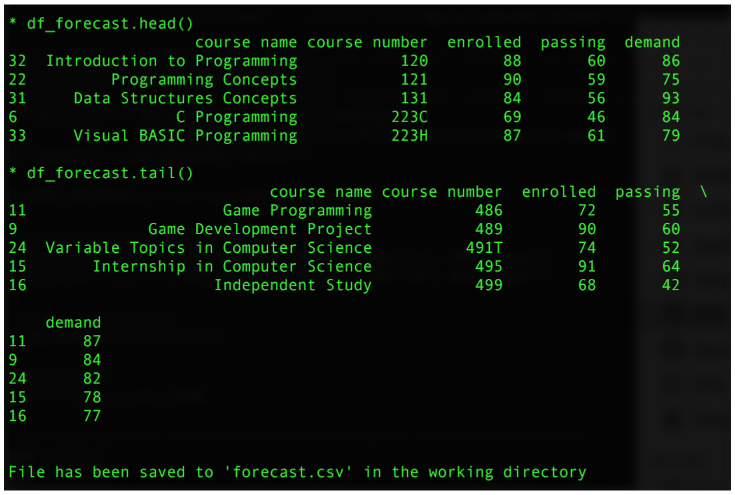
This data frame is then saved to grade_stats.csv and returned to the `forecast_enrollment()` function.

Next, the `forecast_enrollment()` function must also extract the necessary information from the student records. The `student_demand()` function is called and the data frame containing the student records is passed as a parameter. Student demand must go through the records of every individual student generated and count the courses that the students are currently in and the ones that are demanded. The `student_demand()` function counts and aggregates all of the student demand information into a dataframe which contains the course number, course name, amount currently enrolled, and amount requesting course. The data frame is then saved as student_stats.csv and returned to the `forecast_enrollment()` function.

The `forecast_enrollment()` function then creates final data frame which will contain the forecasted contents as well as other important data. The course names and numbers are added to the data frame. The column containing the amount of students currently enrolled in each course is also added. Finally, the amount of students currently enrolled in each course is multiplied by the success ratio in the grade distribution data frame. The column containing the course demand is transferred to the data frame and the head and tail of the contents are displayed to the screen. The data frame contents are then saved to the working directory as forecast.csv and the user is directed back to the main menu.

Results

The program created works as intended. It is robust and did not crash during testing. The results of this study cannot be determined because of the inability to measure which courses are ultimately taken. This simulation would not only need to generate data for the students and grades but would have to also generated enrollment information for the following semester to compute error. However, there is no objective way to compute these values. Therefore, we cannot objectively judge the performance of this approach. An example of the output can be seen in Figure 3 as well as a full run in Figure 4 at the end of this document.



```
* df_forecast.head()
                     course name course number  enrolled  passing  demand
32  Introduction to Programming           120        88       60      86
22         Programming Concepts           121        90       59      75
31      Data Structures Concepts          131        84       56      93
6                  C Programming          223C        69       46      84
33      Visual BASIC Programming          223H        87       61      79

* df_forecast.tail()
                             course name course number  enrolled  passing  \
11                      Game Programming           486        72       55
9               Game Development Project           489        90       60
24  Variable Topics in Computer Science           491T        74       52
15       Internship in Computer Science            495        91       64
16                    Independent Study            499        68       42

    demand
11      87
9       84
24      82
15      78
16      77


File has been saved to 'forecast.csv' in the working directory
```

Figure 3: Full Run

Discussion

The course enrollment forecasting topic affects most departments in higher education. Certainly the larger ones. When researching this topic, little research related to individual department course forecasting was found. This is surprising since the forecasting topic is researched at an institution level to compute the enrollment of the incoming class. The department offers a handbook which contains suggestions on which courses to take and at what time to be done in a timely fashion. However, many students follow their own path which is what makes enrollment forecasting for a department difficult.

There were two difficult parts when approaching this enrollment forecasting problem. First, the lack of research in this subject made it difficult to find an approach to improve on. Instead, it seems that to address this problem one will have to have an original approach which is only limited by the second difficulty, the data. As this study deals with student data there must be institutional review board (IRB) approval to access and use these records. For this study, sudo records were generated however this becomes a problem when approaching the testing phase of the model. Data is key when implementing any type of machine learning algorithm and the quality and quantity of data often dictates the model.

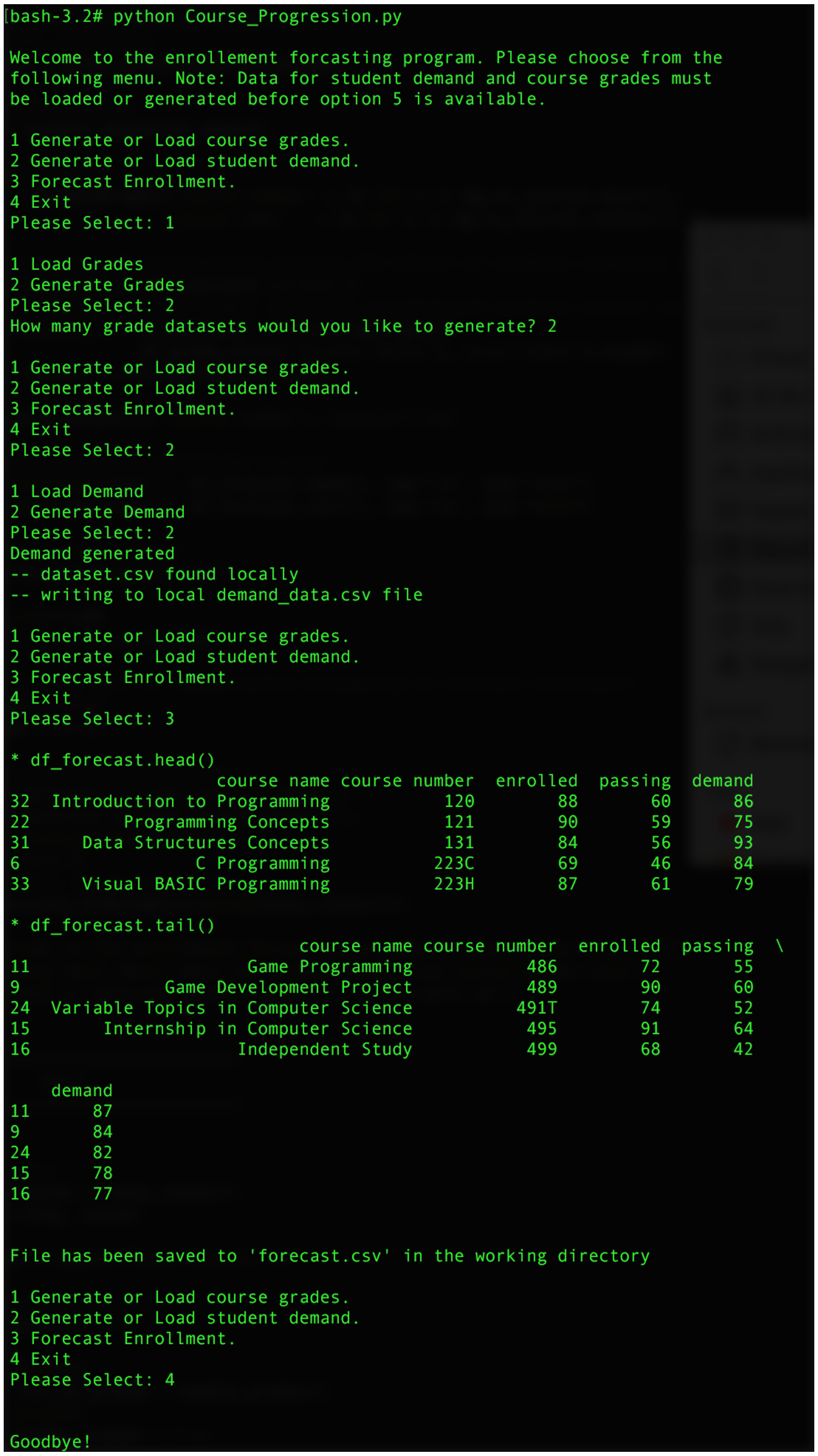
Though department intuition may play a role in the final decision of how many sections and which courses should be offered, there needs to be computational aid through some data analysis be it regression or a more advanced and adaptive model such as a artificial neural network.

References

[1] Ehlert Felts. Prediction model for course demand at MU. 2010.

[2] Christine R Kraft and James P Jarvis. An adaptive model for predicting course enrollment. Technical report, Technical Report, 2005.

[3] Chris J Needham, James R Bradford, Andrew J Bulpitt, and David R Westhead. A primer on learning in bayesian networks for computational biology. *PLoS Comput Biol*, 3(8):e129, 2007.



```
[bash-3.2# python Course_Progression.py

Welcome to the enrollement forcasting program. Please choose from the
following menu. Note: Data for student demand and course grades must
be loaded or generated before option 5 is available.

1 Generate or Load course grades.
2 Generate or Load student demand.
3 Forecast Enrollment.
4 Exit
Please Select: 1

1 Load Grades
2 Generate Grades
Please Select: 2
How many grade datasets would you like to generate? 2

1 Generate or Load course grades.
2 Generate or Load student demand.
3 Forecast Enrollment.
4 Exit
Please Select: 2

1 Load Demand
2 Generate Demand
Please Select: 2
Demand generated
-- dataset.csv found locally
-- writing to local demand_data.csv file

1 Generate or Load course grades.
2 Generate or Load student demand.
3 Forecast Enrollment.
4 Exit
Please Select: 3

* df_forecast.head()
                   course name course number  enrolled  passing  demand
32  Introduction to Programming           120        88       60      86
22         Programming Concepts           121        90       59      75
31     Data Structures Concepts           131        84       56      93
6                 C Programming          223C        69       46      84
33     Visual BASIC Programming          223H        87       61      79

* df_forecast.tail()
                           course name course number  enrolled  passing  \
11                    Game Programming           486        72       55
9             Game Development Project           489        90       60
24  Variable Topics in Computer Science          491T        74       52
15      Internship in Computer Science           495        91       64
16                  Independent Study           499        68       42

    demand
11      87
9       84
24      82
15      78
16      77


File has been saved to 'forecast.csv' in the working directory

1 Generate or Load course grades.
2 Generate or Load student demand.
3 Forecast Enrollment.
4 Exit
Please Select: 4


Goodbye!
```

Figure 4: Full Run