CS235 Quarter'XX Project Final Report: Your Project Title Here

Ruotian Shi #1
NetID: rshi023

Tianyang Li #2
NetID: tli172

Luofeng Xu #3
NetID: lxu092

Jayden Guan #4
NetID: yguan030

Yijian Chai #5
NetID: ychai024

Mingxuan Zhang #6
NetID: mzhan104

ABSTRACT

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

KEYWORDS

data, mining

ACM Reference Format:
Ruotian Shi #1, Tianyang Li #2, Luofeng Xu #3, Jayden Guan #4, Yijian Chai #5, and Mingxuan Zhang #6. 2018. CS235 Quarter'XX Project Final Report: Your Project Title Here. In *Proceedings of Data Mining Techniques (CS235 F22).* ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

1.1 Introduction

In this project we are going to work with the Breast Cancer Wisconsin Dataset with 6 different methods(1.Random Forest classifier 2.Multi-Layer Perceptron (MLP) 3.K-nearest neighbors 4. DBSCAN clustering 5.Spectral clustering 6.Agglomerative Clustering with Single Linkage).We are going to evaluate all of them, and find out the differences between different methods under the same Dataset.

2 PROPOSED METHODS

2.1 Random Forest Method

In the decision tree method, we use a basic algorithm: the tree constructs in a top-down recursively. At the beginning top, all the training examples are at the top root. For each split, we will find the larger information gain to determine which features and points should be split. Then we will have a left child node and a right child node. We repeat those steps until we get the final leaves. Once, we have input, we can check which node belongs to this input, then we will return a prediction. Using this decision tree and its prediction result, we can construct a random forest that consists of some decision trees. The random forest method will depend on many trees' results and return the prediction. If many trees predict a feature A, then the random forest will show feature A as the final result. In random forest, we will random sub-sampling and random select features to construct each tree.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CS235 F22, Fall 2022 quarter,, Riverside, CA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

2.2 2.2 MLP Method

In this part, we used the multilayer perceptron (MLP) method to classify the Breast Cancer Wisconsin data.Since our task is binary classification, we converted the diagnostic results "M" to 1 and "B" to 0. And I deleted two columns: "id" and "Unnamed: 32" in original data. The final sample dimension is 30-dimensional feature information. Then we used StandardScaler in sklearn to normalize the data. In the middle we implemented a neural network manually. I used optim in pytorch to update the weight of each layer, so that I can implement the whole process of the MLP, which is to input data set, forward propagation to get the loss, calculate the gradient by the loss, update the weights by the gradient, and finally get the global best solution. Now we need to tune our hyperparameters by using gridsearch and bayesain optimization. And we need to use GridSearchCV in sklearn and BayesSearchCV in skopt, which require us to pass in a MLP model that can set hyperparameters directly. So we choose to use MLPClassifier in sklearn to implement our MLP model. We input our data to our MLP medel and use ten-fold cross validation to train it. During the process of tuning the hyperparameters, We experimented with different hyperparameter combinations using Grid Search and Bayesian Optimization, evaluated the scores of each model, compared them, and finally selected the hyperparameter combination with the best performance. In order to determine the best architecture of MLP for the specific problem, we systematically optimized the following design options and hyperparameters: 1) number of layers 2) width of each layer 3) activation function of each layer 4) optimizer selection 5) learning rate. On this basis, the performance difference between SVD dimension reduction and MLP dimension reduction is considered.

2.3 K-nearest Neighbors Method

2.3.1 Algorithm Introduction. The k-nearest neighbors (k-NN) algorithm is a popular machine learning algorithm used for classification and regression tasks. It is a non-parametric method that works by finding the k-nearest data points in a given dataset to a new data point and predicting its label or value based on the labels or values of its nearest neighbors. In other words, the k-NN algorithm classifies or predicts the label or value of a new data point based on the majority label or average value of its k-nearest neighbors. The choice of the value of k is critical, as a small value of k may lead to overfitting, while a large value may lead to underfitting. The k-NN algorithm is simple and easy to implement, making it a popular choice for many applications. The algorithm has a solid theoretical foundation and has been shown to perform well in many practical applications. However, the algorithm can be computationally intensive, especially for large datasets. Nevertheless, with the availability of modern computing resources, the k-NN algorithm remains a powerful and useful tool in machine learning. In this project, we will include 2 different distance calculating function (Euclidean and Manhattan), and 2 different feature representations methods (SVD and MLP-based AutoEncoder).

2.3.2 Data Pre-processing. Except the normal data cleaning, we also apply two different dimension reduction methods. The first one is low rank approximation of the data using the Singular Value Decomposition (SVD) for 2 different selections of the approximation rank: (i) a “low” value (i.e., before the singular values drop dramatically) and (ii) a “high” value (i.e., right after the singular values drop dramatically). In total, we have 2 different SVD-based feature reps. Through SVD decomposition, we can keep most useful information from the original feature matrix, and reduce the noise. In the figure 1 shows the plot of singular value in semi-log, the dramatically drop is between the first and the second singular value, and they will be the "low" and "high" value.

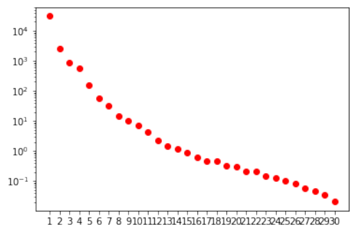


Figure 1: Singular value of original feature matrix

The second method is an off-the-shelf MLP-based AutoEncoder where the encoder and the decoder have 2 layers with ReLu activations. With the bottle neck have two different value: (i) 5% of the original features, which is 2, (ii) 20% of the original features, which is 6. Autoencoders are a type of deep learning architecture that specializes in learning representations of data, primarily for the purpose of reducing dimensionality. This is accomplished by constructing a deep learning architecture that aims to replicate the input layer in its output layer.

2.3.3 Hyperparamter Tuning. The hyperparamter for k-nearest neighbors method is the "k", which indicates how many nearest neighbors we need to measure. We use cross validation and accuracy score for the hyperparamter tuning. There is no specific range for K, but it will be better to use a odd value to avoid even result. Since we have total 10 different situation need to measure, here I put a list of odd value from 1 to 11 as the hyperparamters, and use the one with highest accuracy score for the next step. The k value for each situation is list in the table 1.

Table 1: K value

Distance	Features	K
Euclidean	Original	7
Manhattan	Original	5
Euclidean	Low SVD	11
Euclidean	High SVD	11
Euclidean	5% MLP	7
Euclidean	20% MLP	5
Manhattan	Low SVD	11
Manhattan	High SVD	7
Manhattan	5% MLP	9
Manhattan	20% MLP	7

2.4 Method: DBSCAN Clustering

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a clustering algorithm used in machine learning and data mining. It is a density-based clustering algorithm that groups together points that are close to each other and separates those that are far away.

The algorithm starts with an arbitrary point in the dataset and examines its neighborhood to determine whether it should be included in a cluster. If the point has a sufficient number of neighbors, it is added to the cluster. The process continues until all points have been examined, and the algorithm assigns each point to a cluster or marks it as an outlier.

2.5 Spectral clustering

Spectral Clustering: We used the k-nearest neighbors algorithm to calculate the similarity matrix, which was then used to calculate the degree and Laplacian matrices. We then calculated the eigenvalues and eigenvectors of the Laplacian matrix and sorted them in ascending order. We selected the smallest k eigenvectors and normalized them. Finally, we applied K-means to the normalized eigenvectors to cluster the data.

K-Means: We initialized K-means with default three clusters, a maximum of 100 iterations, and no random seed. We then fit the model to the data and predicted the cluster labels for the data.

Visualization: We visualized the data using a scatter plot with the first two eigenvectors as the x and y axes. We colored each point according to its cluster label and plotted the centroids of each cluster in red.

There are some part missing and unexpected errors happened in cross-validation for the midterm method so I changed the direction.

2.6 Agglomerative Clustering with Single Linkage

2.6.1 Method Introduction. Agglomerative Clustering with Single Linkage: Agglomerative clustering is a type of hierarchical clustering used to group objects in clusters based on their similarity. The algorithm starts by treating each object as a singleton cluster. Next, pairs of clusters are successively merged until all clusters have been merged into one big cluster containing all objects.

To implement Agglomerative Clustering with Single Linkage, I separate the coding process into five different functions, the first function computes the pairwise distance matrix between all data points, the second function finds the two closest clusters and returns them, the third function merges the two closest clusters into a single cluster, the fourth function update the distance matrix to reflect the new cluster, and the last function combined all the steps together to implement agglomerative clustering with single linkage.

2.6.2 Preprocessing. To clean up data, "id" and "diagnosis" have been removed as long as the last empty column, since we only care about the similarity between clusters. The function $StandardScaler()$ and function $normalize()$ have been used for cccccccccccccccccc-ccc, $StandardScaler()$ standardize features by removing the mean and scaling to unit variance, and $normalize()$ scales vectors individually to a unit norm. PCA has also been used for the purpose of Reducing the dimensionality of the data inside of the dataset.

3 EXPERIMENTAL EVALUATION

3.1 Experimental Evaluation 3.1: Random Forest Method

Potential modifications to the existing dataset and task: In this dataset, we are only removing the ID attribute from this dataset because ID only represents the observation and will be meaningless for predicting the diagnosis. In data cleaning and integration: I remove ID because it is unrelated to our diagnosis, ID is only to help us to identify each observation. In data normalization: I do the standard Z-score normalization to remove effects that are different across features. In data reduction: I use PCA to reduce the dimension of the dataset. I will discuss in the discussion section whether it is good or not to use PCA data reduction. In sampling: sample random sampling without replacement. In random forest, I will use the bootstrap method to get samples from the training set. Firstly, I need to do data normalization and PCA to reduce the dimension penalty and computation cost. Then using Random Forest Classifier classify that data point into one of two classes of diagnosis. I will find Information Gain and use bootstrap to construct a random forest. I will calculate F1, precision, and recall seeing how well to fit a model to get a correct diagnosis in prediction by this method. I also need to tune some hyper-parameters in Random Forest Classifier model in order to get the lowest Recall and highest F1. In this step, I will use 3-fold Cross Validation to find a good hyper-parameter in Random Forest Classifier model.

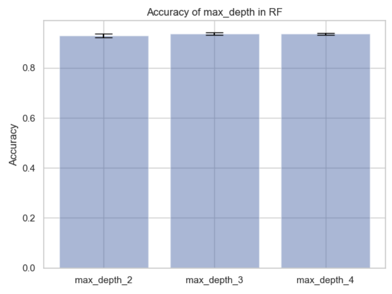


Figure 2: random forest max depth

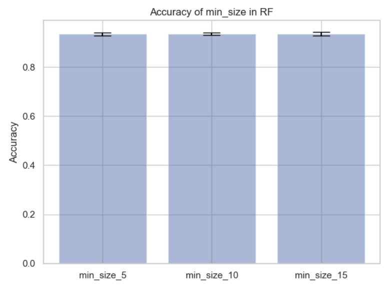


Figure 3: random forest min size

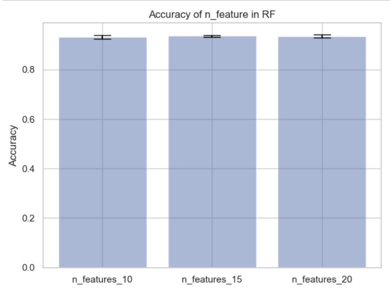


Figure 4: random forest n features

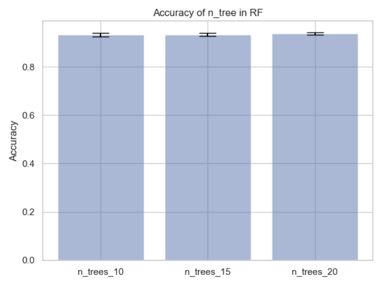


Figure 5: random forest n trees

In this model, I need to tun 4 variables which are the number of trees(n_trees:5,10,20), the depth of trees(max_depth:2,3,4), the minimum size of the leaf(min_size:5,10,15), and the number of features (n_features:10,15,20). Cross-validation is 3 folds (k=3) and the proportion of samples we used in training is 50% (sample_frac=0.5) is fixed. I will not discuss sample_frac and the number of CV folds in tuning. We can find the best average accuracy in 3-fold CV is 0.906 and its parameter are max_depth=2, min_size=5, n_features=20, and n_trees=5. (more details in my code to find good hyper-parameter) Apparently, I can find the max_depth, n_features, and n_trees larger which is better. Since it has higher accuracy and lower variance from the error bar plot (Figure 1-4). Those parameters may not be a good choice, since I only try 3*3*3*3=81 ways of tuning, but it takes a very long running time to see the result for my computer.

3.2 Experimental Evaluation 3.2: MLP Method

In this part, we evaluated the MLP model with the best hyperparameters we selected by Grid Search and Bayesian Optimization. The best hyperparameters are:

hidden layer sizes	(20,15,10)
activation	identity
optimizer	adam
learning rate	0.001

We evaluated the MLP model by F1, accuracy and recall, as showing in figure6 9:

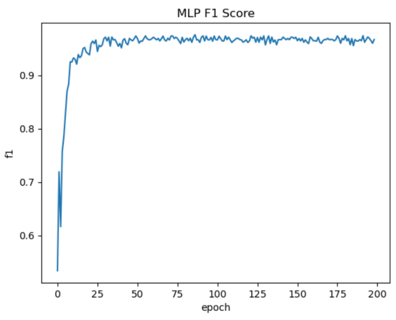


Figure 6: MLP F1 score

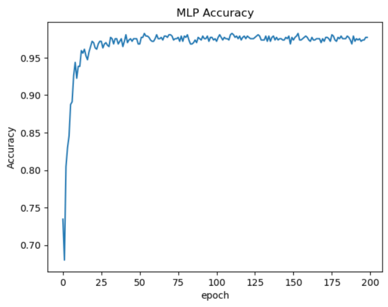


Figure 7: Accuracy

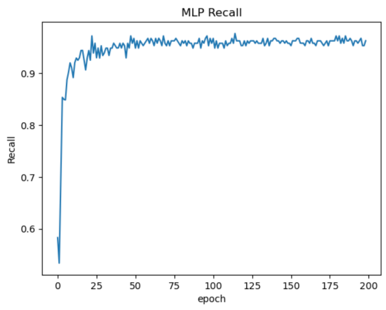


Figure 8: MLP Recall

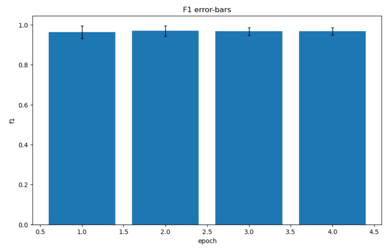


Figure 9: MLP F1 error-bars

3.3 K-nearest Neighbors Method

After tuning the hyperparamter, we evaluate the result with four scores: accuracy, f1, precision and recall. Figure 10 shows the evaluation score for accuracy, f1, precision and recall. In general, the original features give the best performance. Between Euclidean and Manhattan distance function, the difference is not significant; high value SVD features perform better than low value SVD features, since high value SVD contain more information of the original features, it is reasonable, and both of their performance are close the original features; the MLP methods have the worst performance compare to all the other methods, the gap between 5% features and 20% features is also huge. Table 2 shows the detail scores.

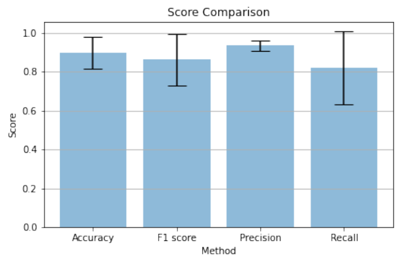


Figure 10: KNN evaluation score

Table 2: KNN evaluation score

Methods	Accurracy	F1	Precision	Recall
Euclidean+Original	0.952128	0.943396	0.961538	0.925926
Manhattan+Original	0.962766	0.956522	0.962500	0.950617
Euclidean+Low SVD	0.930851	0.918239	0.935897	0.901235
Euclidean+High SVD	0.946809	0.936709	0.961039	0.913580
Euclidean+5% MLP	0.750000	0.624000	0.886364	0.481481
Euclidean+20% MLP	0.925532	0.913580	0.913580	0.913580
Manhattan+Low SVD	0.930851	0.918239	0.935897	0.901235
Manhattan+High SVD	0.952128	0.944099	0.950000	0.938272
Manhattan+5% MLP	0.728723	0.571429	0.894737	0.419753
Manhattan+20% MLP	0.920213	0.903226	0.945946	0.864198

3.4 3.4 Experimental Evaluation DBSCAN Clustering

There are two hyperparameters in DBSCAN model: One is 'eps', which is the maximum distance between two samples that can be considered as the neighborhoods. Another hypermarameter is called 'minsamples', which is the number of samples in a neighborhood for a point to be considered as a core point. In our case, we set 'eps' to 1.25 and 'minsamples' to 2.

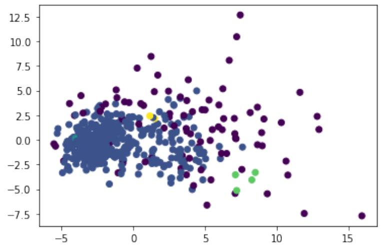


Figure 11: Visualization of DBSCAN clusters

We can see most of points are concentrated on the coordinates around the origin, which can be considered as the class 'B', which points away from the origin, which is labeled as the purple dots, are the class 'M'.

From the graph below, the plot shows that when minsamples increases, the evaluation metric normalized mutual information increases and then decline slightly, when minimum samples reach to 6.

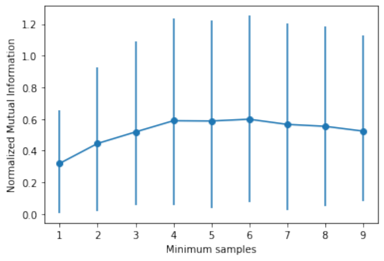


Figure 12: Error bar of normalized mutual information with different min_samples value in DBSCAN

3.5 Spectral Clustering

We performs 10-fold cross-validation for evaluating the performance of Spectral Clustering. For each fold, we split the data into training and test sets, trains the clustering algorithms on the training set, and evaluates the performance of the algorithms on the training set using two metrics: normalized mutual information (NMI) and silhouette coefficient. The NMI measures the similarity between the true labels and the predicted labels, while the silhouette coefficient measures the quality of clustering. We computes the mean and standard deviation of the NMI and silhouette coefficient across the five folds for both clustering algorithms and prints the results, shown as follows:

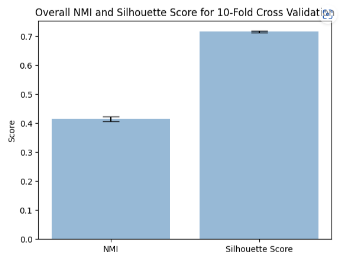


Figure 13: Overall NMI and sihouette score for cross-validation

Table 3: Score table

NMI	Silhouette scores
0.41331503910265877 ±0.014500177045525956	0.7148508268613021 ±0.0023961860646361265

The scatter plot with the first two eigenvectors as the x and y axes provided a clear visualization of the clusters in the data.

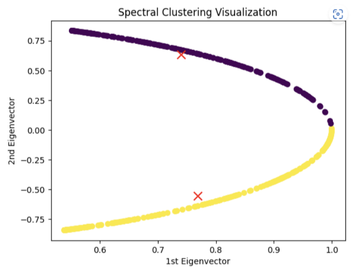


Figure 14: Spectral clustering visualization

3.6 Agglomerative Clustering with Single Linkage

The distance metric and number of clusters are two of the hyperparameters in agglomerative clustering with single-linkage. Agglomerative clustering with single-linkage start with treating each data point as a singleton cluster, the process of preprocessing and normalization of the dataset becomes a necessary step, the data, as the scatterplot below, the dataset after preprocessing become easier to recognize and able to calculate the distance between clusters.

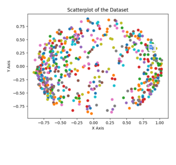


Figure 15: Scatterplot of the dataset

The dendrogram below is the dendrogram of single linkage agglomerative clustering where all same integer number has been put inside of one cluster, this shows the visualized result of single linkage agglomerative clustering after it merges all the clusters into one cluster.

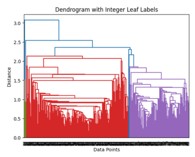


Figure 16: Dendrogram for Agglomerative Clustering with Single Linkage

4 DISCUSSION & CONCLUSIONS

4.1 Random Forest Method

In cancer data, we can find the precision is 0.971, recall is 0.867, F1 is 0.916 and accuracy is 0.889 when we only have 5 trees and 20 features and the training sample proportion is 0.5. We also want to see if data reduction by PCA, how does the resulting change? Remember, We have reduced the dimension from 30 to 10 which has around 95% in the variation. I have created a PCA data called data_pca and run it in our random forest model. Since it only has 10 dimensions of features, we cannot choose 20 for n_features. Hence I want to use n_feature=10 instead to compare results with data reduction and without data reduction in the random forest model.

Table 4: Performance of PCA and Original data in random forest table

Model	Precision	Recall	F1	Accuracy
rf_data	0.971	0.867	0.916	0.889
rf_data 10	0.986	0.865	0.921	0.894
rf_pca 10	0.978	0.774	0.863	0.806

Comparing the two dataset results, I can find the PCA does not have a better result than the original dataset. PCA has a smaller precision which is 97% and the original dataset has 98% precision. It has good precision to predict a result that is benign, but there is a very bad recall which is only 77%. which means if we predict 100 patients are benign, only 77 patients are really benign and 23 patients are malignant but we mark them are benign. When we find PCA's F1, which is 0.86 smaller than 0.92. Hence, PCA is not well. I think one of the reasons is that, when we use PCA to do data reduction, some information is lost and it will reduce the model's interpretation.

4.2 MLP Method

After systematically optimizing the hyperparameters of out MLP model, we got a good MLP model with best F1 score 0.9733918104961781. The hyperparameter optimization through Bayesian optimization and grid search has greatly improved the performance of the MLP model. It shows the importance of the selection of super parameters to MLP model.

4.3 K-nearest Neighbors Method

From the results in previous section, k-nearest neighbors method has an average performance in this dataset. During the implementing, the most difficult part for me is to figure out how MLP works in this method, I am still not sure if my implementation is correct, further research is necessary for me to understand this better.

4.4 Discussion & Conclusions 5.4: DBSCAN Clustering

Finally, we can see that DBSCAN its ability to identify clusters of any shape and handle datasets with varying densities. Unlike other clustering algorithms, DBSCAN can find clusters that are not necessarily spherical or have similar sizes.

In summary, DBSCAN is advantageous because it can handle datasets with varying densities, identify clusters of any shape, handle noise and outliers, and is efficient and scalable for large datasets.

4.5 Spectral Clustering

In conclusion, spectral clustering is a powerful technique for clustering data points in a low-dimensional space. Our experimental results demonstrate the effectiveness of this technique on breast cancer and artificial datasets. Our findings suggest that spectral clustering can be a valuable tool for clustering tasks in machine learning and data analysis.

Future research directions include the investigation of different similarity matrices and the use of different algorithms for spectral clustering. Also, the development of scalable spectral clustering algorithms for large datasets can be an interesting research direction. Finally, the investigation of the performance of spectral clustering in unsupervised learning tasks, such as anomaly detection and outlier detection, can also be an interesting research direction. The intrinsic evaluation and Agglomerative Clustering with Single Linkage we cannot produce errorbars since it is a deterministic algorithm

4.6 Discussion & Conclusions 5.6: Agglomerative Clustering with Single Linkage

The average Silhouette Coefficient for data points and the NMI for the data set are both not in the "correct" range, and the result with k-mean cross-validation of the Silhouette Coefficient and NMI is also not in the "correct" range, therefore cross-validation does not really work with cross-validation. Agglomerative Clustering with Single Linkage is a deterministic algorithm, which means it cannot calculate the accuracy of the clustering result, and it cannot produce error bars.

In conclusion, Agglomerative Clustering with Single Linkage can be powerful, but it can also be limited by the data set since it needs to calculate the minimum distance between data points, if there are data that far away from other clusters, the efficiency of the clustering result may be influenced by these data points, which means it can be sensitive to noise and outlier data.

5 IMPLEMENTATION CORRECTNESS REPORT

5.1 Random Forest Method

For the decision tree:
(1) In Implementation correctness we have 17 data points and 2 features. In one decision tree method, the tree selects feature one firstly and then selects feature two second. The reason we choose the order like that is that in feature has a larger information gain which will be chosen first.
(2) The feature split value is chosen which is 8.5 and the second feature split value is 7.405. Those values are chosen because when we split this threshold value, we will get the larger information gain. This is the reason we create the find_best_split function to find the split value and feature which has the largest information gain.
For random forest:
clarify: By LabelEncoder(), label transformation into 1 or 0, 1 represents class 2, and 0 represents class 1
(1) In the random forest method, basically it is very similar to the decision tree. When we create a tree, we will use a decision tree to create tree. Here I use n_trees=10 which means I create 10 trees and run build_tree 10 times and split the data 10 times and then the random forest will collect each tree's final result to make the decision for prediction. In the example below, I have 10 trees with all same result which is 0 (class 1). Finally, random forest predicts it as class 1.
(2) The hyperparameter are n_trees=20, n_features=2, max_depth=2, and min_size=2. In 20 trees, most of the predictions are class 1 when we input [4,4]. Only 4 predictions are class 2. It is a reasonable result for the split. Because when the tree finds the best feature and split from this dataset, feature 1 has many more than feature 2. When we create a threshold value and make a split. Feature 1 has a larger region than feature 2. [4,4] is easy to drop in the feature 1 area.
Without using PCA, the random forest method has good performance to make a prediction but it has a lower recall comparing its precision. I think I can improve my random forest function in the future to improve its recall and accuracy. Comparing the sklearn.ensemble's function RandomForestClassifier, we get label 0 which means [4,4] is class 1. Our implement's result is the same as the sklearn.ensemble's function result.

5.2 Implementation Correctness Report 6.2: MLP Method

6.2.1 MLP-Grid

Use sklearn's library functions GridSearchCV to implement Grid search, The selection range of hyperparameters is as follows:

hidden layer sizes	(20,20,10),(20,15,10),(20,15),(15,10),(10)
activation	identity, logistic, relu
optimizer	sgd,adam
learning rate	0.001,0.0001

Within the preset range of hyperparameters, the hyperparameters are traversed, and we use ten fold cross validation to train and test the model. The final classification results are obtained in turn, and the best parameters are selected by comparison of f1 score.The best F1 score we got is 0.9733918104961781, the best hyperparameters are showing below:

hidden layer sizes	(20,15,10)
activation	identity
optimizer	adam
learning rate	0.001

For different hyperparameters, we compared there behavior by showing a graph to compare the F1 score of each combination of hyperparameters, as showing in figure17:

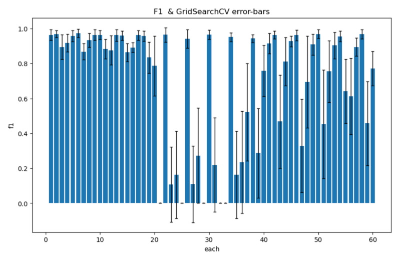


Figure 17: Grid Search F1 error-bars

6.2.2 MLP-BO

Use skopt's library functions BayesSearchCV to implement Grid search, The selection range of hyperparameters are as follows:

hidden layer sizes	(20~10,20~10,20~10),(20~10,20~10),(20~10)
activation	identity, logistic, relu
optimizer	sgd,adam
learning rate	0.001~0.0001

We also use ten fold cross validation to train and test the model. After the process of Bayesian Optimization, the best parameters are selected by comparison of f1 score. The best F1 score we got is 0.9807146908678389, and the best hyperparameters are showing below:

hidden layer sizes	(10,15,15)
activation	identity
optimizer	adam
learning rate	0.001

For different hyperparameters, we compared there behavior by showing a graph to compare the F1 score of each combination of hyperparameters, as showing in figure18:

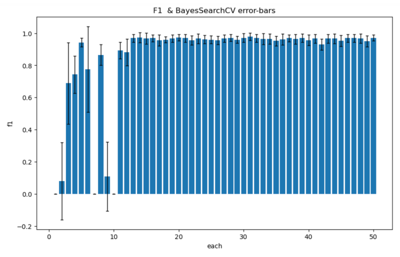


Figure 18: Bayesian Optimization F1 error-bars

6.2.3 MLP-Funnel

We build a "funnel" shape MLP with the width of its hidden layers progressively reduces the dimensionality of the data. In order to identify the best such MLP, we use the same grid search process as above. After the process of grid search, we got a "funnel" shape MLP with hidden-layer-sizes: (40, 30, 15); activation function: relu; learning-rate: 0.001; optimizer: adam. And it's best f1 is 0.9741455761733697.

6.2.4 Low-rank SVD

As we can see in figure19 that when the number of principal components is less than 7, the singular values drop sharply in terms of order of magnitude. Therefore, we reduce the dimension of data to 7. We input the data after dimensionality reduction into the two MLP modes obtained in the previous section, which are MLP-BO and MLP-grid.

Model	MLP_Grid	MLP_BO	MLP_Funnel
F1	0.959439	0.964425	0.974145

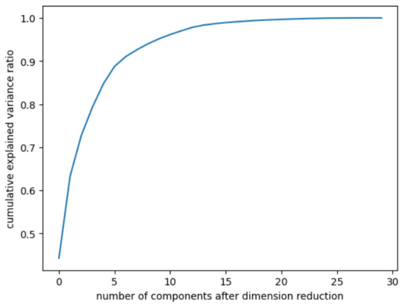


Figure 19: select thE rank of the low-rank approximation

As we can see from the table, dimensionality reduction within the MLP performs slightly better than dimensionality reduction as pre-processing.

5.3 K-nearest Neighbors Method

For the correctness test, I build a test mode in the original function, we use two different distance function to test the artifical dataset. The test data point is classified as class 1 based on Euclidean distance. It is classified as class 2 based on Manhattan distance.

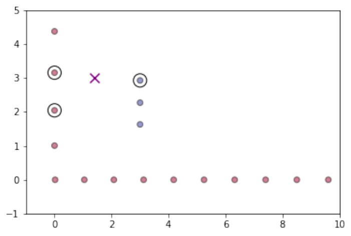


Figure 20: Euclidean distance

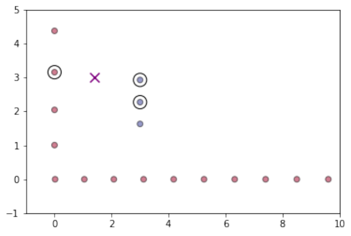


Figure 21: Manhattan distance

5.4 Implementation Correctness Report 6.4: DBSCAN Clustering

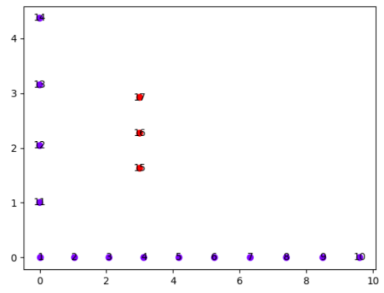
A core point is a point that has at least minpts points in its neighborhood, including itself. This means that a core point has a relatively dense neighborhood, and can potentially form a dense cluster.

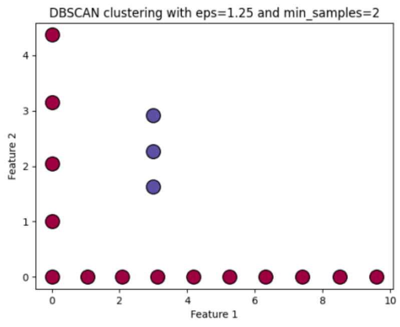
A non-core point, on the other hand, has fewer than minpts points in its neighborhood. Non-core points may still belong to a cluster, but they cannot form their own cluster. Instead, they depend on core points to form a cluster.

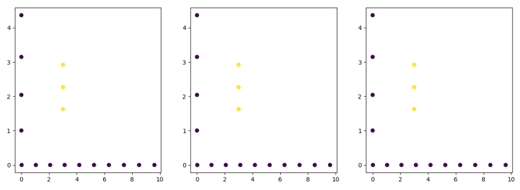
In our case, we have set minpts=2 and eps=1.25. Looking at the dataset, we can see that the first 10 points all have a neighborhood of at least two points (including themselves), and are therefore core points. The remaining 8 points have only one point in their neighborhood, and are non-core points.

We can denote the core points with filled circles and non-core points with empty circles to differentiate between them in the scatterplot.

From the graph blew, we can see that when we use the artificial data, we have the correct result.







5.5 Implementation Correctness Report 6.5: spectral clustering

1 and 2: Here is the Embedding Latex Table

Table 5: Embedding Latex Table

-0.73121772	-0.68214415
-0.82105689	-0.57084638
-0.98628014	-0.16508022
-0.99872771	-0.05042784
-0.93661662	0.35035598
-0.78444938	0.62019286
-0.60921766	0.79300305
-0.56858725	0.82262296
-0.53582777	0.8443273
-0.5108038	0.85969732
-0.70647569	-0.70773731
-0.681711	-0.73162156
-0.65723577	-0.75368505
-0.56517265	-0.82497265
-0.98454139	-0.1751521
-0.98454139	-0.1751521
-0.9607742	-0.2773318

3:Spectral embeddings can provide benefits over clustering the original datapoints directly in certain cases. Spectral embeddings are a type of dimensionality reduction technique that transforms high-dimensional data into a lower-dimensional space while preserving the relationships among the data points. The lower-dimensional space is often referred to as the spectral space. The main benefits of spectral embeddings is that they can help to overcome some of the limitations of clustering algorithms that work directly on the original high-dimensional data. High-dimensional data can be noisy, sparse, and complex, which can make clustering difficult. Spectral embeddings can reduce the dimensionality of the data, which can help to overcome some of these issues by highlighting the most important features of the data. Another advantage of spectral embeddings is that they can be used to uncover underlying patterns or structures in the data that may not be apparent in the original high-dimensional space. By mapping the data to a lower-dimensional space, spectral embeddings can help to identify clusters that may be more difficult to identify in the original space.

4: The objective function of k-means clustering is to minimize the sum of squared distances between each data point and its assigned cluster center (also known as centroid). In other words, the goal of k-means is to partition a given dataset into k clusters, where each data point belongs to the cluster whose centroid is closest to it. It is shown as follows:

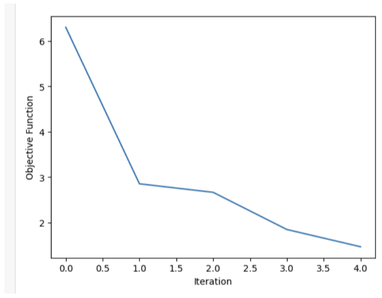


Figure 22: K-mean plot

5: The scatterplot is shown as follows:

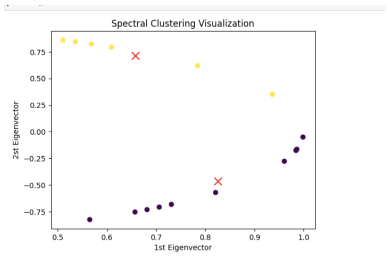


Figure 23: Spectral clustering scatterplot

5.6 Implementation Correctness Report 6.6: Agglomerative Clustering with Single Linkage

Agglomerative Clustering with Single Linkage is used for clustering rather than classification, so all the data points will be merged inside one cluster at the end. The scatterplot below shows the beginning cluster for the artificial dataset, each point has a unique integer number on top of it.

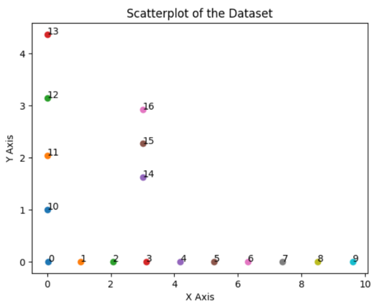


Figure 24: Scatterplot of the dataset

The dendrogram below is the dendrogram of single linkage agglomerative clustering for the artificial dataset where all same integer number has been put inside of one cluster, this dendrogram shows the visualized result of single linkage agglomerative clustering after it merges all the clusters into one cluster. Comparing the dendrogram from real-world data, we can see that all the distances between the clusters is more uniform, as can be seen from the above figure.

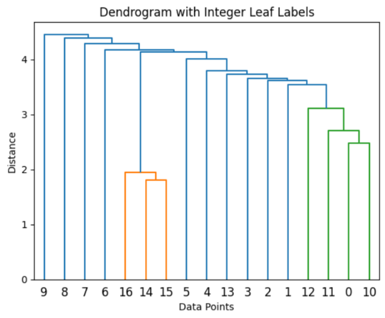


Figure 25: Dendrogram for Agglomerative Clustering with Single Linkage

Theoretically, if the distance metric and link criteria are properly selected, and if the number of clusters is correctly specified, it is possible to obtain a cluster that is the same as the cluster displayed in the data set with single-linkage. However, this is not always possible, especially with datasets that have 13 attributes with hundreds of points of data. To obtain the same cluster as the ground-truth clusters, one method is to set the number of clusters to the number of real clusters in the data, use the distance measurement reflecting the underlying structure of the data, and the selection of linkage criteria will also affect the clustering results.

6 REFERENCE

[1] Dadecic, D. (2021, July 5). Master Machine Learning: Random Forest from Scratch with Python. Towards Data Science. https://towardsdatascience.com/master-machine-learning-random-forest-from-scratch-with-python-3efdd51b6d7a

[2] Brownlee, J. (2019, March 25). How to Implement Random Forest From Scratch in Python. Machine Learning Mastery. https://machinelearningmastery.com/implement-random-forest-scratch-python/

[3]Ng A, Jordan M, Weiss Y. On spectral clustering: Analysis and an algorithm[J]. Advances in neural information processing systems, 2001, 14. https://proceedings.neurips.cc/paper/2001/file/801272ee79cfde7fa5960571fee36b9b-Paper.pdf

[4]Patrick, L. (2019, Sep 03). KNN (K Nearest Neighbors) in Python - ML From Scratch 01. https://www.python-engineer.com/courses/mlfromscratch/01_knn/

[5] Lloyd S. Least squares quantization in PCM[J]. IEEE transactions on information theory, 1982, 28(2): 129-137.http://mlsp.cs.cmu.edu/courses/fall2010/class14/lloyd.pdf

[6] Chen X, Cai D. Large scale spectral clustering with landmark-based representation[C]//Twenty-fifth AAAI conference on artificial intelligence. 2011. https://ojs.aaai.org/index. php/AAAI/article/view