Performance comparison of artificial neural network and logistic regression model for predicting chance of admit for students based on applicant data

Weiyou Li

Abstract

Artificial neural networks (ANN) and logistic regression (LR) are two widely used classification models. This study compares their predictive performance on a small-scale (400) student-admissions dataset, filling a gap in existing studies on limited-data situations. We systematically test ANN hyperparameters such as decision threshold, hidden-layer structure, number of epochs, and batch size. We then compare LR through feature-correlation analysis. Our results indicate LR to be more stable on this small data set, with both models performing similarly on mean and best-case accuracy (ACC) and area under the ROC curve (AUC), while LR showing marginally higher average metrics for all trials.

Keywords: Artificial neural network; Logistic regression; Model comparison;

1 Introduction

Predicting graduate school admissions outcomes has become increasingly essential for both applicants and universities. Admissions decisions often rely on several factors, such as GRE and TOEFL scores, undergraduate GPA, and research experience. While artificial neural networks (ANN) and logistic regression (LR) are both widely used classification tasks, there is limited research comparing their performance when sample sizes are small.

Although ANN models are known for their flexibility for non-linear relationships, they often require large datasets to perform reliably. Conversely, LR is more stable and interpretable, especially when strong linear relationships exist between predictors and outcomes.

This study addresses that gap: we compare ANN and LR on a publicly available dataset of 400 student applicants to UCLA graduate programs. We explore how ANN performs under varying hyperparameters (e.g., threshold, hidden-layer structure, batch size, epochs) and assess LR with correlation checks and standard assumptions. We evaluate both models using accuracy (ACC) and area under the ROC curve (AUC), analyzing performance and stability through multiple trials.

The remainder of the paper is organized as follows. Section 2 describes the data, preprocessing steps, and modeling framework. Section 3 presents the performance results, and Section 4 interprets the findings. Finally, Section 5 summarizes the main conclusions.

2 Methods

2.1 Dataset and Preprocessing

We use a publicly available dataset [2] of 400 candidates for UCLA graduate admission from an Indian perspective. The dataset contains eight parameters which are considered important during the application, namely :

1. GRE Scores (out of 340)
2. TOEFL Scores (out of 120)
3. University Rating (out of 5)
4. Statement of Purpose (out of 5)
5. Letter of Recommendation Strength (out of 5)
6. Undergraduate GPA (out of 10)
7. Research Experience (either 0 or 1)
8. Chance of Admit (ranging from 0 to 1)

We use only "Chance of admit" as the target variable and treat the other seven variables as features since we are only interested in how likely a student will be admitted. The data is split in a way so that 80% is used for training and 20% for validation. We then standardize the training features via normal standardization and apply the same scaling to the validation features.

Finally, we build two models to compare:

- An artificial neural network (ANN)
- A logistic regression (LR)

2.2 Logistic Regression

2.2.1 Correlation and Assumption

Before fitting a logistic regression model to the data, we should satisfy as many of the following assumptions for LR as possible [18, 21]:

1. **Linearity**: A straight line should exist between the independent and dependent variables' log odds.
2. **Datasize**: An adequate number of events per independent variable to avoid an overfit model, with commonly recommended minimum "rules of thumb" ranging from 10 to 20 events per covariate.

3. **Independence**: The observations should stand alone from one another. This implies that the response variable's value for a given observation shouldn't be tied to its value for any other observation.

4. **No multicollinearity**: There shouldn't be much correlation between the independent variables.

5. **Outliers**: Outliers represent data points that significantly differentiate themselves from the remainder of the data and may profoundly affect the analysis findings. If outliers exist, they must be located and either eliminated or researched.

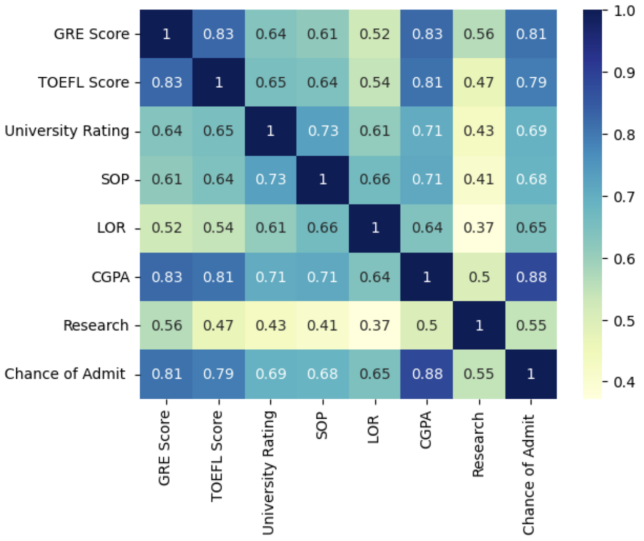

Figure 1: Heat map of correlation coefficient between each factors

To evaluate the linearity assumption, we created a heat map of correlation which is shown in figure 1. The smallest correlation coefficient between any feature and the target is about 0.55, which indicates a moderate correlation. The rest of feature variables result strong or very strong correlations to the target variable. Hence, the first assumption is satisfied. Now we shall check for the data size. According to Motrenko's study[13], the minimum data size would be

$$\frac{(10 * \textit{Number of predictors})}{\textit{Event rate}}$$

where the event rate is calculated by

$$\frac{\text{Number of positive cases}}{\text{Total number of cases}}$$

Figure 11 presents the confusion matrix used to estimate the event rate. Since there are 29 true positive and 2 false negative, the total number of positive cases is

$$29 + 2 = 31,$$

yielding an event rate of

$$\frac{31}{100} = 31\%.$$

Given 7 feature variables, the recommended minimum sample size is

$$\frac{10 \times 7}{0.31} \approx 226.$$

Our dataset contains 400 observations, thus satisfying this requirement. In addition, each observation is independent, meeting the independence assumption.

Outliers were detected using the interquartile range (IQR) method [24]. Following Schwertman's result [17], any observation below

$$Q_1 - 1.5\,\text{IQR}$$

or above

$$Q_3 + 1.5\,\text{IQR}$$

was classified as an outlier. The code is provided in Appendix A.2, and Section 3.1 presents the outlier count for each variable. We identified only three outliers among the 400 data and extracted their corresponding rows and columns (see Figure 3). To determine whether any of these outliers have influence on LR, we computed Cook's distance (code in Appendix A.4) and found none of them are influential. Therefore, the outlier assumption is satisfied and we keep all the outliers since they are not influential. Although Figure 1 suggests some multicollinearity, four of the five key LR assumptions hold. This means a logistic regression model is reasonable, though it does not fit the data perfectly.

Finally, while feature selection often benefits large datasets, Kuncheva's team warn that it can mislead when samples are very small. [11] Hence, we chose not to apply feature selection here.

2.2.2 AUC and ACC Curves

We implemented logistic regression model based on the sklearn built-in module, and fit it with the training set of feature and target variables, respectively. On the validation set, we generated predicted classes and probability scores, which were then used to compute accuracy and AUC. The ROC curve is presented in Figure 4, with the corresponding ACC and AUC values listed below.

```
Validation Accuracy: 0.9400
Validation AUC: 0.9762
```

2.3 Artificial Neural Network

2.3.1 Initial Threshold Value

The artificial neural network, unlike logistic regression, is a classification. Since each applicant is either admitted or rejected, we convert the "Chance of admit" score into a binary label: scores at or above a chosen cutoff are labeled 1 (accepted), and scores below that cutoff are labeled 0 (rejected). Unfortunately, the original dataset lacks information for programs and year-specific admission rates for UCLA graduate programs, so we take the overall admission rates for all programs from 2013 to 2018 into consideration, which can be found here:[7] These rates consistently average about 25% (±1%), so we may assume only the top 25 percent of students is considered to be accepted at this stage.

Figure 2 shows the distribution of accept rate for the original dataset [2]:

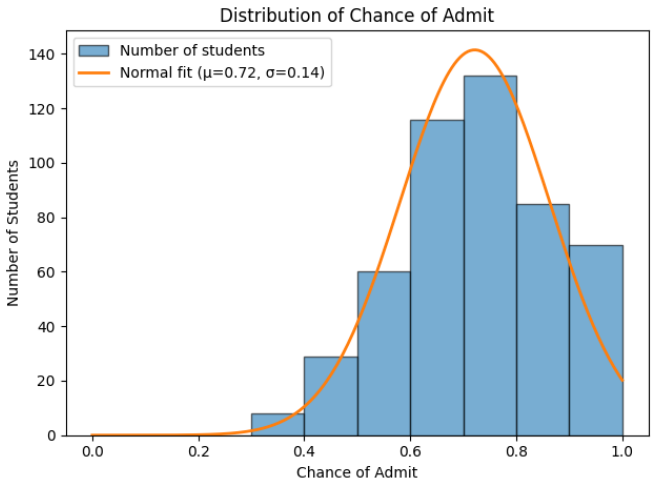

Figure 2: Distribution of Chance of Admit with mean = 0.72

Based on the graph, we can see that the distribution is slightly skewed to the left with a mean of 0.72. Since only the top 25% is considered to be accepted, we can calculate the 75th percentile to get an initial threshold value. By using python, we have the top 25% is 0.82. For simplicity, we set the initial threshold value to be 0.8.

2.3.2 Hidden Layers

Although there are eight features in the original data set, the application ID (serial number) does not influence admission probability. Therefore, the size of the input layer is $8 - 1 = 7$. Next, we need to decide on the number of hidden layers and the neurons in each. A research in 2020 found that too many hidden layers would slow down the training [23]. Karsoliya recommends two or three hidden layers for arbitrary accuracy, warning that a fourth layer may introduce issues [9]. Panchal shows that if accuracy is the priority, multiple hidden layers are beneficial, but if training time is critical, a single hidden layer is preferable [14].

Beyond the number of hidden layers, how many neurons in each hidden layers is also a problem. Karsoliya's study [9] provides a rule of thumb for determining the size of hidden layers in neural networks:

- The number of neurons in the hidden layers should fall between the size of the input and output layers.
- Keep each hidden layer's size below twice the number of input neurons. For example, using approximately two-thirds of the input size for the hidden layer is often considered a reasonable starting point.

Although there is limited evidence to support that gradually reducing the number of neurons in each hidden layer improves efficiency, we followed this approach to reduce the total number of parameters deal with. To examine the impact of different structures, we built three models with varying numbers of hidden layers and neurons, while keeping all other factors the same.

- **Model 1**: Two hidden layers with 4 and 2 neurons.
- **Model 2**: Three hidden layers with 5, 4, and 3 neurons.
- **Model 3**: Three hidden layers with 4, 3, and 2 neurons.

After that, compile the three models and calculated the corresponding mean AUC and standard deviation. The output is shown in table 2. We can see that the first and second model have roughly same mean AUC and standard deviation, with a difference of approximately 0.000715 and 0.002063, respectively. However, the third model results a lower mean AUC and more separate statistics than the other two. To reduce training time without sacrificing accuracy, we chose to use 2 hidden layers.

2.3.3 Dropout and Normalization

Although dropout has been widely recognized as an effective regularization technique against overfitting[20], the inclusion of dropout layers is not strictly necessary in this context. In an earlier implementation, we applied a 15 % dropout rate. However, as shown in Figure 5, this led to a poorer performance. As a result, dropout layers were excluded from the final model.

A similar conclusion was reached regarding normalization layers. Figure 2 shows the distribution of the target variable, which is approximately a normal distribution. Given this relatively balanced distribution, normalization layers were not strictly necessary. In practice, the use of batch normalization introduced additional noise into the batch statistics, as illustrated in Figure 6. Therefore, we omitted the batch normalization layers from the final architecture.

2.3.4 Epoch and Batch Size

Batch size of 32 is often recommended as a starting point [3], but with only 400 samples it can be excessive.. Devarakonda [5] suggests starting with a small batch size and then double that. Based on this guidance, we evaluated batch sizes ranging from 1 to 8, as the result of batch size of 16 is already presented in table 2. For each batch size, we conducted 50 training trials and calculated the corresponding mean AUC and standard deviation. To prevent the impact of outliers and variability due to randomness, the entire process was repeated three times for each batch size. The results are summarized in Table 1, where the bold row indicates the mean values across the three repetitions.

As shown in Table 1, batch sizes of 2 and 4 yielded similar performance, with differences in mean AUC and standard deviation of only 0.001061 and 0.016377, respectively. While batch size 2 achieved a slightly higher best-case mean AUC, batch size 4 demonstrated more consistent performance across trials. A batch size of 1 was not considered due to its significantly longer training time. Given the minimal difference in performance between batch sizes 2 and 4, both are reasonable choices. To improve time efficiency, we selected a batch size of 4 for our final experiments.

2.3.5 Best Threshold Number

We now revisit the selection of the threshold number. To identify the best threshold, we first computed the precision, recall, and corresponding threshold values at each point along the precision-recall curve. To ensure completeness, we also

Table 1: Validation AUC (mean ± SD) over 50 random trials via 2 layers

Epoch Size	Mean AUC	SD
8	0.968312	0.018893
8	0.942113	0.114311
8	0.960851	0.065834
8(mean)	**0.957092**	**0.066346**
4	0.971973	0.012742
4	0.962903	0.068312
4	0.962221	0.068852
4(mean)	**0.965699**	**0.049969**
2	0.972366	0.015341
2	0.973095	0.013652
2	0.954820	0.093634
2(mean)	**0.966760**	**0.040876**
1	0.963619	0.067610
1	0.966166	0.068510
1	0.973534	0.013666
1(mean)	**0.967772**	**0.049929**

included edge cases (i.e., thresholds of 0 and 1) for implementation purposes and stored all values in a list. Next, we initialized variables to track the best threshold and the highest accuracy observed. We then iterated through the list of threshold values, calculating the accuracy for each. If the current accuracy was greater than or equal to the previously recorded maximum, we updated the best threshold accordingly. Figure 11 is an example of the best threshold value for one training set, and the corresponding accuracy is 0.95.

2.4 Data Leakage

At the end, we examined the dataset for potential data leakage to ensure the validity of our results. Following several methods proposed in Yang's study [25], we performed checks including: detection of duplicate rows, verification that the target variable is not mistakenly included in the feature set, and analysis of the top 10 correlation coefficients between feature variables and the target. The code used for these data leakage checks is provided in A.3, and the corresponding output is discussed in Section 3.3. The results show that there are no duplicate rows in the dataset and that the target variable is not present in the feature list. Among all feature-target correlations, the highest observed value was between CGPA and the target variable, with a correlation coefficient of 0.74. While this reflects a relatively strong relationship, it is not unusually high or indicative of data leakage. Therefore, based on these checks, we conclude that no data leakage is present in the dataset.

3 Results

3.1 Logistic Regression

```
Serial No.          0
GRE Score           0
TOEFL Score         0
University Rating   0
SOP                 0
LOR                 1
CGPA                0
Research            0
Chance of Admit     2
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
92	93	298	98	2	4.0	3.0	8.03	0	0.34
347	348	299	94	1	1.0	1.0	7.34	0	0.42
376	377	297	96	2	2.5	2.0	7.43	0	0.34

Figure 3: Outliers for original dataset

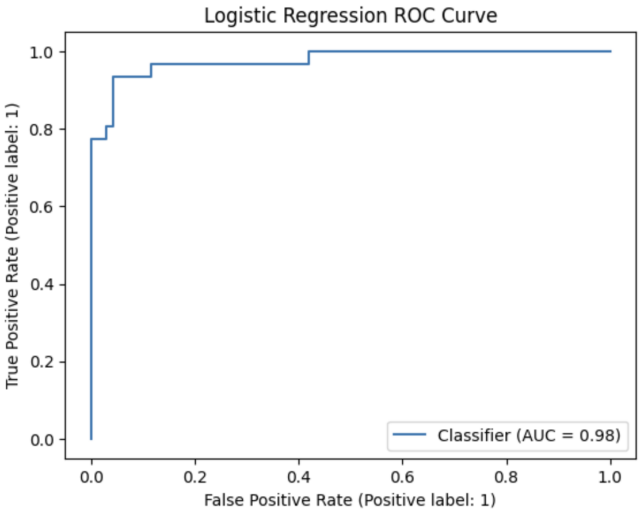

Figure 4: ROC Curve for Logistic Regression

3.2 Artificial Neural Network

3.2.1 Dropout and Normalization

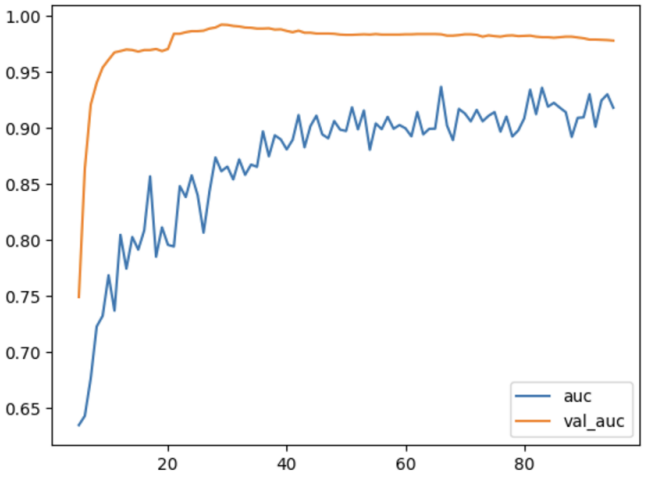

Figure 5: AUC curve after applying 15% dropout layers

3.2.2 Output Curves

Figures 7–10 present the ANN's ROC, loss, AUC, and accuracy curves. Training time was measured using the function in Appendix A.2. The main numerical results are:

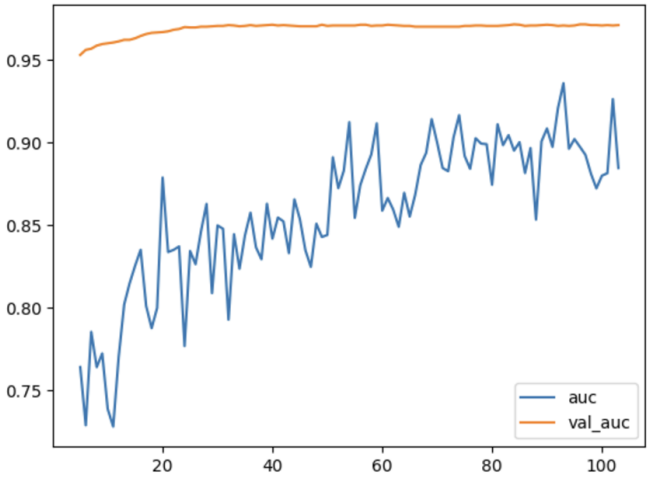

Figure 6: AUC curve after applying batch normalization layers

```
Best Validation Loss: 0.1696
Mean AUC: 0.9126
Best AUC: 0.9724
Mean Validation Accuracy: 0.9067
Best Validation Accuracy: 0.9400
Elapsed time: 37.4175 seconds.
```

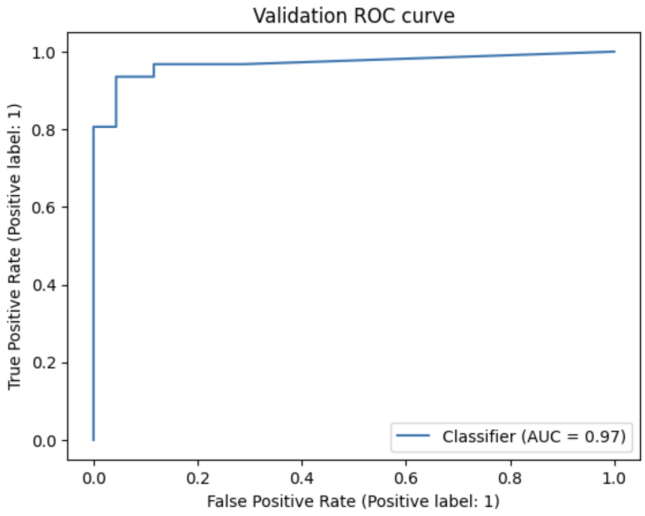

Figure 7: ROC Curve for ANN

Table 2: Validation AUC (mean ± SD) over 50 random trials for epoch = 16

	Mean AUC	SD
2 layers (4,2)	0.951075	0.091352
3 layers (5,4,3)	0.950365	0.093415
3 layers (4,3,2)	0.893679	0.172518

3.3 Data Leakage

```
Exact-row duplicates across splits: 0
```

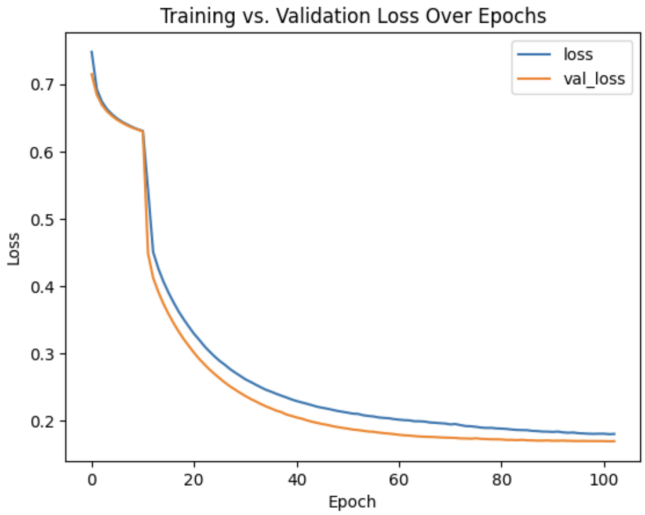

Figure 8: Training and Validation Loss Curve

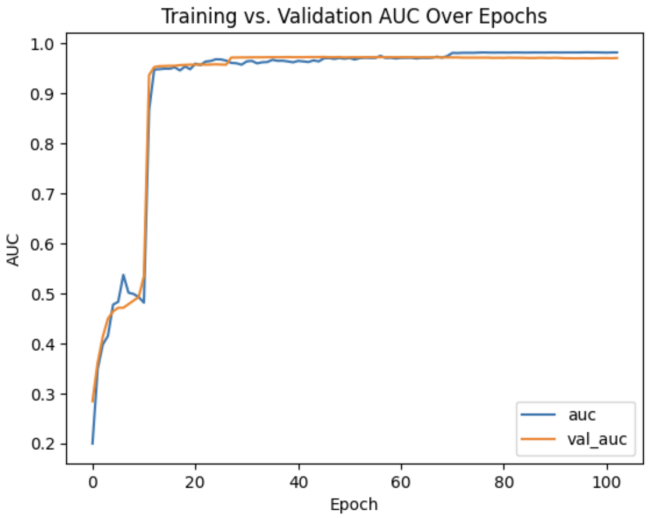

Figure 9: Training and Validation AUC Curve

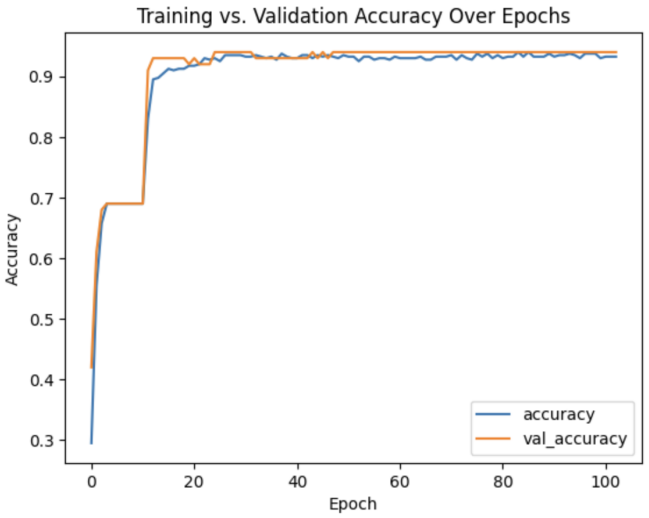

Figure 10: Training and Validation ACC Curve

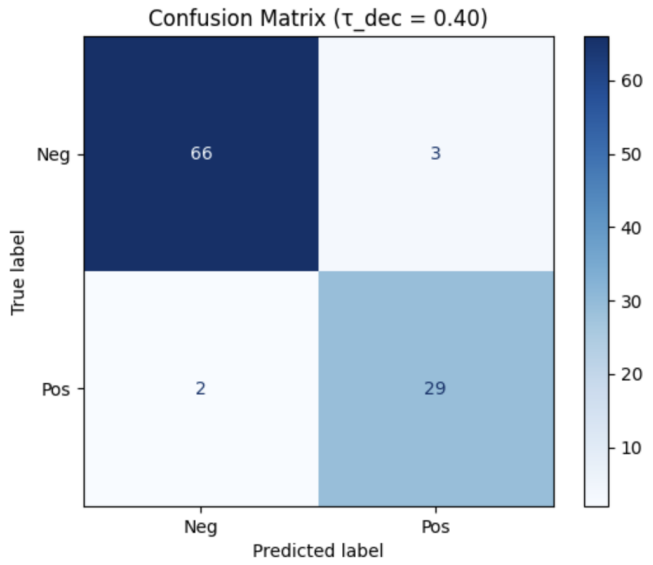

Figure 11: Confusion matrix for the best threshold value of a training set

```
Target column present in feature list? False

Top 10 abs(corr) between features and target:
target               1.000000
CGPA                 0.742278
TOEFL Score          0.699101
GRE Score            0.683633
University Rating    0.618279
SOP                  0.581077
LOR                  0.500391
Research             0.498311
Name: target, dtype: float64
```

4 Discussion

Hassanipour's study [8] highlights several limitations of logistic regression (LR). While LR is effective for modeling causal relationships under certain statistical assumptions, its applicability becomes more restricted compared to artificial neural networks (ANN). In contrast, ANN does not require the same assumptions and are capable of learning complex, non-linear functional relationships from the data. That is, if some of the cells in the network are deleted or have a false function, then there is still a chance to get the correct answer. As a result, ANNs generally achieve higher AUC and accuracy (ACC) scores compared to LR. According to Dreiseitl's research [6], the non-linear output of ANN makes them more flexible than LR, especially when working with continuous data. However, this advantage is not always realized when the dataset is small. In our experiments, ANN is less stable compared to LR. Zantvoort [26] suggests that ANN tend to become more stable when there are more than 500 data. Since our dataset contains only 400 samples, the ANN models were more sensitive to training conditions, resulting in less stability than LR.

Figure 1 displays the correlation coefficients between all feature variables and the target, as well as inter-feature correlations. All feature variables show a meaningful correlation with the target, indicating that a regression model is well-suited for this dataset.

Section 3.2.2 presents the performance of the ANN models, while Section 3.1 provides the results for LR. The best-performing ANN case yields AUC and ACC values comparable to those of LR. However, due to the randomness in training set selection, ANN results show greater variability in both mean and best-case performance, whereas LR results remain consistent across trials.

Table 3: ACC and AUC for ANN

Trial	Mean AUC	Best AUC	Mean ACC	Best ACC
1	0.9308	0.9682	0.9091	0.9500
2	0.9732	0.9805	0.9353	0.9500
3	0.9685	0.9764	0.9035	0.9500

The table above shows that the performance of the ANN model changes depending on the training data used. In comparison, the results for the logistic regression (LR) model stay the same in all trials, indicating that LR is more stable.

5 Conclusions

This study compared the performance of artificial neural networks (ANN) and logistic regression (LR) on a small-scale student admissions dataset. We tested different settings for the ANN model, including the number of hidden layers, threshold values, batch sizes, and epochs. Both models were then evaluated using accuracy (ACC) and area under the ROC curve (AUC) scores. The results show that both models can achieve high accuracy (ACC) and area under the ROC curve (AUC), but LR consistently produced stable results for all trails, while ANN's results changed more depending on the training data. Even though ANN is more flexible and tend to have better performance in general, our ANN model showed greater variability due to the limited data size. We also checked for data issues like duplicate rows or the target variable being included as a feature, and found no signs of data leakage. The feature-to-target correlations showed that a linear model like LR fits this dataset well. In conclusion, although ANN has advantages in handling flexible and non-linear relationships, logistic regression (LR) proves to be more stable for tiny datasets, especially when there is a possible linear relationship between the features and the target variable.

References

[1] M. S. Acharya, A. Armaan, and A. S. Antony. A comparison of regression models for prediction of graduate admissions. In *2019 International Conference on Computational Intelligence in Data Science (ICCIDS)*, pages 1–5, Chennai, India, 2019. IEEE. doi: 10.1109/ICCIDS.2019.8862140. URL `https://doi.org/10.1109/ICCIDS.2019.8862140`.

[2] Mohan S. Acharya. Graduate admission 2, December 2018. URL `https://www.kaggle.com/datasets/`

mohansacharya/graduate-admissions?resource=download.

[3] Jason Brownlee. What is the difference between a batch and an epoch in a neural network. *Machine Learning Mastery*, 20(1):1–15, 2018. URL https://machinelearningmastery.com/what-is-the-difference-between-a-batch-and-an-epoch-in-a-neural-network/.

[4] Jason Brownlee. A Gentle Introduction to Threshold-Moving for Imbalanced Classification. MachineLearningMastery.com, January 4 2021. URL https://machinelearningmastery.com/threshold-moving-for-imbalanced-classification/.

[5] A. Devarakonda, M. Naumov, and M. Garland. Adabatch: Adaptive batch sizes for training deep neural networks. arXiv preprint arXiv:1712.02029, 2017. URL https://arxiv.org/abs/1712.02029.

[6] S. Dreiseitl and L. Ohno-Machado. Logistic regression and artificial neural network classification models: A methodology review. *Journal of Biomedical Informatics*, 35(5-6):352–359, 2002.

[7] UCLA Graduate Education. Historical fall admissions. UCLA Graduate Education. URL https://grad.ucla.edu/graduate-program-statistics/admissions/?t=Historicaltrends.

[8] S. Hassanipour, H. Ghaem, M. Arab-Zozani, M. Seif, M. Fararouei, E. Abdzadeh, and S. Paydar. Comparison of artificial neural network and logistic regression models for prediction of outcomes in trauma patients: A systematic review and meta-analysis. *Injury*, 50(2): 244–250, 2019.

[9] S. Karsoliya. Approximating number of hidden layer neurons in multiple hidden layer bpnn architecture. *International Journal of Engineering Trends and Technology*, 3(6):714–717, 2012.

[10] S. H. Khan, M. Hayat, and F. Porikli. Regularization of deep neural networks with spectral dropout. *Neural Networks*, 110:82–90, 2019.

[11] L. I. Kuncheva, C. E. Matthews, Á. Arnaiz-González, and J. J. Rodríguez. Feature selection from high-dimensional data with very low sample size: A cautionary tale. arXiv preprint arXiv:2008.12025, August 2020. URL https://arxiv.org/abs/2008.12025.

[12] William Franz Lamberti. An overview of explainable and interpretable ai. In Feras A. Batarseh and Laura J. Freeman, editors, *AI Assurance*, pages 55–123. Academic Press, 2023. ISBN 9780323919197. doi: 10.1016/B978-0-32-391919-7.00015-9. URL https://doi.org/10.1016/B978-0-32-391919-7.00015-9.

[13] A. Motrenko, V. Strijov, and G. W. Weber. Sample size determination for logistic regression. *Journal of Computational and Applied Mathematics*, 255:743–752, 2014.

[14] G. Panchal, A. Ganatra, Y. P. Kosta, and D. Panchal. Behaviour analysis of multilayer perceptrons with multiple hidden neurons and hidden layers. *International Journal of Computer Theory and Engineering*, 3(2):332–337, 2011.

[15] S. Park and N. Kwak. Analysis on the dropout effect in convolutional neural networks. In *Computer Vision–ACCV 2016: Revised Selected Papers, Part II*, pages 189–204. Springer International Publishing, 2017.

[16] Parag C. Pendharkar. A threshold-varying artificial neural network approach for classification and its application to bankruptcy prediction problem. *Computers & Operations Research*, 32(10):2561–2582, 2005. ISSN 0305-0548. doi: 10.1016/j.cor.2004.06.023. URL https://doi.org/10.1016/j.cor.2004.06.023.

[17] N. C. Schwertman, M. A. Owens, and R. Adnan. A simple, more general boxplot method for identifying outliers. *Computational Statistics & Data Analysis*, 47(1): 165–174, 2004.

[18] D. Shah. Logistic regression: Definition, use cases, implementation, 2025.

[19] N. Srivastava. Improving neural networks with dropout. Technical Report 182(566), University of Toronto, 2013.

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[21] J. C. Stoltzfus. Logistic regression: A brief primer. *Academic Emergency Medicine*, 18(10):1099–1104, 2011.

[22] A. J. Thomas, M. Petridis, S. D. Walters, S. M. Gheytassi, and R. E. Morgan. Two hidden layers are usually better than one. In G. Boracchi, L. Iliadis, C. Jayne, and A. Likas, editors, *Engineering Applications of Neural Networks (EANN 2017). Communications in Computer and Information Science, vol. 744*. Springer, Cham, 2017. doi: 10.1007/978-3-319-65172-9_24. URL https://doi.org/10.1007/978-3-319-65172-9_24.

[23] M. Uzair and N. Jamil. Effects of hidden layers on the efficiency of neural networks. *2020 IEEE 23rd International Multitopic Conference (INMIC)*, pages 1–6, 2020. doi: 10.1109/INMIC50486.2020.9318195. URL https://doi.org/10.1109/INMIC50486.2020.9318195.

[24] H. P. Vinutha, B. Poornima, and B. M. Sagar. Detection of outliers using interquartile range technique from intrusion dataset. In *Information and Decision Sciences: Proceedings of the 6th International Conference on FICTA*, pages 511–518, Singapore, April 2018. Springer Singapore.

[25] C. Yang, R. A. Brower-Sinning, G. Lewis, and C. Kästner. Data leakage in notebooks: Static detection and better processes. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–12, October 2022.

[26] K. Zantvoort, B. Nacke, D. Görlich, et al. Estimation of minimal data set sizes for machine learning predic-

tions in digital mental health interventions. *npj Digital Medicine*, 7:361, 2024. doi: 10.1038/s41746-024-01360-w. URL https://doi.org/10.1038/s41746-024-01360-w.

A Python Code

A.1 Outliers

```
Q1 = student_data.quantile(0.25)
Q3 = student_data.quantile(0.75)
IQR = Q3 - Q1

outlier_mask =
(student_data < (Q1 - 1.5 * IQR))
| (student_data > (Q3 + 1.5 * IQR))
outlier_counts = outlier_mask.sum()
```

A.2 Time Function

```
def tic():
    global start_time
    start_time = time.perf_counter()

def toc():
    if 'start_time' in globals():
        end_time = time.perf_counter()
        elapsed_time = end_time - start_time
        print(f"Elapsed time:
        {elapsed_time:.4f} seconds.")
    else:
        print("Toc: timer not started.
        Call tic() first.")
```

A.3 Data Leakage

```
train_rows = X.iloc[train_y.index]
val_rows   = X.iloc[val_y.index]

def row_hash(df):
    return pd.util.hash_pandas_object
    (df, index=False).values

train_hash = row_hash(train_rows)
val_hash   = row_hash(val_rows)

dup_overlap = np.intersect1d
(train_hash, val_hash)
print("Exact-row duplicates across splits:",
len(dup_overlap))

print("\nTarget column present in feature
    list?", 'Chance of Admit ' in features)

cor = (
    X.assign(target=y)
    .corr()['target']
    .abs()
    .sort_values(ascending=False)
)
print("\nTop 10 abs(corr) between features
and target:")
print(cor.head(10))
```

A.4 Cook's Distance

```
influence = model.get_influence()
cooks = influence.cooks_distance[0]
flagged = df.index[cooks > 4/len(df)]
print("Potentially influential points:"
, flagged.tolist())
```