THE RESEARCH AGENDA

Using Artificial Neural Networks to Predict Matriculation of University Prospects

By David M. Hansen

In recent years we have developed a data analytics pipeline using artificial neural networks to predict prospective student matriculation for university admissions using very limited demographic data. Predictions are generated at the earliest stages of the admissions process and successfully inform recruiting and admissions staff about the likelihood of matriculation. Results over numerous years of matriculation predictions are highly predictive and reliably consistent. We provide a detailed account of data collection, formatting, and transformation processes used, enabling others to replicate the process and results.

In recent years our university has relied on predictive data analytics to predict the likelihood of a prospective student enrolling (*i.e.* matriculating). While the university historically has enrolled between 450 and 650 new undergraduates each year, those students come from a pool of as many as 70,000 "prospects" about whom little is known beyond basic demographic information (*e.g.*, name, address, age, gender, race, high school). With limited resources, it is the daunting task of the recruiting and admissions staff to identify and pursue the small number of "prospects" who ultimately will matriculate.

In 2011 the university asked us to investigate machine learning via artificial neural networks (ANNs) for predicting student matriculation using these limited demographic data (Goodfellow, Bengio and Courville 2016; Kelleher and Kelleher 2019; Witten and Frank 2005). The university had previously engaged a commercial service to compute a numerical ranking of each prospective student using a statistical modeling approach. That process took a number of months to complete and resulted in a predictive ranking for each prospect based on the data available at a single point in time. The predictive analytical pipeline we developed using ANNs outperformed this commercial service at very little cost and enabled near instantaneous regeneration of increasingly accurate predictions throughout the recruiting cycle as additional information (*e.g.*, attendance at a campus preview event) became available.

Over the next five years, the mechanisms implemented proved highly predictive and reliably consistent. Prospective student matriculation predictions have become an integral part of the university's data-driven recruitment and admissions process, contributing to the university's ability to defy broad trends in declining enrollments.

In this paper we provide a detailed description of the data, tools, and straightforward data-processing pipeline we developed together with the results of applying ANNs to early-stage predictive analytics for university matriculation.

Related Work and Context

The use of predictive data analytics for predicting the likelihood that a student will enroll at a university has been an ongoing area of research and application (Amburgey and Yi 2011; Bogard 2013; Chang 2006; DesJardins, *et al*. 2006; Goenner and Pauls 2006; Hasnat 2017; Ledesma 2009; Nandeshwar and Chaudhari 2009; Perry and Rumpf 1984). Factors that influence matriculation and academic success are also well studied (Anders 2012; Chen and Zerquera 2018; Gorard, *et al*. 2019; Moogan 2011; Nurnberg, Schapiro and Zimmerman 2012; Shah, Nair and Bennett 2013; Shaw, *et al*. 2009; Weiler 1996); however, the prospect stage in the admissions process is data-poor, with little information available about a prospect beyond basic demographic data.

Our work has a few distinguishing characteristics: First, we utilize ANNs rather than the more traditional statistical techniques used by most (DesJardins, *et al*. 2006; Goenner and Pauls 2006; Hasnat 2017; Ledesma 2009; Nandeshwar and Chaudhari 2009; Perry and Rumpf 1984). The results we present in Section 4 concur with the general observation made by Chang (2006) that ANNs provide better accuracy than traditional statistical techniques, such as logistical regression, for predicting university matriculation; Goenner and Pauls (2006) use logistic regression and Bayesian model averaging with less accuracy than we have achieved, for example.

Second, we make matriculation predictions at the earliest stage of the admissions process that begin by (1) acquiring basic demographic information about a prospect in the hope of (2) generating an "inquiry" that would lead to (3) an "application" followed by (4) "admission" and, ultimately, (5) matriculation. (An "inquiry" differs from a "prospect" in that an "inquiry" has made contact with the university.) Because of the paucity of data on at the prospect stage, predictive analytics are generally not used until after a putative student has applied. Yet generating a meaningful prediction earlier in the process can be tremendously valuable to university admissions staff in their efforts to find and attract students, focusing limited time and budget on prospects who are most likely to matriculate. Goenner and Pauls (2006) describe similar work to predict matriculation at the slightly later inquiry stage, providing excellent background on the data available and the benefits of providing early-stage predictions; we recommend their work and will not reiterate their useful exposition on the background and utility of generating predictions early in the admissions process.

Most uniquely, we endeavor to provide a sufficiently detailed description of our simple and straightforward process so as to allow others to replicate our success. This contrasts with others, such as Goenner and Pauls (2006), who note that their approach "requires statistical software that is not typically used by institutional researchers" (954). The mechanisms we present can be implemented and deployed by almost anyone with a modicum of computer programming skill and require no background in machine learning or artificial neural networks.

Predictive Analytical Pipeline

The predictive analytical process is straightforward:

- gather and prepare five years of historical data on past prospects (using fewer than five years of historical data was less predictive whereas longer periods might not adapt quickly enough to demographic changes at the university);
- train an ANN to predict matriculation using the historical data; and
- use the ANN to generate a matriculation prediction (0.0–1.0) for each current prospect.

The first two steps can be performed as soon as the fall after new students arrive, providing "ground truth" about who matriculated that year. The last step can be executed repeatedly as new prospects are added to the university's database and as additional information is added about these prospects (*e.g.*, after a campus "preview" event some prospects may have attended).

We generated matriculation predictions for the five years from 2011 through 2015 using datasets of the magnitude described in Table 1. (The process continues to be used by university admissions, but our involvement ended in 2015. One benefit of our success, as evidenced in Table 1, was that the university gradually eliminated its purchase of prospect data from unproductive sources.)

Data Processing and Transformations

Gathering, transforming, and encoding data in preparation for performing predictive analytics is the most time-consuming and complex aspect of the process. The principal source of data is the university's Oracle PeopleSoft database. We programmatically access and query the database, retrieving nine data values for each historical and current prospect:

> *state*, *zip code*, *race*, *gender*, *last school attended*, *data source*, *date data acquired*, *attendance of a "preview" event*, *number of other campus visits*, *matriculated (where "matriculated" is a binary 1/0 value that we train an* ANN *to predict)*

These data comprise nearly all of the information available at the prospect stage of the admissions process. In addition, because distance between home and the university is a significant predictor of matriculation (Chen and Zerquera 2018), we used zip code to calculate the distance between our university and the prospect using a datafile that mapped each zip code to its latitude/longitude. (The haversine formula is used to compute the distance in miles.)

Because any database is likely to be customized for individual academic institutions, it is not useful to present our specific database query; nevertheless, most academic institutions have access to very nearly identical information as well as staff who can assist with data retrieval.

TABLE 1 ➤ Datasets

Year	Prospects	Matriculated	
		n	%
2011	70,537	417	0.59
2012	58,225	407	0.70
2013	55,813	584	1.05
2014	32,748	602	1.84
2015	13,415	559	4.17

Like Goenner and Pauls (2006), we augment the minimal information from our university's database with zip code–based demographic data from the U.S. Census Bureau's *American Community Survey* (ACS)[1] dataset. It is well-documented that finances, urban/suburban demography, ethnic diversity, and education levels are among the attributes relevant to college choice (Anders 2012; Goenner and Pauls 2006; Henrickson 2002; Moogan 2011; Nurnberg, Schapiro and Zimmerman 2012; Shah, Nair and Bennett 2013; Weiler 1996), so we retrieve a subset of ACS data that includes eleven zip code–based attributes:

> *median age*, *total population*, *Caucasian population*, *household income*, *median home value*, *total educated persons*, *number with less than high school*, *high school*, *associate degree*, *bachelor's degree*, *graduate degree*

The various attribute identifiers (*e.g.*, B01002_001E) correspond to particular attributes to be retrieved and returned as an array of values. The web service request in Listing 1 accesses the five-year ACS dataset (*i.e.* acs5) that provides the widest and most accurate coverage and also retrieves data from the 2011 ACS dataset; ACS data are updated periodically, and our production data retrieval code varies the year in the web-service request to match the prospect data being augmented.

We did little to optimize our selection of data attributes, choosing to use as much information as we could

[1] The program used to access the ACS database retrieves and caches zip code data with a web service call such as the Python code fragment in Figure 1 (on page 24). A free "API Key" must be obtained and embedded in the call to use the ACS web service; *see* <census.gov/developers/>.

```
...
if not zipCode in zipCodeCache:
zipCodeCache[zipCode] = urllib.request.urlopen(
    "https://api.census.gov/data/2011/acs/acs5"
    + "?get=B01002_001E ,B01003_001E ,B02001_002E,"
    + "B19013_001E , B25077_001E , B06009_001E,"
    + "B06009_002E , B06009_003E , B06009_004E,"
    + "B06009_005E , B06009_006E"
    + "&for=zip+code+tabulation+area:" + zipCode
    + "&key = ...").read()
...
```

FIGURE 1 ➤ Retrieving U.S. Census American Community Survey Data (Code Fragment)

access easily whether or not a particular attribute was "useful" to the model the ANN implements; we trust that the ANN will learn during training to ignore any input attribute that does not correlate with the output.

At their simplest, ANNs take a "vector" of numeric input values and return one or more numeric output values. So before an ANN can be developed and trained, data must be suitably transformed into a vector of numeric input values. Transforming data that are already numeric (such as "median age" or a zip code) generally requires no additional work. However, not all "numbers" are suitable as ANN inputs (*e.g.*, zip code, as discussed below).

Non-numeric data take a number of forms, and it is important that transformations be done carefully so as to avoid introducing irrelevant artifacts into the data. Three common forms of non-numeric "categorical" data are *ordinal*, *interval*, and *nominal* data (Witten and Frank 2005).

Ordinal categorical data are collections of named values that have an intrinsic ordering. For example, an attribute "course grade" with values {A, B, C, D, F} that can be transformed into numeric values (*e.g.*, {4.0, 3.0, 2.0, 1.0, 0.0}). The data we utilize contain no ordinal categorical data.

Like ordinal data, *interval* data are ordered, but implicitly, by fixed-sized units of measure. Dates, for example, can be treated as a sort of numeric data, and the magnitude of the interval between two dates can be used in a meaningful way. Our data include the date when the prospect entered the system; a prospective student is counted as matriculated if enrolled in classes after a particular future date (*i.e.* the first Friday of the fall semester). Using that future date as an end-point, we compute an interval for other dates that is the distance to that end-point. For example, the "date data acquired" of a prospect for the 2013 academic year who was entered into the database on June 1, 2011, would be represented by the number of days between June 1, 2011, and the matriculation date of September 8, 2013.

Nominal categorical data are non-numeric enumerated values that lack any order among the values (*e.g.*, state, race, last school attended). It is a mistake to map nominal categorical data onto numeric values (as we did above for "grade") because the ANN will try to make sense of the meaningless notion that, for example, *SchoolX* is somehow "less than" or "greater than" *SchoolY*. In fact, some nominal categorical data masquerade as numeric data with the false implication that order and magnitude are relevant. We previously noted that zip code, although represented as a number, should be treated as a nominal category lest the ANN nonsensically treat zip code 10101 as "less than" 10102 and much less than 97132.

One approach to encoding nominal categorical data is to create a binary attribute for each of the possible categorical values, setting one of the attributes to 1 and the rest to 0. (Mapping a nominal category to a binary vector is sometimes described as "one hot" encoding or "binarization.") Instead of a "zip-code" attribute, we might introduce thousands of mutually exclusive attributes, one for each zip code. This approach works for categories with small numbers of values (*e.g.*, gender) but becomes cumbersome and uninformative for many-valued categories (*e.g.*, the hundreds of high schools and zip codes from which prospects are drawn) that would result in input vectors largely comprised of hundreds or even thousands of 0 values.

Our approach to many-valued nominal category attributes such as zip code and "last school attended" is to

convert these attributes into another common type of numeric data: a ratio. Specifically, we compute *two* ratios for each attribute based on historical data. The first is the ratio of how often the value occurs among historical prospects [*e.g.*, (prospects with zip 97132)/(total prospects)]—*relative frequency of occurrence*. The second is the ratio of how often prospects with that same value matriculated [*e.g.*, (matriculated with zip 97132)/(prospects with zip 97132)]–*relative frequency within a subset*. While we could compute just the relative frequency of a particular value (*e.g.*, a specific high school), our intuition is that, while some nearby high schools, for example, will show up quite frequently, a prospect from a high school that we do not see very often has likely taken some intentional action to connect with the university, indicating a higher likelihood of matriculation. These two ratios give the ANN additional information beyond just the relative frequency.

Converting nominal categorical data from the university database into two ratios and augmenting with ACS data and the distance from the prospect's zip code yields an input vector for each prospect that contains 28 input values and 1 output value (whether or not this prospect matriculated), as shown in Figure 2. A file of space-delimited numeric vectors, one per line, comprises the dataset for training an ANN. The training dataset for predicting 2013 matriculation, for example, contained 296,846 historical prospects from 2008 through 2012.

Artificial Neural Network Training and Execution

We use the Fast Artificial Neural Network (FANN) framework (Nissen 2003) for training and predicting with ANNs. FANN is implemented as a C-library that can be downloaded and utilized from a number of programming languages including C, C++, PHP, and the Python code shown here.

FANN requires a simple file format where the first line has three values: (1) the number of lines of data, (2) the number of inputs in each line, and (3) the number of outputs (*i.e.* "296846 28 1" for the 2013 training datafile). Each subsequent line in the data file contains a space-delimited list of numeric input values and one or more output values such as those in Figure 2.

```
1  3.750139611175742E-4 0.3120923509615397
2  0.09259834690199491 0.004093814244132793
3  8.750868888356707E-6 0.8722521321419121
4  0.38995433789954337 1.0 0.0 0.005869670143231168
5  0.2041640375338395 0.15917790597323733
6  0.5883333333333333 0.04802 0.8898375676801332 0.45019
7  657.8 35.4 27999 24521 59243 250200 17611 2058 4504 6194
8  3301 1554 0.0
```

FIGURE 2 ➤ An Example Data Vector

Figure 3 (on page 26) presents a minimal Python program demonstrating how to use FANN to train an ANN. (For the sake of brevity, we show hard-coded filenames and parameters and omit error-handling and more sophisticated processing that includes segregating training data into "train" and "validate" sets.)

FANN, while straightforward and simple to use, has a number of parameters that must be set.

Lines 7–12 in Figure 3 create a three-layer ANN (input, hidden, output) that has 28 inputs and 1 output (as dictated by the input file). Our choice for hidden-layer size is one-half the number of inputs (a not uncommon rule-of-thumb choice; we have experimented using values from one-quarter to twice the number of inputs, with a decrease in accuracy at lower numbers and an increase in training time with higher numbers.)

Lines 14–16 set the functions FANN is to use for training and for the internal nodes of the ANN; again, we have experimented with many of the options FANN offers and have found that the TRAIN_RPROP training algorithm works well for this application, and FANN_ELLIOT works well for hidden and output layer functions. These "functions" and alternatives are described in the FANN documentation.

Finally, line 21 initiates the process of training the ANN on the data, instructing FANN to train for a maxi-

```
import fann.libfann as fann
# Populate a training dataset from a data file
    data = fann.training_data()
    data.read_train_from_file("training.dat")
# Create an ANN
    network = fann.neural_net()
    network.create_standard_array(
        (data.num_input_train_data(),
        # hidden layer size of #inputs/2
        data.num_input_train_data()/2,
        data.num_output_train_data())
    )
# Set training algorithm & hidden/outputfunctions
    network.set_training_algorithm(fann.TRAIN_RPROP)
    network.set_activation_function_hidden(fann.ELLIOT)
    network.set_activation_function_output(fann.ELLIOT)
# Scale the training data to 0.0–1.0
    network.set_input_scaling_params(data, 0.0, 1.0)
    network.scale_train(data)
# Train a network for a maximum of 150 epochs
    network.train_on_data(data, 150, 1, 0.001)
# And save the trained ANN
    network.save("ann.net")
```

FIGURE 3 ➤ Neural Network Training

mum of 150 "epochs" or until the mean-squared-error of the evolving ANN reaches 0.001. (An "epoch" is one pass through the training set.) The mean-squared-error along with other statistics will be printed out after every (*i.e.*, 1—the third parameter) epoch during training. Our experience is that good ANNs quickly evolve after 50 to 150 epochs and become stable with a mean-squared-error ≈ 0.03, showing no improvement with additional epochs. (Our production program is more complex, splitting training data into train and validate sets with subset_train_data(), as well as using train_epoch() and

```
import fann2.libfann as fann
# Create and load an ANN
    network = fann.neural_net()
    network.create_from_file("ann.net")
# Open and read the file of data to predict
    data = fann.training_data()
    data.read_train_from_file("predict.dat")
# Scale the data
    network.scale_train(data)
# Iterate over the data file and print a prediction
    for anInput in data.get_input():
        print(network.run(anInput)[0])
```

FIGURE 4 ➤ Prediction Generation

test_data() instead of train_on_data(), allowing us to stop training more precisely when the mean-squared-error begins to rise on the train or validate data sets.)

Using a typical laptop with an Intel CoreTM i5 processor, this training program takes only minutes to evolve and save a trained ANN on an input file of 296,846 prospects from the period 2008 through 2012, for example. Once an ANN has been trained, predictions are generated by loading the previously trained and saved ANN and using it to predict the output for a data vector.

The code in Figure 4 requires the input data file to have the same format as the training file (the last value in each line can be an ignored "output" of 0.0). "Running" the ANN on a data vector returns an array of predictions; because our ANN has only one output—a matriculation prediction between 0.0 and 1.0—the code prints the first element of the return array.

Generating predictions for each of the 55,815 prospects of 2013, for example, takes seconds.

Assessment of Prediction Results

Using an ANN constructed for each year from 2011 through 2015, we generated (and regenerated) ma-

triculation predictions for every prospective student throughout the recruiting and admissions season prior to the beginning of the academic year. The confusion matrix in Table 2 aggregates the 2011–2015 trained ANNs' predictions showing how often the ANNs predict "yes" and "no" correctly, along with false positives (*i.e.* predicted "yes" when the actual was "no") and false negatives (*i.e.* predicted "no" when the actual was "yes").

Because the ANN generates a prediction between 0.0 and 1.0, the frequency counts in Table 2 are affected by the "cut-off" value used to categorize a prediction as "yes" or "no." While it may seem obvious to choose 0.5 as the cut-off (*i.e.* <0.5 is "no" and ≥0.5 is "yes"), we provide evidence below that a lower cut-off is a better choice; we use a cut-off of ≥0.3 means "yes" for the confusion matrix in Table 2 and the related statistical measures of performance in Table 3.

In this context, "precision" can be thought of as the probability that a (randomly selected) prospect classified as a "yes" actually matriculates while "recall" is the probability that a (randomly selected) matriculating prospect is categorized as a "yes." Because of the large imbalance in this domain between "yes" and "no" categories, "Accuracy" is not a particularly useful measure (Powers 2011); a model would have 96 to 99 percent accuracy simply by predicting "no" for all instances. *F1-Score* (aka *F-Measure*) provides a better measure of a model's performance by taking into account *Precision* and *Recall*, and our ANNs consistently measure ≈ 0.5, where 1.0 predicts perfectly and 0.0 is a complete failure to predict. While we might hope for a higher *F1-Score*, it is notable that we are able to achieve this level of success with only minimal demographic prospect data.

The performance measures in Tables 2 and 3 are useful for assessing ANN performance and validating that the ANNs are indeed predictive, yet they may not seem particularly impressive. However, the predictions are never used in the binary "yes"/"no" fashion suggested by the confusion matrix in Table 2. Instead, the predictions are used as a measure of the *likelihood* of a particular prospect matriculating, an interpretation that has proven to be accurate and quite useful.

TABLE 2 ➤ 2011–2015 Confusion Matrix

		Yes	No
Predicted	Yes	1197	1239
	No	1364	226940

TABLE 3 ➤ ANN Prediction Performance

Year	Accuracy (%)	Precision (%)	Recall (%)	F1-Score
2011	99.5	58.1	44.9	0.51
2012	99.1	36.1	38.1	0.37
2013	99.0	51.0	50.1	0.51
2014	98.3	52.6	47.6	0.50
2015	95.7	48.9	49.6	0.49

Year-to-Year Consistency

Visualizing our results provides perspective about the accuracy, consistency, and usefulness of these predictions. Figure 5 (on page 28) presents the sorted individual matriculation predictions of only the top 10,000 prospects for each of the years 2011 through 2015, demonstrating that the ANN trained for each year is very discriminating with only a very small number of the prospects predicted to have any probability of matriculating; as Table 2 shows, the ANN is very adept at ruling out the vast majority of prospects (*i.e.* true negatives). Moreover, there is a high degree of consistency across the five years, with the ANNs being very accurate at the top of the rankings, where 70 to 80 of the top-ranked 100 prospects matriculated each year.

Figure 6 (on page 29) demonstrates the high accuracy of the ANN's predictions among the 2,000 highest ranked prospects in 2013, grouped into cohorts of 100, showing more than 75 percent of the top 100 prospects matriculating.

Notably, Figure 7 (on page 29) shows that we have identified more than 70 percent of matriculating students within the top-ranked 1,000 prospects and consistently identify 90 percent among the top 3,000.

Figures 5 and 7 also provide some rationale for choosing a cutoff of less than 0.5 for the confusion ma-

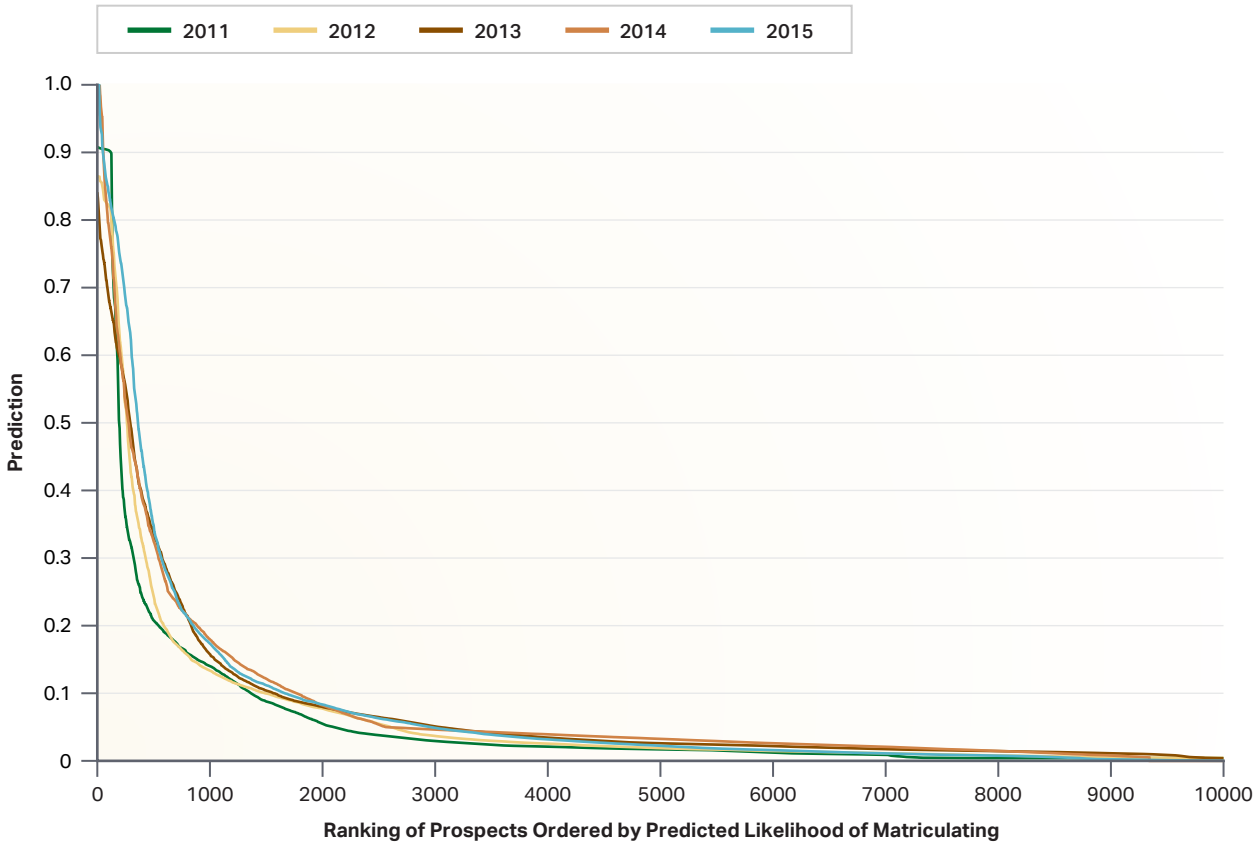

FIGURE 5 ➤ Matriculation Predictions, by Number and Year

trix in Table 2 and the performance measures in Table 3. Figures 5 and 7 show that a fair number of matriculating students are present even up to rank 3,000, where Figure 2 consistently shows predictions of 0.1 and lower.

Figure 8 (on page 30) shows the value of augmenting university data with additional attributes from the ACS by comparing the performance of the ANN used for predicting matriculation for 2013 (*see* Figure 7) against an ANN trained using only university data. While the university data are predictive on their own, the addition of ACS data improves accuracy by as much as 10 to 15 percent.

Practical Prediction Use

Despite the somewhat arbitrary decision to classify predictions in Table 2 (on page 27) as a "yes" or "no" at a cutoff of 0.3, this is not how the predictions of the ANN are used in practice.

One use of the prediction values is to quickly narrow the list of prospects from, for example, more than 50,000 each in 2012 and 2013 to a few thousand. As Figure 7 shows, the highest ranked 3,000 prospects in any year will include ≈ 90–95 percent of matriculating students.

Most important, ANN predictions can be used by admissions staff to effectively allocate their limited time and budget. Prospects who are somewhat less likely to matriculate might receive more recruiting effort than those who are highly likely. We have found that where the prospect falls among the total ordering of all prospects (i.e. their "rank") is as valuable, if not more valuable, than the raw ANN prediction value. Thus both the raw prediction value and the prospect's rank, updated frequently, are available to admissions staff to help guide their assessment of a prospect's likelihood of matriculating.

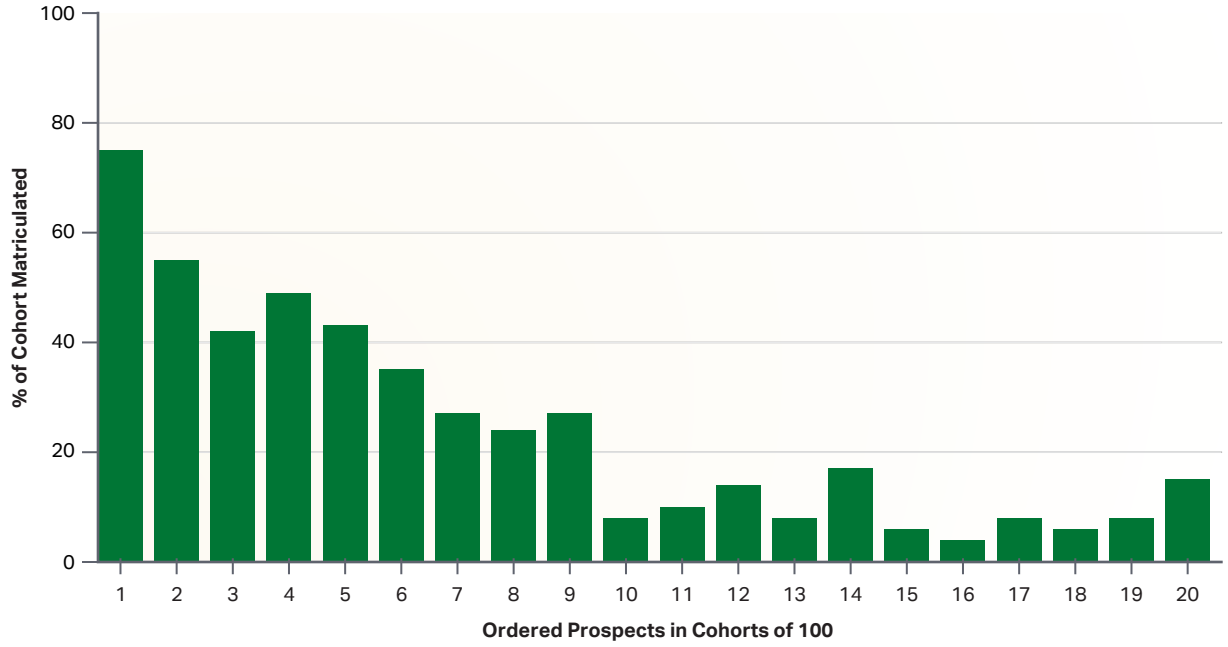

FIGURE 6 ➤ Matriculation Predictions, by Number and Year

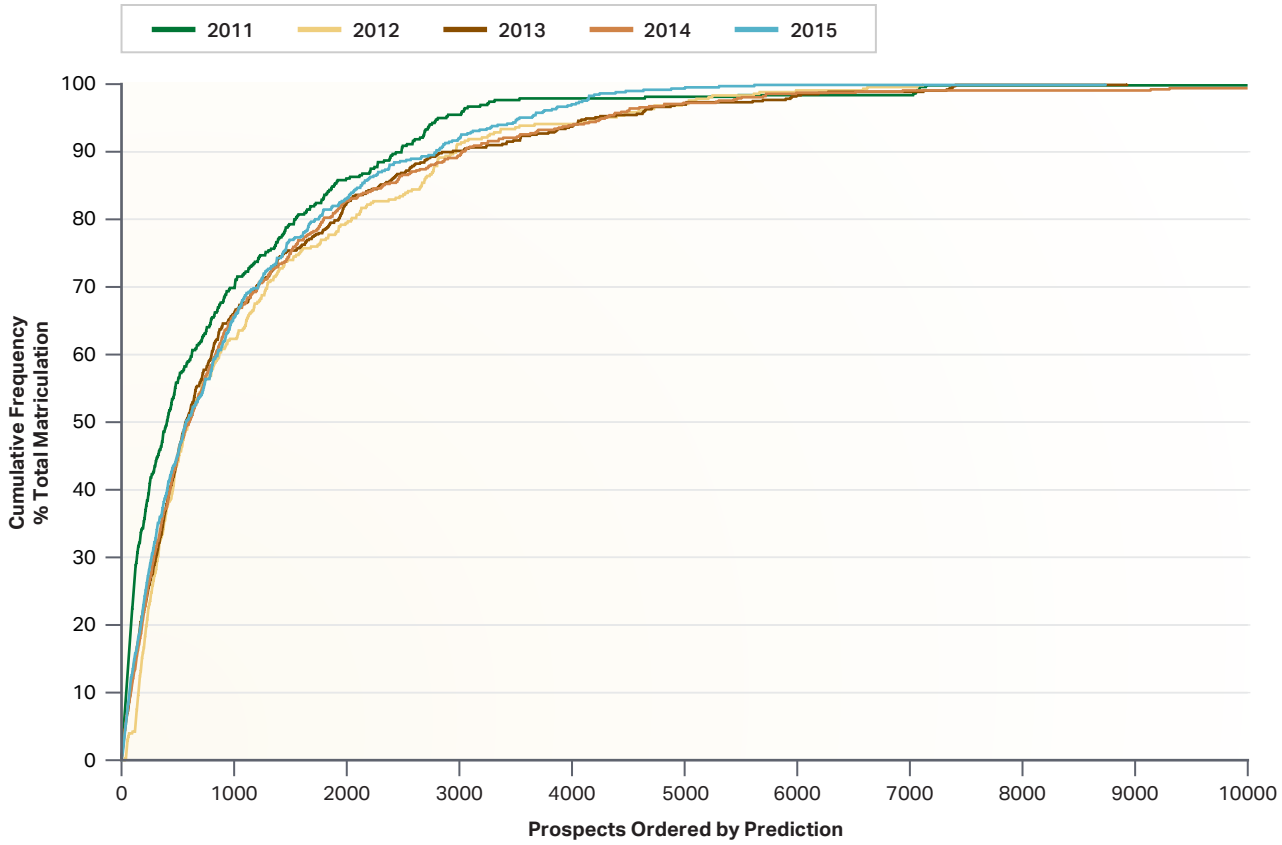

FIGURE 7 ➤ Identifying Matriculating Students

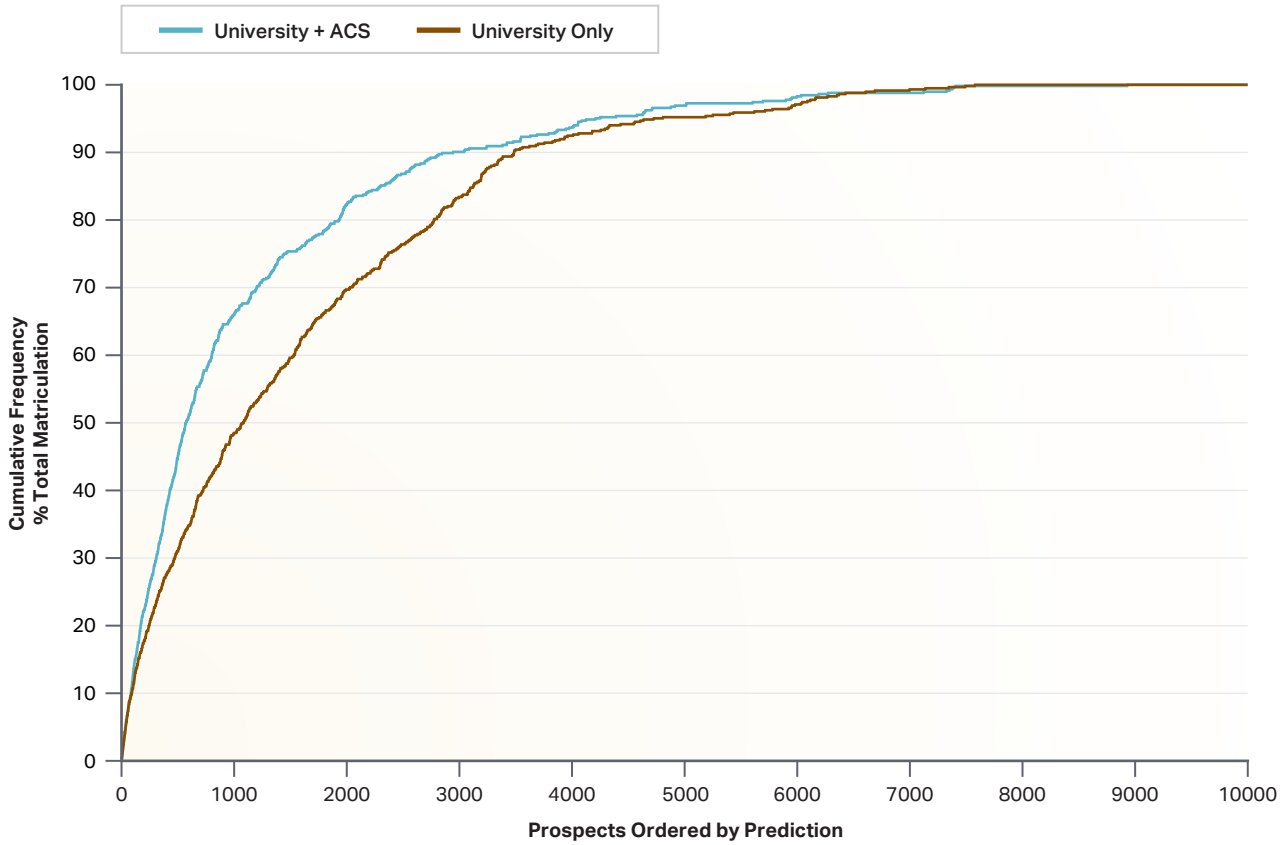

FIGURE 8 ➤ Utility of American Community Survey Data

ANN predictions can also be assessed to expose or validate broader changes to enrollment. The predictions generated in 2014, for example, served to assess the success of a new honors program at the university. The honors program was launched in anticipation that it would draw students to the university who previously were less likely to matriculate; the program was designed to appeal to students with well-above-average standardized test scores, and its unique curriculum and pedagogy were expected to draw students from a broader geographic area. Evaluating the predictions for students who enrolled in the honors program showed that the program had indeed attracted a cohort of students the ANN predicted was less likely to matriculate. Prospects who enrolled in the honors program had a mean prediction of 0.28 and mean rank of 6,387 compared to a mean prediction of 0.41 and mean rank of 1,131 among all who matriculated in 2014.

These mechanisms have also been used to develop a model that predicts matriculation among prospects who have completed an application. Developing an "applied" ANN is a matter of augmenting the data on prospects with additional information known about applicants, including:

> *date applied, how applied (e.g., web, Common App), class standing, GPA, standardized test scores, intended major(s), parents' adjusted gross income*

Table 4 and especially 5 show how much more accurate an ANN predicting matriculation for applicants is at that later stage of the admissions process.

Figure 9 shows the performance and accuracy of an "applied" ANN for 2013 with the ANN very effectively identifying the 584 of ≈ 2,200 applicants who would ultimately matriculate.

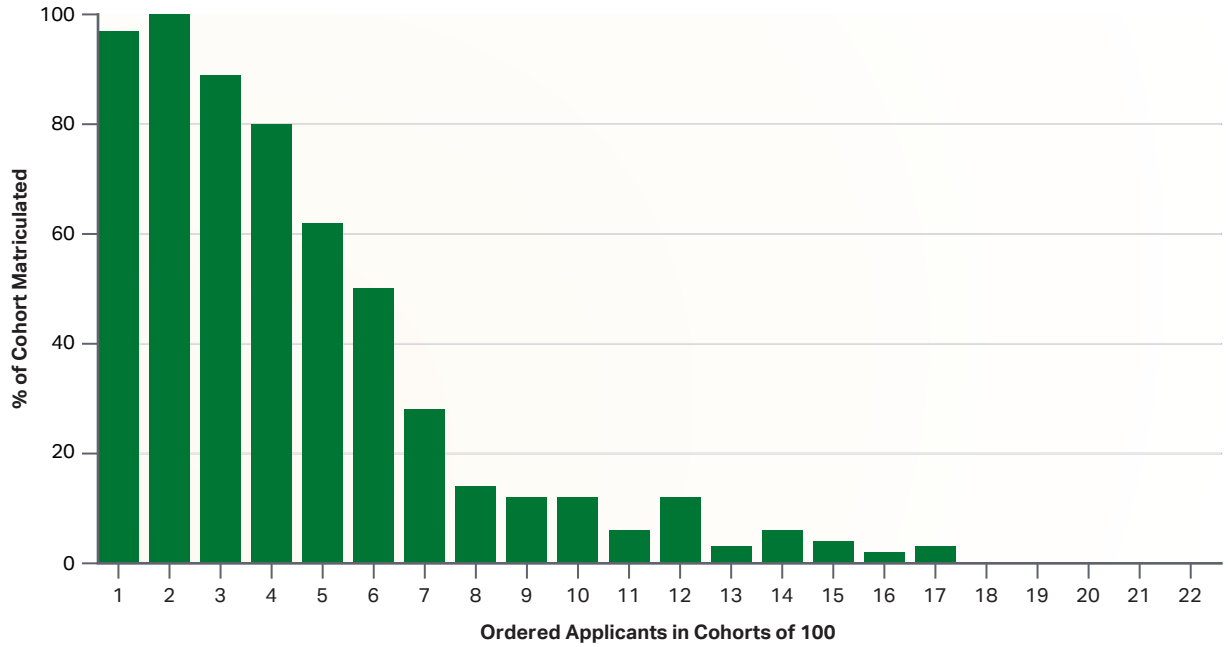

FIGURE 9 ➤ Applied ANN Prediction Performance

Conclusion

Predictive analytics have become an integral part of our university's recruiting and admissions processes. We have presented the results of using ANNs to generate matriculation predictions early in the university admissions process. The tools and techniques presented are straightforward to implement and use, requiring no expertise in machine learning or Artificial Neural Networks.

We began using ANNs in 2012 by developing the process and mechanisms presented. We tested, refined, and assessed the accuracy by initially generating predictions for 2011—a year for which we already had "ground truth"—giving us some confidence that the predictions generated for 2012 would be useful. Each following year we used "ground truth" about who matriculated, confirming that the process and predictions were consistently useful.

We continue to explore the use of ANNs in related areas, such as predicting retention (Astin 1997), that contribute to the success of students and the university.

TABLE 4 ➤ Applied Confusion Matrix

		Yes	No
Predicted	Yes	1461	102
	No	119	1549

TABLE 5 ➤ Applied Accuracy by Cohort

Accuracy (%)	Precision (%)	Recall (%)	F1-Score
90.1	81.9	79.5	0.81

References

Amburgey, W.O.D., and J. C. Yi. 2011. Using business intelligence in college admissions: A strategic approach. *International Journal of Business Intelligence Research*. 2(1): 1–15.

Anders, J. 2012. The link between household income, university applications, and university attendance. *Fiscal Studies*. 33(2): 185–210.

Astin, A. 1997. How "good" is your institution's retention rate? *Research in Higher Education*. 38(6): 647–58.

Bogard, M. A. 2013. A data driven analytic strategy for increasing yield and retention at Western Kentucky Uni-

versity using SAS Enterprise BI and SAS Enterprise Miner. In *SAS Global Forum.* Available at: <wku.edu/instres/documents/analytic_strategy.pdf>.

Chang, L. 2006. Applying data mining to predict college admissions yield: A case study. *New Directions for Institutional Research.* 2006(131): 53–68.

Chen, J., and D. Zerquera. 2018. Leaving or staying home: Predicting where students attend college. *Education and Urban Society*. 50(4): 376–399.

DesJardins, S.L., D. A. Ahlburg, and B. P. McCall. 2006. An integrated model of application, admission, enrollment, and financial aid. *Journal of Higher Education*. 77(3).

Goenner, C., and K. Pauls. 2006. A predictive model of inquiry to enrollment. *Research in Higher Education*. 47(8): 935–956.

Goodfellow, I., Y. Bengio, and A. Courville. 2016. Deep learning. *Adaptive Computation and Machine Learning Series.* Cambridge: MIT Press.

Gorard, S., V. Boliver, N. Siddiqui, and P. Banerjee. 2019. Which are the most suitable contextual indicators for use in widening participation to higher education? *Research Papers in Education*. 34(1): 99–129.

Hasnat, S. N. 2017. Analyzing the fundamental aspects and developing a forecasting model to enhance the student admission and enrollment system of msom program. Master's thesis, University of Arkansas.

Henrickson, L. 2002. Old wine in a new wineskin: College choice, college access using agent-based modeling. *Social Science Computer Review*. 20(4): 400–419.

Kelleher, A., and A. Kelleher. 2019. *Machine Learning in Production: Developing and Optimizing Data Science Workflows and Applications.* Addison-Wesley Data & Analytics Series, Pearson Education.

Ledesma, R. 2009. Predictive modeling of enrollment yield for a small private college. *Atlantic Economic Journal*. 37(3): 323–324.

Moogan, Y. 2011. Can a higher education institution's marketing strategy improve the student-institution match? *International Journal of Educational Management*. 25(6): 570–589.

Nandeshwar, A., and S. Chaudhari. 2009. Enrollment prediction models using data mining. *ResearchGate*. April. Available at: <researchgate.net/publication/263046466_Enrollment_Prediction_Models_Using_Data_Mining>.

Nissen, S. 2003. Implementation of a Fast Artificial Neural Network library (FANN). *ResearchGate*. December. Available at: <researchgate.net/publication/228878360_Implementation_of_a_Fast_Artificial_Neural_Network_library_FANN>.

Nurnberg, P., M. Schapiro, and D. Zimmerman. 2012. Students choosing colleges: Understanding the matriculation decision at a highly selective private institution. *Economics of Education Review*. 31(1).

Perry, R., and D. Rumpf. 1984. Predicting the likelihood of matriculation for college applicants. *Research in Higher Education*. 21(3): 317–328.

Powers, D. 2011. Evaluation: From precision, recall and f-measure to ROC, informedness, markedness, and correlation. *Journal of Machine Learning Technologies*. 2(1): 37–63.

Shah, M., C. Nair, and L. Bennett. 2013. Factors influencing student choice to study at private higher education institutions. *Quality Assurance in Education.* 21(4): 402–416.

Shaw, E. J., J. L. Kobrin, S. F. Packman, and A. E. Schmidt. 2009. Describing students involved in the search phase of the college choice process: A cluster analysis study. *Journal of Advanced Academics*. 20(4): 662–700.

U.S. Census Bureau. *American Community Survey* (ACS) dataset. Washington, D.C.: Author. Available at: <census.gov/programs-surveys/acs>.

Weiler, W. C. 1996. Factors influencing the matriculation choices of high ability students. *Economics of Education Review*. 15(1): 23–36.

Witten, I. H., and E. Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd edition. Morgan Kaufmann Series in Data Management Systems. Boston: Morgan Kaufmann.

About the Author

David M. Hansen

David M. Hansen received his Ph.D. in Computer Science and Engineering in 1995 from the Oregon Graduate Institute of Science & Technology. Dr. Hansen is currently an Associate Professor of Computer Science and Information Systems at George Fox University where he teaches a wide range of undergraduate courses that span the gamut from AI to Architecture & Assembly Language. Recent research has included Computer Science pedagogy, applied machine learning, and partition- tolerant distributed information systems.