Reproducibility Project for CS598 DL4H in Spring 2023: Application of deep and machine learning techniques for multi-label classification performance on psychotic disorder diseases

Shaun Niemeyer
shaunan2@illinois.edu

Group ID: 26
Paper ID: 169
Presentation Link: https://youtu.be/cyg1IaaQ2f8
Code Link: github.com/saniemeyer/dl4h
Notebook Bonus: github.com/saniemeyer/dl4h/blob/main/NotebookBonus.ipynb

1 Introduction

This paper will attempt to replicate and extend the findings of "Application of deep and machine learning techniques for multi-label classification performance on psychotic disorder diseases" conducted by Elujide et al. (2021).

Psychotic Disorder Diseases (PDD) are difficult and time-consuming to diagnose in a clinical setting. Diagnoses require the expert opinion of psychiatrists, who must consider both the patient's medical and psychiatric history. To facilitate this process, the authors propose a diagnostic tool based upon machine learning (ML) algorithms and a small set of easily collected patient data. Their models attempt to identify the presence of five distinct PDD in patients admitted to a psychiatric hospital: bipolar disorder, attention deficit hyperactive disorder (ADHD), schizophrenia, vascular dementia and insomnia. The classification task is implemented both as a single-label task, in which separate binary classification models are constructed for each PDD, and as a multi-label task, where multiple PDD may be non-exclusively assigned to each patient.

2 Scope of reproducibility

The principal claim of the paper is that the bespoke Keras Multilayer Perceptron (MLP) model developed by the authors achieves an accuracy score of 0.7517 and outperforms all of the baseline scikit-learn algorithms that they tested on an imbalanced version of the dataset for the multi-label classification task. This claim is surprising and worth replicating, given the paucity of the dataset, which contains only 500 observations and 12 features. Classification tasks using low-n, low-dimensional, class-imbalanced datasets are precisely the types of problems where traditional ML algorithms frequently outperform neural networks. If the MLP architecture that the authors developed truly outperforms all other traditional ML algorithms, then it might be more generally applicable to a variety of other application domains.

The paper also claimed that the Random Forest classifier was able to achieve an impressive *balanced accuracy* score of 0.6407 on the balanced dataset, outperforming all other algorithms.

As we attempted to experimentally replicate these findings, it became obvious that the authors committed several serious methodological errors that indisputably compromised their results on the balanced dataset. Therefore, rather than focusing on replicating other specific results, which are influenced by the same underlying problems, we will more broadly address the question of whether we can have confidence in *any* of the paper's findings, given the numerous methodological issues that we uncovered.

In summary, we will address the following claims:

- The Keras MLP model developed by the authors achieves an accuracy score of $\sim$0.75 and outperforms the Random Forest, Decision Tree, SVM, and Multi-layer Perceptron classifiers from the scikit-learn library on multi-label classification of the original *imbalanced* version of the dataset.

- The scikit-learn Random Forest Classifier achieves a *balanced accuracy* score of $\sim$0.64 and outperforms all other algorithms in a multi-label classification scenario when data is balanced by oversampling the minority classes.

In the "Discussion" section, we will examine which of the paper's findings are compromised by its methodological errors.

2.1 Additional experiments and ablations

Strangely, the original paper did not investigate the performance of the scikit-learn algorithms on the single-label classification problem. We will therefore extend its research to evaluate these algorithms in this scenario. For both the single-label and multi-label experiments, we will also evaluate one of the most promising ML competitors to deep learning – Gradient Boosting.

In the process of investigating the methodological issues with the study, it was also necessary to conduct multi-*class* classification experiments on both the balanced and imbalanced datasets, so these results will be presented as well.

Finally, ablations on the authors' Keras MLP model will be performed to identify the significance of the particular architecture they have proposed. The ablations will consist of (1) removing one to all of the hidden layers and (2) varying the level of dropout (from 0 to 0.5). If the proposed model is fully optimized for its predictive task, then we expect each of these ablations to result in reduced performance.

3 Methodology

3.1 Model descriptions

The Keras MLP model consists of two separate architectures – one for the multi-label classification task and one for the single-label task. These were implemented using the Keras TensorFlow API. The multi-label model consists of five fully-connected layers with the following units: 15-20-20-40-5. The single-label model consists of five fully-connected layers with the following units: 15-20-40-50-1. For both models, the first four layers use RELU activation, while the output layer utilizes a Sigmoid function.

The ML algorithms against which the authors' Keras MLP model was compared were described as "multilayer perceptron (MLP), random forest (RF), decision tree (DT) and support vector machine (SVM)." The paper did not describe its implementation of these algorithms, other than to say that the libraries included "pandas, numpy, matplotlib and sklearn." Therefore, we assume that the following algorithms from the scikit-learn library were utilized: MLPClassifier, RandomForestClassifier, DecisionTreeClassifier and SVC. Note that SVC does not natively support multi-label classification. We therefore had to wrap the SVC classifier using the sklearn.multioutput.MultiOutputClassifier.

3.2 Data descriptions

The dataset contains 500 records, with each record corresponding to a patient admitted to Yaba Psychiatry Hospital in Lagos, Nigeria. It is publicly available and can be obtained from the Supplementary Data section of Adejumo et al. (2017). There are five target variables, each indicating the binary presence or absence of one of the five PDD. The independent variables consist of generic demographic attributes (sex, age, age range, gender, marital status, religion, and occupation) as well as attributes seemingly more associated with PDD (loss of parent, history of PDD in the family, divorce, head injury, and heredity disposition to one of the disorders). All of the non-numeric features were one-hot encoded.

This dataset exhibits high class imbalance, with the class distributions as follows:

Schizophrenia:	75.0%
Vascular Dementia:	69.2%
ADHD:	43.6%
Bipolar disorder:	40.2%
Insomnia:	40.6%

The combination of these 5 labels results in 19 distinct classes in the data: 01110, 11100, 11101, 11001, 11111, 01100, 01000, 01111, 10001, 10011, 00000, 10111, 11011, 00100, 00110, 01010, 10101, 00010 and 10100 (out 32 *possible* classes). However, 7 of these classes contained between 1 and 5 records, complicating the ability to perform oversampling. Therefore the authors removed any records for classes with less than six records, leaving a total of 12 classes and 484 records. A second "balanced" version of this dataset was created using the synthetic minority oversampling technique (SMOTE). After performing SMOTE to oversample all of the minority classes, the dataset consisted of 101 records for each of the 12 classes, for a total of 1,212 records.

Inexplicably, the authors used a 70/30 train-test split for the Keras MLP model and a 20/80 train-test split for the scikit-learn ML models, complicating the comparison of the models.

3.3 Hyperparameters

For the Keras MLP model, the authors used an Adam optimizer with a learning rate of 0.01. Each model was run for 40 epochs. An early stopping callback was specified with a monitor on validation loss. Loss was computed as Binary Cross-Entropy. The authors did not specify the level of dropout, though they did note that dropout was used. Consequently, we experimented with various levels of dropout using grid search, testing rates between 0 and 0.5.

No hyperparameters for the baseline ML algorithms were specified in the paper, but there is reason to believe that the authors used scikit-learn's default parameters. In their previous study on the same data set [3], they reference Tahir et al. (2010) as their justification for using default parameters for the ML algorithms, so presumably they adhered to the same recommendation for this study.

3.4 Implementation

The original paper provided no source code, but it did describe the principal Keras MLP model in sufficient detail. It was therefore necessary to re-implement all experiments using our own code, using the same computing environments and programming libraries that the authors referenced. The github repository for our code can be found at https://github.com/saniemeyer/dl4h-psychotic-disorders-replication

3.5 Computational requirements

The authors used two different environments for their study. Their Keras MLP model was run in a Google Colaboratory Notebook, using Python 3.6. The scikit-learn ML models were run in Python 3.7 on a Windows 10 operating system with 32 GB of RAM. We used the same environments, specifically configuring Google Colab to use a Python 3.6 interpreter. As the dataset is tiny, the default resources provided by Google Colab (Intel Xeon CPU @ 2.30GHz, Tesa T4 GPU) were sufficient for this task. RAM utilization did not exceed 2.54 GB for any of the experiments.

We performed a total of 10 experiments, not including the ablations. Each experiment utilized 100 splits of the data. The ablations for the Keras MLP model used 30 splits each and tested 26 combinations of hyperparameters. The resulting processing times (in minutes) are shown in Table 1.

Table 1: Experiment Run-Times, in Minutes

	Imbalanced			Balanced		
	Multi-Label	Multi-Class	Single-Label	Multi-Label	Multi-Class	Single-Label
Keras MLP	6.5	N/A	23.3	8.8	N/A	29.3
Scikit-Learn	3.6	5.6	13.4	7.0	10.3	19.9

4 Results

The principal claim of the paper, that the Keras MLP model developed by the authors outperforms all of the scikit-learn ML models was *not* supported by our findings; three out of the four scikit-learn algorithms outperformed the Keras model. However, using binary accuracy as our evaluation metric, our results for the Keras MLP very closely match the reported results. We were able to replicate the finding that Random Forest outperforms all other algorithms on the balanced dataset and obtains a balanced accuracy score comparable to that reported in the paper. However, on the whole, our results were very different from those reported in the paper. We believe that there is a good reason for these discrepancies, which we will subsequently discuss.

4.1 Imbalanced Multi-Label Classification

The results of the experiments on the imbalanced dataset are shown in Table 2. Using binary accuracy, we were able to replicate the reported results, obtaining a score within 0.7 percentage points of that reported in the paper. However, using binary accuracy to evaluate the scikit-learn algorithms resulted in scores *far above* those reported in the paper, indicating that this is *not* the measure that the authors used to evaluate these algorithms. If, instead, we use multi-class classification and evaluate accuracy using the sckit-learn "accuracy_score" function, then we obtain results lower than those reported in the paper, but closer than the replicated scores for multi-label binary accuracy. Incidentally, it is peculiar that three of the paper's algorithms produced exactly the same accuracy score; this seems quite unlikely for such diverse algorithms on a multi-label classification task where thousands of individual labels are being predicted. Unfortunately, the authors do not comment on this anomaly, and so these results just raise additional questions about the paper's methodology.

4.2 Balanced Multi-Label Classification

The results of the experiments on the balanced dataset are shown in Table 3. Here, the Random

Table 2: Imbalanced Multi-Label Classification

	Reported	Replicated	
	"Accuracy"	Multi-Label Binary Acc.	Multi-Class Accuracy
Keras MLP	0.7517	0.7450	N/A
MLP (sklearn)	0.5844	0.7703	0.3963
SVM	0.4691	0.7478	0.4253
RF	0.4691	0.7573	0.3533
DT	0.4691	0.6942	0.2809
GBM	N/A	0.7674	0.3443

Table 3: Balanced Multi-Label Classification

	Reported	Replicated	
	Balanced Accuracy	Multi-Label Balanced Acc. (conv)	Multi-Class Balanced Accuracy
Keras MLP	0.5587	0.3103	N/A
MLP (sklearn)	0.5853	0.3870	0.4785
SVM	0.4982	0.2254	0.5203
RF	0.6407	0.6076	0.6245
DT	0.4982	0.5623	0.5582
GBM	N/A	0.4843	0.5985

Forest classifier achieved a balanced accuracy score of 0.6076 using multi-label classification (achieved only by converting the predictions to combinations representing the target classes), and a balanced accuracy score of 0.6245 using multi-class classification. The balanced accuracy scores obtained using multi-*class* classification are clearly much closer to the reported results than those for multi-*label* classification.

4.3 Additional results not present in the original paper

Our additional experiments on the performance of the scikit-learn ML algorithms on the single-label classification tasks, shown in Table 4, found that Random Forest outperformed all other algorithms on the *balanced* dataset, while SVM performed best on three out of five PDD on the imbalanced dataset, achieving an impressive 94% accuracy for schizophrenia.

Table 4: Scikit-learn Imbalanced Single-Label Accuracy

PDD	Best Model	Accuracy
Insomnia	SVM	0.7254
Schizophrenia	SVM	0.9406
Vascular dementia	MLP	0.8032
ADHD	GBM	0.7916
Bipolar disorder	SVM	0.7254

Contrary to expectations, the Gradient Boosting Classifier did not outperform the other scikit-learn algorithms, except on single-label imbalanced classification of ADHD.

The ablation studies found that, for the imbalanced data set and the multi-label task, the best binary accuracy was achieved by excluding all three hidden layers and using dropout of 0.1. This increased binary accuracy to 0.7503. For the balanced dataset, the best balanced accuracy was achieved by excluding the first two hidden layers and setting dropout to 0.15. Both of these results, together with the finding that most of the scikit-learn algorithms outperform the Keras model, suggest that the reported architecture is not uniquely suited to this task and could be improved upon.

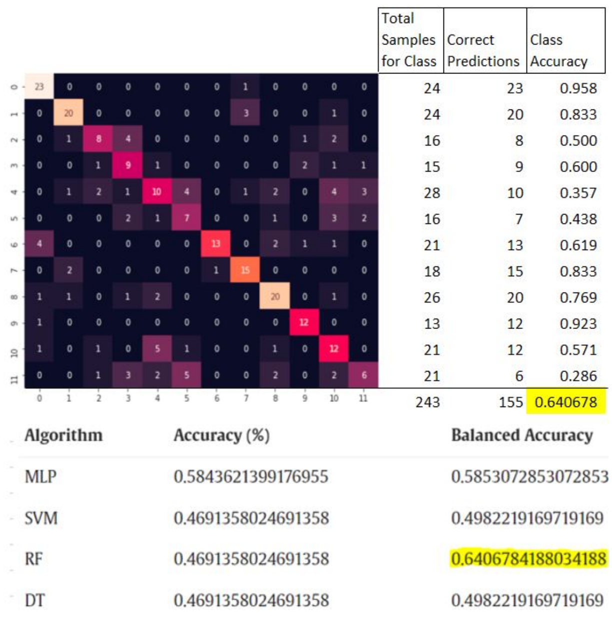
5 Discussion

While attempting to replicate the paper's findings, we discovered a number of methodological errors that invalidate many of these findings and shed doubt on others. We believe that discrepancies between the reported and replicated results are largely a consequence of these errors.

The most significant problems with the study arose from the authors' attempt to perform multi-label classification. After performing SMOTE on the data, as described above, they split the data into train and test sets. This, by itself, is a very basic error. SMOTE should only be performed on training data, never test data. Utilizing data in the test set that has been synthetically generated by the SMOTE technique violates the assumption that the train and test sets are independent, leading to inflated and unrealistic validation performance. Additionally, the authors state that a 70/30 train-test split was used for the Keras MLP models, while an 80/20 split was used for the scikit-learn models. The models were therefore unfairly compared, as they were evaluated on differently sized datasets.

This disparity would have been exacerbated by the small size of the dataset. Similarly, both sets of models should have been run in the same environment, not using different Python interpreters and library versions.

The authors then used a multi-*class* accuracy metric, "balanced accuracy," to evaluate what were *supposedly* multi-label classifiers. We know exactly how balanced accuracy was calculated because they provide a confusion matrix in Figure 8 of the original paper which we can compare to the balanced accuracy results presented in Table 3 of the original paper (both are reproduced here as Figure 1, with annotations):



	0	1	2	3	4	5	6	7	8	9	10	11	Total Samples for Class	Correct Predictions	Class Accuracy
0	23	0	0	0	0	0	0	1	0	0	0	0	24	23	0.958
1	0	20	0	0	0	0	0	3	0	0	1	0	24	20	0.833
2	0	1	8	4	0	0	0	0	0	1	2	0	16	8	0.500
3	0	0	1	9	1	0	0	0	0	2	1	1	15	9	0.600
4	0	1	2	1	10	4	0	1	2	0	4	3	28	10	0.357
5	0	0	0	2	1	7	0	0	1	0	3	2	16	7	0.438
6	4	0	0	0	0	0	13	0	2	1	1	0	21	13	0.619
7	0	2	0	0	0	0	1	15	0	0	0	0	18	15	0.833
8	1	1	0	1	2	0	0	0	20	0	1	0	26	20	0.769
9	1	0	0	0	0	0	0	0	0	12	0	0	13	12	0.923
10	1	0	1	0	5	1	0	0	1	0	12	0	21	12	0.571
11	0	0	1	3	2	5	0	0	2	0	2	6	21	6	0.286
													243	155	0.640678

Algorithm	Accuracy (%)	Balanced Accuracy
MLP	0.5843621399176955	0.5853072853072853
SVM	0.4691358024691358	0.4982219169719169
RF	0.4691358024691358	0.6406784188034188
DT	0.4691358024691358	0.4982219169719169

Figure 1: Random forest confusion matrix

From the confusion matrix, we can see that balanced accuracy is being calculated by counting the number of correctly predicted samples for each class, dividing by the total number of samples in that class, and then averaging over this class-level accuracy. For the Random Forest classifier, this results in a balanced accuracy score of 0.640678, which corresponds to the score the authors reported in Table 3 of the original paper and subsequently compared to the Keras MLP classification accuracy. This calculation also corresponds to scikit-learn's sklearn.metrics.balanced_accuracy_score. Although there *is* a multi-label definition of balanced accuracy, that is clearly not what was used here.

We know for a fact that the *Keras MLP* model is performing multi-*label* classification because its output layer has 5 units and uses Sigmoid activation. So by using balanced accuracy to evaluate the scikit-learn classifiers, the authors necessarily committed one of the following fatal errors: Either they performed multi-*class* classification using the scikit-learn classifiers and compared this performance against the Keras MLP multi-*label* classifier, or they performed multi-label classification with both Keras and sckit-learn and performed a post-prediction conversion of the 5 predicted labels to the 12 target classes. It is obviously inappropriate to convert the predictions of a multi-label classifier to classes by concatenating the predictions; this results in a loss of information, because a multi-label classifier can predict combinations of the labels that may not exist in the data (there are 32 possible combinations of the 5 labels, but only 12 classes in the data), while a multi-class classifier will only predict classes which do exist.

Regardless of which approach the authors took, either approach would render their experiments on the balanced dataset invalid. However, we performed additional experiments using multi-class classification in order to forensically identify the approach that was implemented. As we suspected, our results from multi-class classification were much closer to the paper's reported results than were our results from multi-label classification. However, yet another strong indicator that the authors used multi-class classification with the scikit-learn algorithms is that the SVM algorithm *cannot be directly used for multi-label classification*, but it can be used for multi-class classification. In order to use SVM for multi-label classification, it has to be wrapped in the scikit-learn MultiOutputClassifier, or it is necessary to perform some version of one-vs.-one or one-vs.-all classification. The authors do not mention taking either of these actions.

Moving on to the experiments on the imbalanced dataset, it makes sense to question whether multi-class classification was *also* used for the scikit-learn algorithhms in this scenario. If the authors used the concatenated target class as their dependent variable for the balanced experiments, it would be strange if they reverted to using the five individual PDD labels as targets for the imbalanced experiments. The problem here is that, unlike the balanced scenario, we really don't know what measure of accuracy the authors selected. They define accuracy ambiguously as "the sum of number of true positives and true negatives (number of correct predictions) divided by the total number of predictions made."

$$\frac{TP + TN}{TP + TN + FP + FN}$$

Interpreted literally, in the multi-label scenario this should equate to binary accuracy, and the Keras MLP results indicate that this *is* the metric that they inadvertently used. However, it does not appear that binary accuracy was used for the scikit-learn experiments on the imbalanced dataset because the reported scores are far below those achieved from replication. Moreover, there is no binary accuracy measure in the scikit-learn library, and the authors do not mention importing any additional libraries or implementing their own custom metrics.

So this brings us to the other likely error that the authors committed - using inconsistent measures of accuracy. With the Keras API, when compiling a model it is necessary to specify the evaluation metric (using the "metrics" parameter) and the loss function, which the authors state was binary cross-entropy. If they naively used the string "accuracy" for the "metrics" parameter of the "compile" method, then given the 5-unit Dense output layer with Sigmoid activation and binary cross-entropy loss, Keras would have selected BinaryAccuracy as the evaluation metric. This is likely what the authors did, and is why they achieved such high scores for the Keras MLP model as compared to the scikit-learn models. As we know that the authors used the scikit-learn "balanced_accuracy_score" metric for the *balanced* dataset, it is likely that they used the scikit-learn "accuracy_score" metric for the *imbalanced* dataset. Using "accuracy_score" with multi-label classification would compute "subset" or "exact match" accuracy, and would be very low. Using "accuracy_score" with multi-class classification results in the scores that we presented in Table 2, which are lower but within the same range as those reported in the study. Therefore, this seems to be the most likely scenario and explains the highly discrepant results obtained on the imbalanced dataset.

In conclusion, we established that the paper's experiments with the balanced dataset were compromised for a variety of reasons, and are essentially meaningless without a valid comparative context. We also have good reason to believe that the multi-label (or multi-class) experiments on the imbalanced dataset were also invalid, though we can't make this statement with certainty.

5.1 What was easy

The Keras MLP model definitions were clearly defined, so it was not difficult to duplicate the architecture of the proposed models. As the paper seems to have used default parameters for the scikit-learn algorithms, implementing classification using these models was also straightforward. The models were quick to train, given the small size of the dataset (only 500 records and 26 KB). Apart from some confusion about the separation of the train and test sets, the data preparation was also quite simple and well-documented – we simply had to one-hot encode all non-numeric features, remove under-represented classes, and perform SMOTE to create the balanced dataset.

5.2 What was difficult

The paper's imprecise definition of its accuracy measures required us to perform many permutations of the experiments in order to determine how the paper was calculating "accuracy." The fact that they utilized over-sampling on the test set was also confusing, as we initially assumed the paper's methodology was sound. Once we determined that balanced accuracy was being used for the scikit-learn algorithms on the balanced dataset, we then had to determine how the authors were utilizing this metric for multi-label classification. As it turns out, we don't believe that they were (they were instead performing multi-class classification, in contradiction to the title of their paper). So additional experiments had to be conducted in order to test the hypothesis that multi-class classification was utilized.

5.3 Recommendations for reproducibility

To reliably reproduce the paper's findings, a source code repository would be invaluable. Lacking that, the paper should explicitly define its accuracy measures. The hyperparameters used for both the Keras MLP model and the scikit-learn models should be fully specified. The authors should also address the perceived methodological issues with their research: If they compared the Keras model's multi-label classification against the scikit-learn's multi-class classification, what was the justification for this decision? Why did they perform SMOTE on the dataset before performing a train/test split? Was the data separately balanced for each of the single-label classification experiments?

References

[1] Israel Elujide, Stephen G. Fashoto, Bunmi Fashoto, Elliot Mbunge, Sakinat O. Folorunso, and Jeremiah O. Olamijuwon. 2021. Application of deep and machine learning techniques for multi-label classification performance on psychotic disorder diseases. Informatics in Medicine Unlocked, 23:100545.

[2] O Adejumo Adebowale, Nehemiah A Ikoba, Suleiman Esivue A, Okagbue Hilary I, Oguntunde Pelumi E, Odetunmibi Oluwole A, Job Obalowu. Quantitative exploration of factors influencing psychotic disorder ailments in Nigeria. Data in Brief 2017;14:175–85.

[3] Folorunso SO, Fashoto SG, Olaomi J, Fashoto OY. A multi-label learning model for psychotic diseases in Nigeria. Inf Med Unlocked 2020;19:100326.

[4] M.A. Tahir, J. Kittler, K. Mikolajczyk, F. Yan Improving multilabel classification performance by using ensemble of multi-label, classifiers N. El Gayar, J. Kittler, F. Roli (Eds.), MCS, Springer-Verlag Berlin Heidelberg (2010), pp. 11-21.