THE RESEARCH AGENDA

Enrollment Predictions with Machine Learning

Michael Soltys, Hung Dang, Ginger Reyes Reilly, and Katharine Soltys

A Machine Learning framework for predicting enrollment is proposed. The framework consists of Amazon Web Services SageMaker together with standard Python tools for data analytics, including Pandas, NumPy, MatPlotLib, and Scikit-Learn. The tools are deployed with Jupyter Notebooks running on AWS SageMaker. Based on three years of enrollment history, a model is built to compute—individually or in batch mode—probabilities of enrollments for given applicants. These probabilities can then be used during the admissions period to target undecided students. The audience for this paper is both SEM practitioners and technical practitioners in the area of data analytics. Through reading this paper, enrollment management professionals will be able to understand what goes into the preparation of a Machine Learning model to help with predicting admission rates. Technical experts, on the other hand, will gain a blueprint for what is required from them.

In the 1971 novel *The Book of Daniel* (Doctorow 1971), E.L. Doctorow writes:

> *Technology is the making of metaphors from the natural world. Flight is the metaphor of air, wheels are the metaphor of water, food is the metaphor of earth. The metaphor of fire is electricity.*

Machine Learning (ML) can also be viewed as a metaphor of the human brain. Indeed, Artificial Neural Networks, the foundation of ML, originally called "Perceptrons" (Minsky and Papert 1969), are modeled on brain neurons and their interconnections.

It is only recently that ML technology has become accessible to "regular" users through a new paradigm of computing: the Cloud (Soltys 2020). Through a partnership with Amazon Web Services (AWS), and in particular an AWS Pilot program in ML for universities, state-of-the-art ML technology can be deployed in order

to help admissions departments with an important and long-standing problem: how best to predict the number of students who will be enrolled in the fall.

According to the Fall 2017 Higher Education Institute report, *American Freshman: National Norms* (Stolzenberg, et al. 2017), 36 percent of freshmen applied to seven or more colleges during the fall 2017 admissions cycle. The California State University system received more than 175,000 unduplicated freshman applications for the fall 2019 admissions cycle and admitted 88 percent of those who applied. CSU Channel Islands (CSUCI) admitted 81 percent of its more than 9,000 freshman applicants.

The ability to predict a university's enrollment and who enrolls is a necessity now more than ever, as high school graduation rates are declining due to lower overall enrollments; students are applying to multiple institutions; competition is growing among universities for enrollment; and uncertainty related to budgets has increased.

For universities, enrollment is necessary for budget and fiscal planning. Enrollment management as an organizational function that emerged as early as the 1960s/1970s due to availability of financial aid and a decline in high school graduates. It was necessary for colleges and universities to examine how students would choose which colleges to attend. Hossler and Bean (1990, 4) define enrollment management as a "set of activities and an organizational framework that enabled colleges and universities to influence student enrollments." Kemerer, Baldridge, and Green (1982) viewed enrollment management as a concept in which it is the intentional approach of ensuring constant flow of students into a university in order to maintain its vitality. By examining the enrollment patterns of its students, a university can have a better handle of what its student body will look like. Computer Science is well suited to analyze enrollment patterns (Rosenberg 2020).

In this paper, the authors show how techniques of ML were deployed in order to aid enrollment management at CSUCI. This case study allowed for CSUCI to use three years of admissions data to determine if ML can predict the likelihood of a student enrolling. The University of New Mexico conducted a similar project in 2018 (Slim, *et al*. 2018). As an open source tool, Jupyter Notebook tools can be deployed by anyone for free; it allows for creating and sharing documents containing live code, equations, visualizations, and narrative text. In particular, it allows for sophisticated statistical analysis before training the ML model.

The audience for this paper is both SEM practitioners and technical practitioners in the area of data analytics. In fact, a collaboration between those two groups is essential. Data analysts need subject matter experts in enrollment in order to build an effective ML model, and enrollment specialists need the technical *savoir faire* of data analysts. As is often in the sciences, collaborations across disciplines yield great insights into open problems.

Python has become the most widely used programming language (Jackson 2014; Tung 2019) due to its status as the *lingua franca* of data analytics as well as its familiarity among many non-computer science experts. In addition, Python is a very high-level language, whose syntax can be understood by those without a deep background in programming (*e.g*., enrollment managers). Through reading this paper, enrollment management professionals will be able to understand what goes into the preparation of a ML model to help with predicting admission rates. Technical experts, on the other hand, will gain a blueprint for what is required from them. The details presented in the paper balance the two perspectives.

ML with AWS SageMaker

This section presents a suite of tools used to predict the number of enrolled students from a given list of applicants by using the AWS SageMaker XGBoost (eXtreme Gradient Boosting) algorithm.

Tools

- **SageMaker:** An AWS managed service for ML, it works with Jupyter Notebook and allows for integration of all the open source software tools listed below.
- **Jupyter Notebook:** An open-source application that allows the creation and sharing of documents that contain live code, equations, visualizations, and narrative text. Its uses include: data cleaning and

transformation, numerical simulation, statistical modeling, data visualization, ML, etc.

- **Pandas:** A library for Python, offering tools for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.
- **Seaborn:** A library for making statistical graphics in Python. It is built on top of MatPlotLib and is closely integrated with Pandas.
- **Scikit-learn:** A ML library for Python. It features various classification, regression and clustering algorithms. This tool can be used instead of AWS SageMaker, in order to make the approach completely cost free.
- **Matplotlib:** A comprehensive library for creating static, animated, and interactive visualizations in Python.
- **NumPy:** A library for Python, adding support for large arrays and matrices, along with high-level math functions to operate on these arrays.

Jupyter Notebook Code

Jupyter Notebook was used on AWS SageMaker, as it is a very convenient environment to clean, analyze and finally feed the data into the SageMaker XGBoost ML algorithm. Highlights of the code are provided in this section in order to give the reader insight on about how the solution is built; the solution is a sequence of short Python codes using the tools listed in Section 2.1.

First, the investigators imported basic Python libraries for Data Analytics:

```
import boto3
import io
import pandas as pd
import json
import matplotlib.pyplot as plt
import numpy as np
```

Next, data from three years of enrollment records, 2018–2020, were imported:

```
s3 = boto3.client('s3')
spread_ sheet = s3.get_object(
  Bucket = "sagemakerbucket",
  Key = "Fall2018-2020.xlsx"
)
data = spread_sheet['Body'].read()
df = pd.read_excel(
  io.BytesIO(data),
  encoding = 'utf-8'
)
```

With the command:

```
df.info()
```

The data types can now be viewed in the Pandas DataFrame (*i.e.*, table). For example, Admit Term is an "object," Last School Attended is a "float64," and SAT Math is an "int64." It is clear that some cleaning of data is needed. For example, as mentioned, the type of Last School Attended Code is given as a float64 when it should be an integer. All three types of "Last School Attended" for the College Entrance Examination Board (CEEB) code and Zip Code are not needed; only one is required. Another example of cleaning is SAT, where it is enough to have "Verbal" and "Math" without the total — which can be obtained from the other two, but more importantly, it correlates heavily with the other two, and strong correlations have a detrimental effect on the quality of a ML model.

Data that are not numeric (int64 or float64) need to be converted into numbers. For example, a long list of major codes as strings can be inspected as follows:

```
major = df['Major Code'].value_counts()
print(major)
```

which outputs the following:

```
NURS - PRE                    4713
UNDECLARED                    4654
PSYC - BA                     3833
BUSN - BS                     3308
BIOLOGY - BS                  2218
COMPSCI - BS                  1277
...
```

The following code is used to convert major codes into integers. In fact, this code is used repeatedly on most columns to convert all values into numbers:

```
major_code = df['Major Code'].unique().tolist()
mapping = dict(zip(major_code, range(len(major_
code))))
df.replace({'Major Code': mapping}, inplace =
True)
major = df['Major Code'].value_counts()
```

The target of the ML algorithm is "Admission Status." The number of cases can be counted using the following code:

```
from collections import Counter
Counter(df['Admission Status'])
```

The following is the result:

```
Counter({
  'Not Admitted': 5757,
  'Enrolled': 2442,
  'Committed, but did not Enroll': 776,
  'Applicant Withdrew': 2059,
  'Admitted, no commitment': 20042
})
```

Investigators delete the rows corresponding to those students who have *not* been admitted—this is done mechanically based on official criteria and not suitable for ML. Once the "Not Admitted" students are deleted, then two categories are created: "Enrolled," and everyone else ("Committed, but did not enroll," "Applicant Withdrew" and "Admitted, no commitment"). This is done with the following Python code:

```
binary = []
for i in df['Admission Status']:
  if i == '2':
    binary.append(1)
  else:
    binary.append(0) df['Admission Status'] =
    binary
```

The ML model will later be used on admitted students at the beginning of their enrollment journey, and it will assign a probability in [0, 1] to each student regarding their enrollment status. While the probability itself is of value in determining the resources spent on recruiting a given student, the investigators will also use a *threshold* to convert this probability into a Boolean "1" (enrolled) and "0" (not enrolled).

In order to ensure that the ML algorithm has coherent and uniform input, "cleaning" operations are performed on all the other attributes as well.

In order to give a sense of the quantities involved, note that the command:

```
Counter(df['Admission Status'])
```

outputs:

```
Counter({1: 2442, 0: 22877})
```

That is, only about 10 percent of all the admitted students become enrolled. Since it is not known *a priori* who those students are, the admissions department needs to spend resources influencing students who are undecided. This will help direct resources to those students most likely to enroll.

Basic Statistics

Once the data residing in Pandas is cleaned, it is necessary to understand trends, patterns, and correlations. For example, CSUCI investigators tested the attributes for correlations with the Python package Seaborn. In general, it is desirable to avoid correlations as much as possible for a more accurate ML model. Figure 1 (on page 15) shows the correlation heat map, where lighter colors mean little correlation, and darker colors mean more correlation (the main diagonal is expected to be dark, as every attribute is correlated perfectly with itself). The numbers in the squares give a numeric value to the correlation, in the set [0, 1].

In CSUCI's case, the heat map showed that there was a higher correlation between parent-education level and whether or not a student was a first-generation college student. This correlation was to be expected. Other statistical analysis with density functions, histograms, and box plots are not included in this paper due to space considerations.

Experimental Results

Once the data has been cleaned and anomalies identified with the statistical analysis, data are run through the ML XGBoost algorithm, where the model is trained on 80 percent of the data, and the remaining 20 percent is used for testing. For the testing data, the target column (which contains the "Admission Status") is deleted, and the ML model is asked to predict, in batch mode, whether a given population will enroll or not. The density graph of the probabilities computed by the ML model on the testing set is displayed in Figure 2, on page 16.

The *confusion matrix*, containing correct guesses on the diagonal, as well as false-positives, negatives off the

FIGURE 1 ➤ Correlations Matrix

diagonal, summarizes the accuracy of the CSUCI experiment. (*See* example in Table 1.) A false-positive occurs when the model predicts that a student will enroll, but the student did not, and a false-negative occurs when the model predicts that the student will not enroll, but the student did.

The confusion matrix is determined by proposing a *threshold*. If a student's probability of enrollment is above the threshold, the student is guessed to enroll; otherwise, the student is guessed not to enroll. Figure 3 depicts the error level based on the threshold (computed in 1/100 increments in [0, 1]). CSUCI investigators discovered that 0.09 (9%) is the threshold that minimizes the error; that is it minimizes the sum of false-positives and false-negatives.

TABLE 1 ➤ Confusion Matrix[1]

	Not Enroll	Enroll
Not Enroll	2,160	704
Enroll	96	148

[1] Threshold = 0.09 (9%)

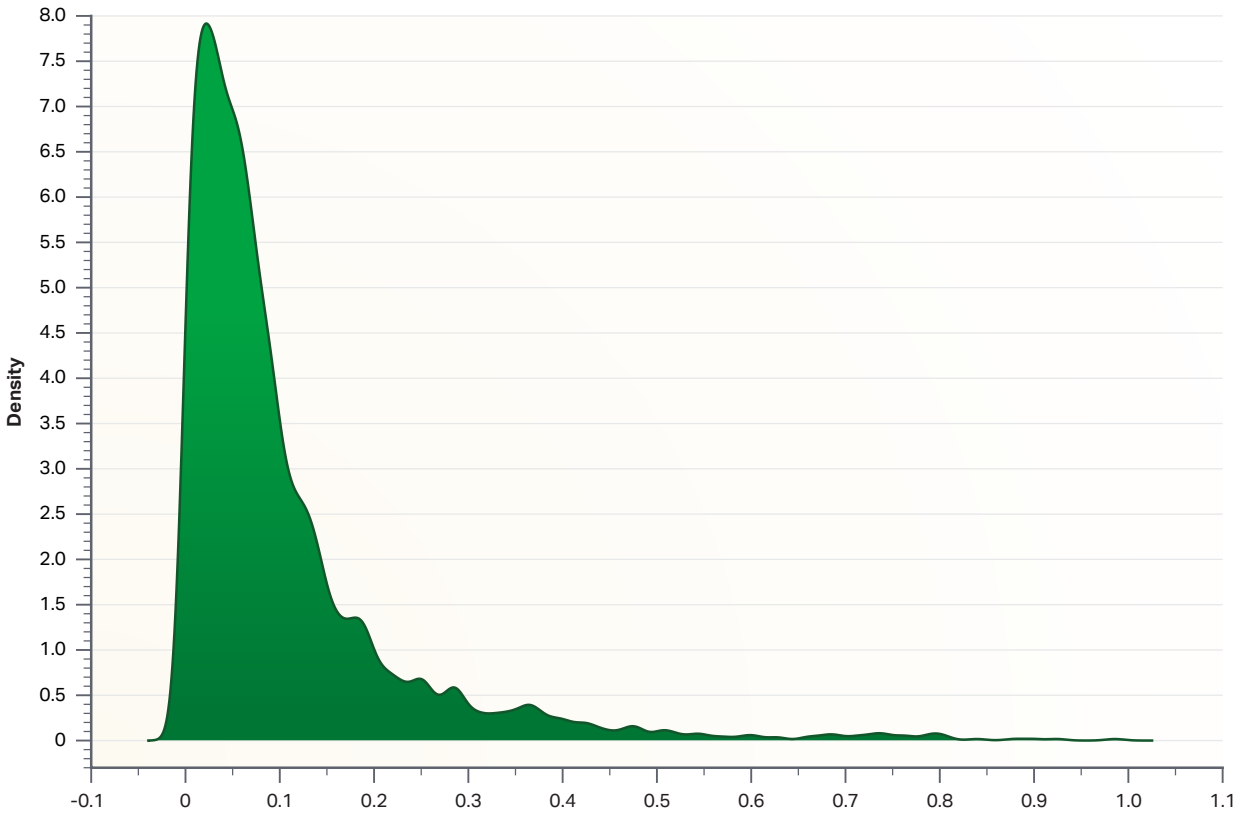

FIGURE 2 ➤ Density Plot of Predictions on Testing Data Set

The data arising from Figure 3 (on page 17) is the principal contribution of the tool, as it provides the following heuristic to CSUCI's admissions team: a prospective student with 0.3 or greater will enroll with high probability (blue line), so there's no need to spend resources on recruiting that student. For those with 0.1 or less, there is a 50-50 chance that the student will enroll. But for values significantly less than 0.1, say 0.03 or less, that prospective student is very likely *not* to enroll, and so once again there's no need to spend resources on that student. A student in the optimal threshold of 0.09 and 0.03 is a good candidate for recruitment as they are on the fence, and hence resources should be on that student.

Conclusion and Future Work

The CSUCI model is not intended to provide a definitive answer to whether a student will enroll or not. As in predicting the weather, there are too many variables to consider, and the answer will be known eventually (time will show).

Yet, just as in the case of predicting the weather, the ML model computed is remarkably accurate: see the confusion matrix in Table 1, and note the error is 25 percent for false-negatives and 39 percent for false-positives. As more years of data become available, including information about student engagement with the university after applying, CSUCI investigators predict that these errors can be reduced significantly.

The CSUCI model offers a useful tool for the admissions department: it provides a strong heuristic to use in order to decide which students to engage in active recruitment. The university will run the ML model on the current application data for fall 2021, which will provide the enrollment probabilities for the admitted cohort in 2021–22. Those probabilities will then inform

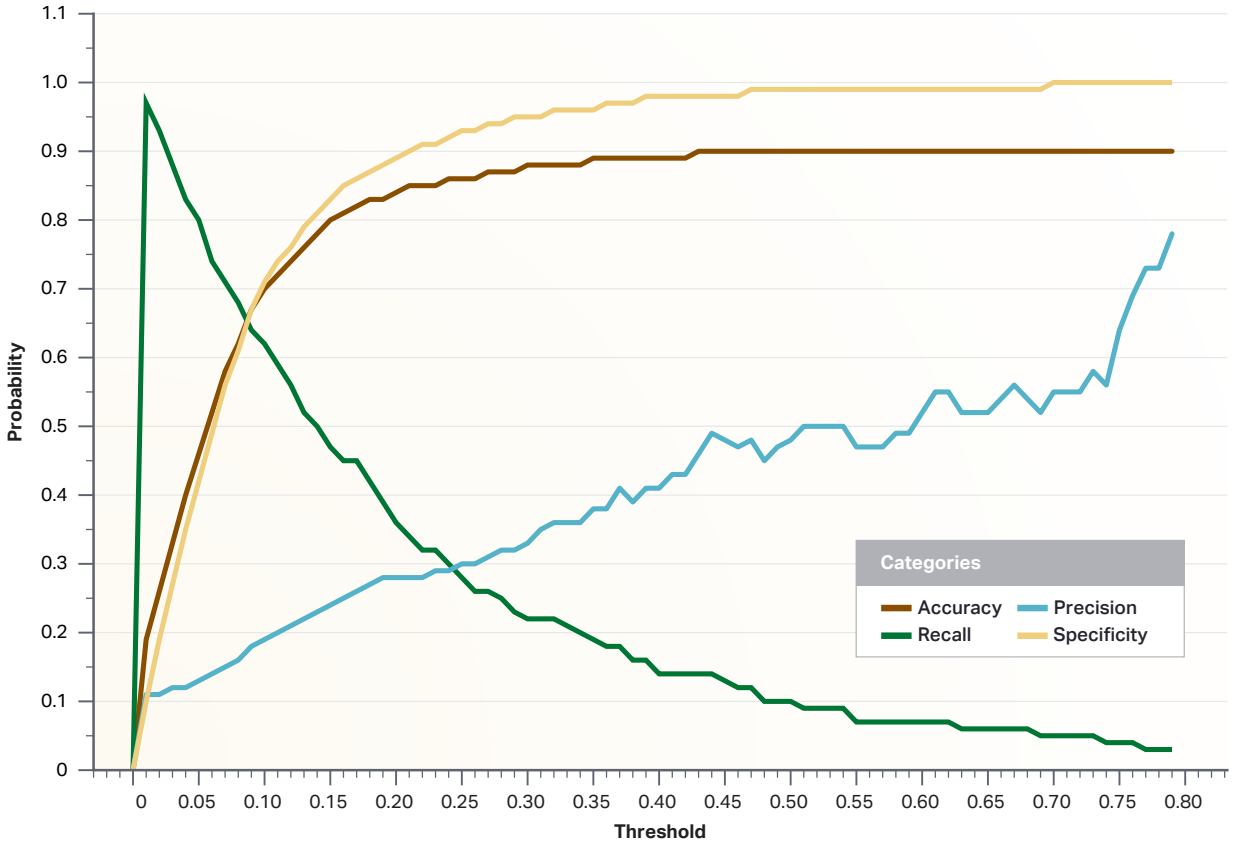

FIGURE 3 ➤ Error Level Relative to Threshold Given, by Accuracy, Recall, Precision and Specificity

the admissions department on how best to invest its limited recruitment resources.

In an era of maximizing limited resources, the ML model will allow CSUCI to:

- target recruitment efforts to admitted students who are undecided;
- follow up with those who are likely to enroll and/or re-confirm their commitment and decision; and
- create a better understanding of the applicant pool, including the characteristics of students deemed to be a good match for CSUCI.

In addition to improving the accuracy of the model, CSUCI plans to apply a time-series analysis by augmenting data with how different student "touchpoints" in the admissions cycle might further improve enrollment probability.

References

Stolzenberg, E. B., K. Eagan, M. C., Aragon, N. M. Cesar-Davis, S. Jacobo, V. Couch, and Cecilia Rios-Aguilar. 2017. *The American Freshman: National Norms Fall 2017.* Los Angeles: UCLA Higher Education Research Institute. Available at: <heri.ucla.edu/monographs/TheAmericanFreshman2017.pdf>.

Kemerer, F. R., J. V. Baldridge, and K. C. Green . 1982. *Strategies for Effective Enrollment Management*. Washington, D.C.: American Association of State Colleges and Universities.

Doctorow, E. L. 1971. *The Book of Daniel*. New York: Random House.

Hossler, D., and J. P. Bean. 1990. *The Strategic Management of College Enrollments*. San Francisco: Joseey-Bass.

Jackson, J. 2014. Python bumps off Java as top learning language. *Computerworld*. July 8. Available at: <computerworld.com/article/2489732/python-bumps-off-java-as-top-learning-language.html>.

Minsky, M., and S. A. Papert. 1969. *Perceptrons*. Cambridge, MA: MIT Press.

Rosenberg, B. 2020. Can algorithms save college admissions? *The Chronicle of Higher Education*. December 15. Available at: <chronicle.com/article/can-algorithms-save-college-admissions>.

Slim, A., D. Hush, T. Ojah, and T. Babbitt. 2018. *Predicting Student Enrollment Based on Student and College Characteristics.* Paper presented at the 11th International Conference on Educational Data Mining (EDM), Jul 16–20.

Soltys, M. 2020. *Cloudifying the Curriculum with AWS*. Camarillo, CA: California State University Channel Islands. Available at: <arxiv.org/pdf/2002.04020.pdf>.

Tung, L. 2019. Tech jobs: Python programming language and AWS skills demand has exploded. *ZDNet*. November 20. Available at: <zdnet.com/article/tech-jobs-python-programming-language-and-aws-skills-demand-has-exploded/>.

About the Authors

Michael Soltys

Michael Soltys, Ph.D., joined California State University Channel Islands (CSUCI) in 2014 as Professor and Chair of Computer Science, Information Technology and Mechatronics Engineering. His vision is to build a world-class department where cutting edge research is put at the service of students and community. His Ph.D. is from the University of Toronto, and he was chair of Computer Science at McMaster University (2001–2014), an Ulam professor at the University of Colorado Boulder (2007–2008), a visiting scholar at UC San Diego (2013), and author of two books and more than 60 research papers. He specializes in algorithms, cybersecurity, and cloud computing.

Hung D. Dang

Hung D. Dang is the Associate Vice President for Enrollment Management at California State University Channel Islands, where he oversees admissions and recruitment, financial aid and scholarships, registration and records, and student data systems. Over the past three decades, he has held similar leadership roles at the University of Hawaii, Manoa; University of Washington, Bothell; and University of Washington, Tacoma. Mr. Dang holds a M.A. in organizational leadership from Gonzaga University and a B.A. in political science from University of Washington.

Ginger Reyes Reilly

Ginger Reyes Reilly, Ed.D., is the Assistant Vice President for Enrollment Management at California State University Channel Islands. She currently oversees the areas of admissions, recruitment, and the registrar's office. Dr. Reyes Reilly has more than 20 years of higher education experience ranging from various student and academic affairs areas. She holds a Ed.D. in organizational leadership from Pepperdine University, a masters in public administration from California State University Northridge, and a B.S. in biology from California State University San Marcos.

Katharine Soltys

Katharine Soltys is a professional communications manager with expertise in enrollment. She is the former assistant director of enrollment communications for California State University, Channel Islands, and previously was director of alumni communication and partnerships at the University of Toronto. Ms. Soltys holds a master's in communication management (MCM) from McMaster/Syracuse University in 2013. Publications include building an online community, the role of social media in strategic community building, and perceptions of foundational knowledge by computer science students, and managing student expectations of university programs.