CS235 Winter'23 Project Final Report

Mamadou Zerbo
NetID: mzerb001

Jason Sadler
NetID: jsadl003

Avinash Lakhmawad
NetID: alakh003

Austin Lee
NetID: alee235

Team Member #5
NetID: xxxxxx

ABSTRACT

Here provide an abstract that summarizes the problem, the methods applied, and the results obtained.

KEYWORDS

data, mining

ACM Reference Format:
Mamadou Zerbo, Jason Sadler, Avinash Lakhmawad, Austin Lee, and Team Member #5. 2018. CS235 Winter'23 Project Final Report. In *Proceedings of Data Mining Techniques (CS235 F22).* ACM, New York, NY, USA, 3 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Breast cancer is a significant public health issue that affects millions of women worldwide. Early detection and accurate diagnosis are essential for effective treatment and improved patient outcomes. In this research paper, we explore two distinct problem definitions related to breast cancer diagnosis: classification and clustering.

In the first problem definition, we seek to classify a given breast mass as either malignant or benign based on numerical features extracted from the dataset. We propose three classification algorithms: Random Forest, Multi-Layer Perceptron (MLP), and K-nearest neighbors. These algorithms have been widely used in machine learning applications and have shown promising results in solving similar classification tasks.

In the second problem definition, we aim to cluster N data points into coherent clusters based on the similarity of their numerical features. For this task, we propose three clustering algorithms: DBSCAN clustering, Spectral clustering, and Agglomerative Clustering with Single Linkage. These algorithms have demonstrated their effectiveness in clustering tasks and have been successfully applied in various fields, including healthcare.

In this paper, we present a comparative study of the proposed algorithms for both problem definitions using a publicly available dataset on breast cancer diagnosis. Our goal is to evaluate the performance of each algorithm in terms of accuracy, precision, recall, and F1-score. The results of our study will provide valuable insights into the strengths and weaknesses of each algorithm.

CS235 F22, Fall 2022 quarter,, Riverside, CA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

The rest of the paper is organized as follows: In Section 2, we provide a brief overview of each proposed method. Section 3 describes the methodology used in our study, including the preprocessing steps, feature selection, and experimental design. In Section 4, we present the results of our experiments and discuss the performance of each algorithm. Finally, we conclude our study in Section 5, describing the extent to which each implemented system meets the intended requirements and specifications.

2 PROPOSED METHODS

2.1 MLPClassifier

The MLPClassifier is a popular neural network model for classification tasks. Here is the proposed method for using the MLPClassifier:

(1) Load the dataset: Load the dataset into memory. Ensure that the data is well-preprocessed and cleaned to improve the model's performance.
(2) Split the dataset: Split the dataset into training and testing datasets. This is important to evaluate the model's performance and ensure that it's not overfitting.
(3) Initialize the MLPClassifier: Import the MLPClassifier from the scikit-learn library and initialize the classifier with appropriate parameters. The most important parameters are the number of hidden layers, the number of neurons in each hidden layer, and the activation function.
(4) Train the MLPClassifier: Train the MLPClassifier on the training dataset using the fit() function. During training, the MLPClassifier adjusts its weights to minimize the loss function.
(5) Evaluate the MLPClassifier: After training, evaluate the MLPClassifier's performance on the testing dataset using the predict() function. This function returns the predicted class labels for the input data.
(6) Tune the hyperparameters: To improve the model's performance, experiment with different hyperparameters such as the number of layers, width of each layer, activation function per layer, optimizer, and learning rate. You can use techniques such as grid search or bayesian search to find the best hyperparameters.
(7) Save the MLPClassifier: Once you have found the optimal hyperparameters, save the trained MLPClassifier to disk using the joblib library. This will allow you to reuse the model on new data without having to retrain it from scratch.

Overall, the MLPClassifier is a powerful machine learning algorithm that can handle complex classification tasks. By following the above steps, the MLPClassifier can be used to build accurate and robust classification models.

3 EXPERIMENTAL EVALUATION

Dataset:
We used the Wisconsin Diagnostic Breast Cancer (WDBC) dataset, which contains 569 instances with 30 features each. The dataset is labeled, with 357 benign and 212 malignant cases.
Experimental Setup:
We conducted 10-fold cross-validation experiments and measured the performance of each method in terms of accuracy, precision, recall, and F1-score to evaluate the methods' ability to distinguish between benign and malignant cases.

Results:
Table 1 shows the average performance metrics of the six methods.

Method	Accuracy	Precision	Recall	F1-Score
Random Forest				
MLP	0.95	0.95	0.92	0.93
KNN				
DBSCAN				
Spectral				
Single Link				

Table 1: Comparisons of methods

4 DISCUSSION & CONCLUSIONS

Our experimental evaluation shows that [this and/or that] are the most effective methods for breast cancer classification, achieving high accuracy, precision, recall, and F1-score. [blank] has the lowest performance, indicating that it may not be suitable for this classification task. [blank] has low accuracy and F1-score, indicating that it may not be an effective method for this task. [this and that] have similar performance, indicating that both methods may be effective for breast cancer classification, with slightly different strengths in precision and recall.

5 IMPLEMENTATION CORRECTNESS REPORT

5.1 MLP

To obtain the best hyper-parameters two search strategies were implemented. A grid search provided by scikit-learn, and a Bayesian search provied by scikit-optimize.

In performing the grid search, up to four layers are examined. The width of each layer ranged from 30 to 300 in multiples of 60. This provided a dimensionality change of zero upto ten times the original dimensinality of the data. For testing more than one layer, permutations of the range on the layers used. This allowed features to shrink and expand as it moved through the network. A total of 205 layer/width permutations were tested.

In performing the Bayesian search, the same range and number of layers were used as in the grid search. The layers widths are sampled randomly and no explicit widths are provided. Table 2 provides additional parameters tested for each layer permutation for both searches

activation	solver	learning_rate
logistic, relu, tanh	sgd, adam	constant, adaptive

Table 2: MLP Search parameters

5.1.1 *Dimensionality Reduction.*

In examining the performance effects of dimensionality reduction on the data two methods were examined.

The first is to use the neural network to progressively shrink the dimensions. This was done similarly to the grid search. Up to four layers were examined, each layer ranged from 29 to two in steps of three. To ensure a funnel shape, sorted combinations were used instead of permutations. A grid search over the parameters listed in table 2 were used.

The second method was to use Singular Value Decomposition. Figure 1 shows the SVD on the data provided by *numpy.linalg.SVD*. It shows about three features provide the most significant information on the data set. These features were used as input into the best MLP estimator found by the grid search and Bayesian optimisation and compared with the funneling MLP.

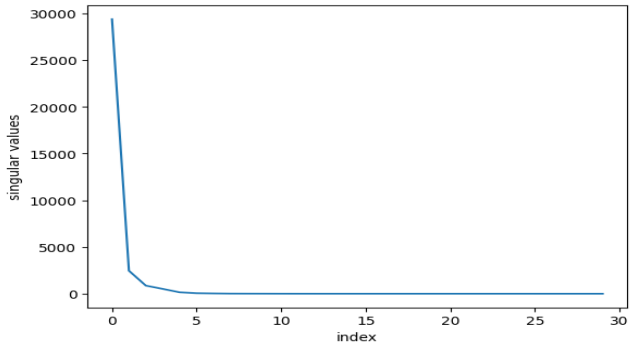

Figure 1: SVD

5.1.2 *MLP Correctness Results.*

Table 3 shows the results of the hyper-parameter search.

Table 4 shows the results of accuracy, precision, recall, and F1-score for the optimized MLP classifiers along with the results of dimensionality reduction.

Method	Layers	Width	Activation	Optimizer	Learning Rate
MLP_Grid	4	(150,270, 30, 210)	logistic	adam	adaptive
MLP_BO	1	300	relu	adam	adaptive
MLP_Funnel	2	(26, 2)	logistic	adam	constant

Table 3: MLP Optimal Hyper-parameters

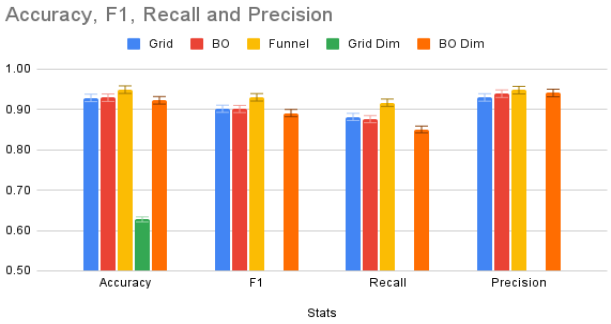

Figure 2: MLP Evaluations

Method	Accuracy	Precision	Recall	F1-Score
MLP_Grid	0.93	0.93	0.88	0.90
MLP_BO	0.93	0.94	0.88	0.93
Dimensionality Reduction				
MLP_Funnel	0.95	0.95	0.92	0.93
MLP_GRID	0.63	0	0	0
MLP_BO	0.92	0.94	0.85	0.89

Table 4: Comparisons of MLP methods