CS235 Winter'23 Project Midterm Report: Default Project

MAMADOU ZERBO, NetID: mzerb001
JASON SADLER, NetID: jsadl003
KANAK DAS, NetID: kdas006
SRINIVASA BIRADAVOLU, NetID: sbira001
AUSTIN LEE, NetID: alee235
AVINASH LAKHMAWAD, NetID: alakh003

Additional Key Words and Phrases: data, mining

ACM Reference Format:
Mamadou Zerbo, Jason Sadler, Kanak Das, Srinivasa Biradavolu, Austin Lee, and Avinash Lakhmawad . 2023. CS235 Winter'23 Project Midterm Report: Default Project. 1, 1 (March 2023), 13 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 BRIEF SUMMARY OF THE STATE OF THE PROJECT

2 PRELIMINARY RESULTS

In this section, you should present results from each method that was proposed, while using existing library/off-the-shelf implementations. Please follow the designated pattern below for all methods (while renaming "Method X" with the actual name of the method. For larger teams, this may exceed 5 pages (which is OK, but please exercise your judgement in presenting your results in a concise manner).

2.1 Preliminary results:Random Forest Classifier (Mamadou Zerbo 862016649)

(1) **Off-the-shelf baseline**: The method used is the RandomForestClassifier from the sklearn.ensemble module of the scikit-learn library. The method consist of multiple decision trees, and each tree makes a prediction for a given data point. A final decision is made by taking the average or the mode of all the predictions made. RandomForestClassifier is instanciated with two arguments that are n estimator which specifies the number of trees in the forest, and random state, which sets the seed for the random number generator. Here is the source link of the method:
   - RandomForestClassifier

(2) **Potential modifications to the existing dataset and task**: nothing to report.

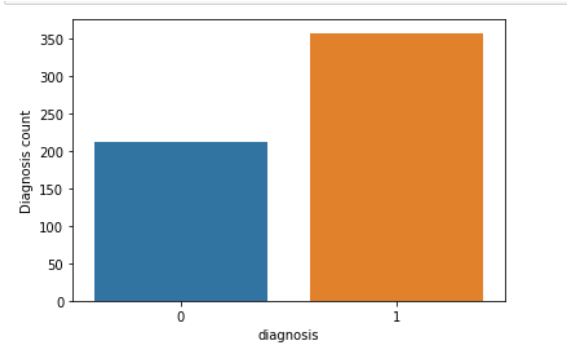
(3) **Preprocessing**: The preprocessing step involved transforming the categorical values of the diagnosis column of the dataset into numerical values using the mapping M = 1 and B = 0. This was done using the replace

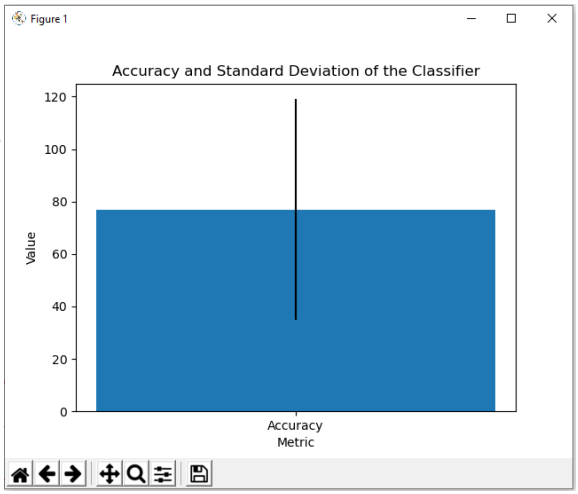
Authors' addresses: Mamadou ZerboNetID: mzerb001; Jason SadlerNetID: jsadl003; Kanak DasNetID: kdas006; Srinivasa BiradavoluNetID: sbira001; Austin Lee NetID: alee235; Avinash Lakhmawad NetID: alakh003.

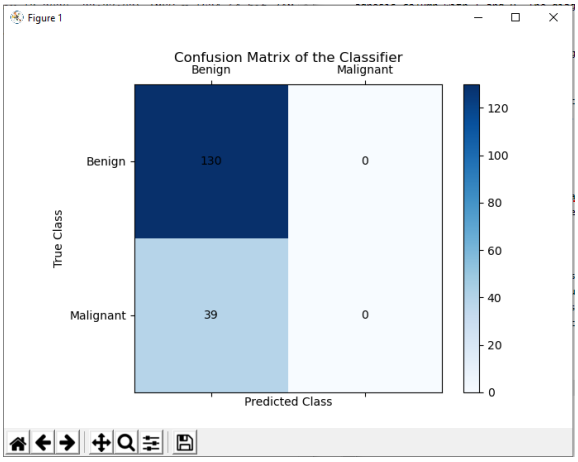
method in pandas. Then the data is split into a training set and a test set using the train test split function from scikit-learn's model selection module. There was no normalization, feature selection, representation learning, or sampling/sketching on the data performed.

(4) **Experimental setting**: The Wisconsin breast cancer dataset was used to evaluate the experiment. The data consists of 569 samples of malignant and benign tumor cells, has 30 features. The diagnosis column classifies the cells as malignant (M) or benign (B). Here are the steps that were used to set up the experimental evaluation:
a) First, the dataset is preprocessed by replacing the 'M' and 'B' in the diagnosis column with 1 and 0. The diagnosis column is converted into a numerical format that could be used as the dependent variable in the machine learning model.
b) Next the dataset is split into two parts, a training set and a testing set, using a 70/30 ratio. The training set was used to fit the model and the testing set was used to evaluate the performance of the model.
c) Next the scikit-learn library is used to implement the Random Forest Classifier. The model is trained using the training set. Three metrics were used to evaluate the performance of the model: accuracy, precision, and recall. Accuracy is the proportion of correct predictions made by the model. Precision is the proportion of positive predictions that were actually positive. Recall is the proportion of positive actual values that were correctly predicted by the model.
d) In the next step, the results were computed using 10-fold cross-validation, and that involved dividing the dataset into 10 parts and training the model on 9 parts while evaluating the performance on 1 part. The process was repeated 10 times so that each part of the dataset is used once as the evaluation set.
e) Finally, the results were then averaged to obtain a final score for each metric.

(5) **Results**: On the "Accuracy and Standard Deviation of the Classifier" image, the single bar represents the the accuracy of the classifier on the test data, and the error bar represents the Standard deviation of the accuracy. The horizontal axis of the bar graph represents the metric being plotted, which is "Accuracy". The vertical axis represents the value of the metric, which is the accuracy of the classifier in percentage. The error bar shows the deviation from the mean accuracy value.
The confusion matrix evaluates the performance of the classification model







2.2 Preliminary results: Multi-Layer Perceptron (MLP) (Jason Sadler 862128623)

(1) **Off-the-shelf baseline**: The baseline classifier used is scikit-learn's Multi-layer Perceptron classifier. For evaluation scikit-learn's cross-validation is used.

The following links are used for reference.

- Cross-Validation
- MLPClassifier

Table 2.2.1 shows the settings *MLPClassifier* which have an impact on the classification results. Values in green are modified, black are default from the base implementation of the class. The *MLPClassifier* has one hidden layer of size 100. The green values were changed through educational experimentation with the class and are not indicative of optimization effort.

(2) **Potential modifications to the existing dataset and task**: See section 2.1.3.

(3) **Preprocessing**: No prepossessing of the dataset was performed.

MLPClassifier Parameters			
Parameter	Value	Parameter	Value
hidden_layer_sizes	102	max_iter	500
activation	logistic	shuffle	False
solver	sgd	random_state	None
alpha	0.0001	tol	1e-4
batch_size	auto	momentum	0
learning_rate	constant	n_iter_no_change	10
learning_rate_init	0.001	early_stopping	False

Table 2.2.1. MLPClassifier Parameters

(4) **Experimental setting**: K-fold cross-validation is used to evaluate the classifier. For each fold the accuracy, recall, precision, F1, and ROC AUC are recorded. Ten folds are used. Since the MLP is highly non-convex with the weights being randomized, to get a reasonable estimation cross-validation is ran ten times and the average of each fold is used for evaluation.

(5) **Results**: Figure 2.2.1 shows the score results for the classifier for each fold averaged across the ten samples and the average of the averages. Accuracy averaged at 89%. Precision remained quite high across all folds with a 98% average. The classifier does suffer from relatively low recall averaging 72% across the folds. This seems to imply that if the classifier sees malignant cancer its really good at correctly identifying the cancer, there seems to be very little false positives. However, it also misses a lot of malignant cases. The classifier has learned a strict criteria for what it thinks malignant cancer is. The ROC curve remained quite high across all folds, averaging 95%.

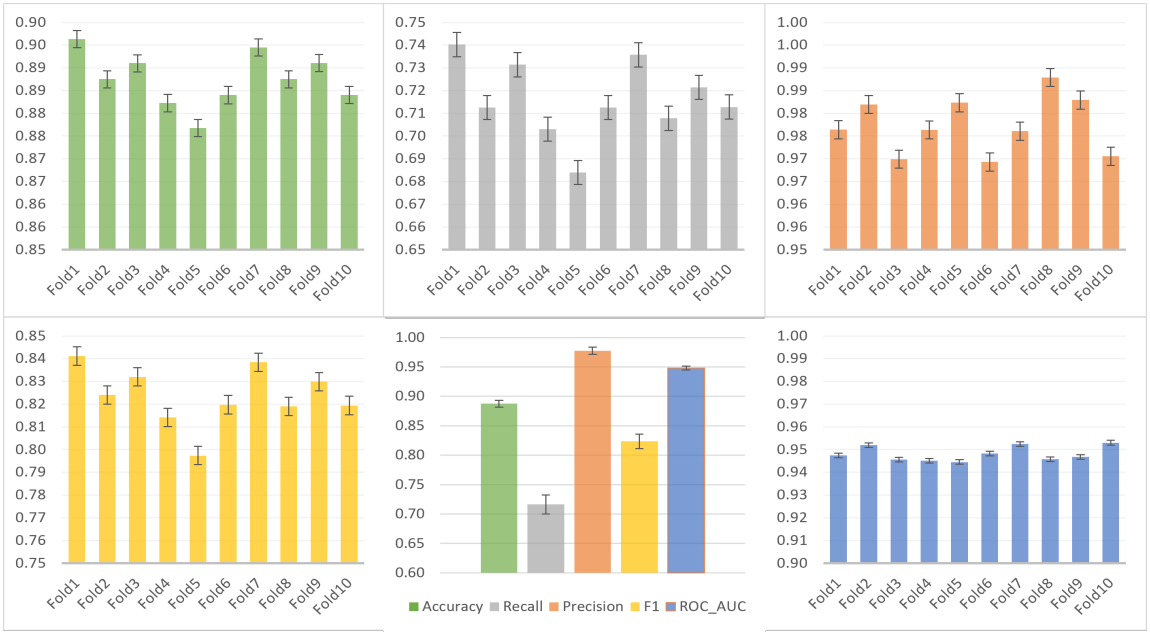


Fig. 2.2.1. Cross-Validation Metrics

2.3 Preliminary results: K-nearest neighbors (Kanak Das 862393193)

(1) **Off-the-shelf baseline**: The baseline classifier used here is *sklearn.neighbors.KNeighborsClassifier* from scikit-learn. The number of neighbors used in this experiment is 5 for the classifier. The source of the used methods are,
   (a) SciKit Learn KNN,
   (b) SciKit Learn Cross Validation.

(2) **Potential modifications to the existing dataset and task**: Nothing to report.

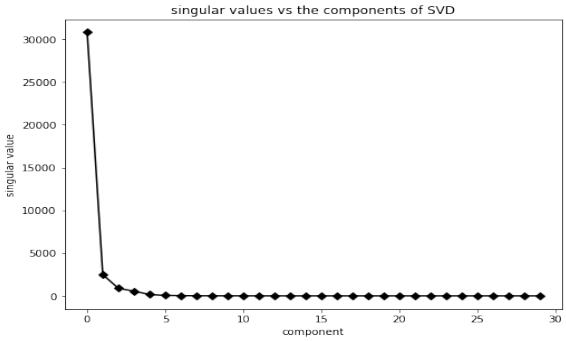
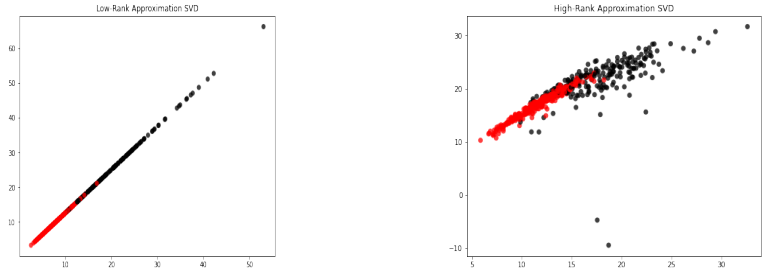


Fig. 2. Semi-Logarithmic plot for Singular Values



(a) 2-d scatter plot for two principle components of approximated low-value rank of 1 using SVD

(b) 2-d scatter plot for two principle components of approximated high-value rank of 3 using SVD

Fig. 3. SVD-based feature representation

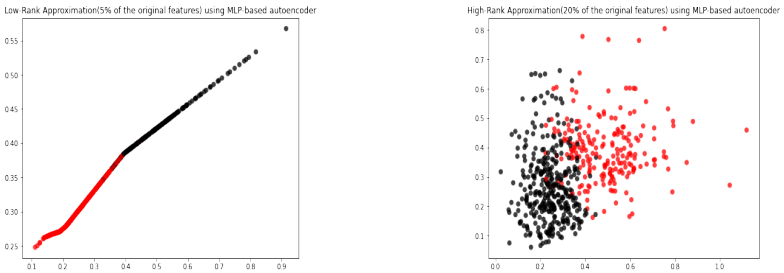
(3) **Preprocessing**: The columns *id* and *Unnamed: 32* have been dropped from the dataset to process only the 30 columns representing the actual relevant data. Further, two different classes of feature representations have been performed.

The first one is two low-rank approximations using Singular Value Decomposition(SVD) based on singular value drops observed from Figure 2. The most dramatic drop in singular values is right after the first component. The two low-rank representations have been created with a low value of 1 and a high value of 3. Figure 3a and Figure 3b presents the first two components in 2-d scatter plots for the two low-rank representations.

The second is MLP-based AutoEncoder, which has been used to represent 5% and 20% of the features. Figure 4a and Figure 4b present 2-d scatter plots for the two representations.

For lower rank approximation, SVD has been performed using *scipy.linalg.svd* from SciPy. For MLP-based Autoencoder, *tensorflow.keras.layers* and *tensorflow.keras.models* have been used. The source to the methods are listed below,

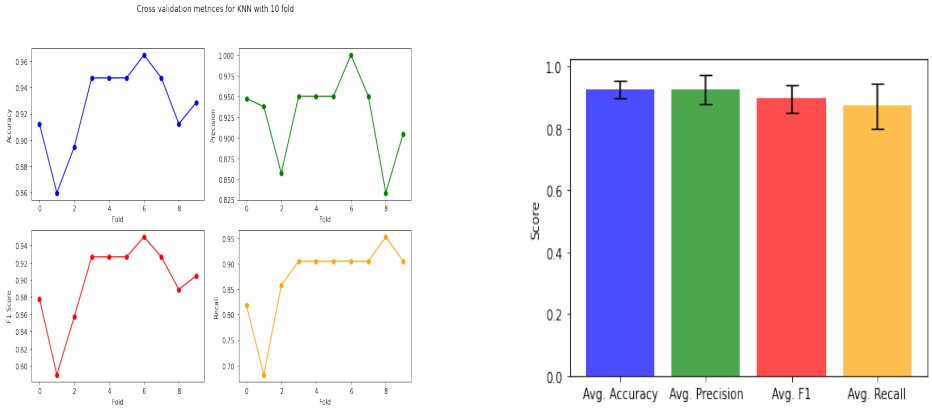
(a) SciPy SVD
(b) Keras Model
(c) Keras Layers



(a) 2-d scatter plot for low rank approximation (5% of original features) using using MLP-based AutoEncoder

(b) 2-d scatter plot for high rank approximation (20% of original features) using using MLP-based AutoEncoder

Fig. 4. AutoEncoder based feature representation



(a) Accuracy, Precision, F1 Score, and Recall of KNN for cross-validation with 10 folds

(b) Average Accuracy, Precision, F1 Score, and Recall with error-bars of KNN for cross-validation with 10 folds

Fig. 5. Evaluation metrics for KNN

(4) **Experimental setting**: Off-the-shelf KNN classifier from SciKit Learn with 5 neighbors was used for the classification task. The Wisconsin Breast Cancer dataset was split into 80% train data and 20% test data. Then 10-fold cross-validation was used to evaluate the classifier. Then Accuracy, Precision, F1 Score, and Recall were calculated for each of the folds.

(5) **Results**: The subfigures in figure 5a plots values for Accuracy, Precision, F1 Score, and Recall in y-axis for each of the 10 folds used for evaluation. Figure 5b plots the average of each of the metrics for the 10 folds along with error bars.

Additionally, two distance functions were implemented to calculate Euclidean and Manhattan distances for using in the later part of the project.

2.4 Preliminary results: DBScan (Srinivasa Biradavolu 862318458)

Please include student name and SID alongside the method. Describe the following:

(1) **Off-the-shelf baseline**:
   (a) The baseline classifier used here is sklearn.cluster.DBScan from scikit-learn. The below link was used in reference:
   SciKit Learn DBScan
   (b) Performed DBScan using the feature matrix multiple times with the modification of mainly two parameters: *eps* and $min - samples$
(2) **Potential modifications to the existing dataset and task**: Nothing to report
(3) **Preprocessing**:
   (a) Dropped the *id* column as they were numbers irrelevant to the other labels
   (b) Dropped the $Unnamed : 32$ column as it was an empty column that will give unnecessary nan values upon creation of feature matrix
   (c) Dropped the *diagnosis* column as it is the independent variable and we only want to analyze the clusters and try to improve the quality of them to study the structure of the data
   (d) Used Standard Scalar normalization from *sklearn.preprocessing* library to create the feature matrix
(4) **Experimental setting**:
   (a) The two main parameters to change in the algorithm are the *eps* and $min - samples$. Lower *eps* in combination with higher $min-samples$ indicates that we want to cluster data points with higher density. So I started-off by experimenting with the first two variables: $radiusmean$ and $texturemean$ with $min-samples$ set to 5 and *eps* set to 2.
   (b) The goal is to find out an acceptable value of both the parameters based on two evaluation metrics: Silhuoette score and Calinski Harabasz score
   (c) The value of $min - samples$ was selected as the elbow of the curve of the distance plot plotted between points sorted by distance and the distance to the 10th nearest neighbor as seen in fig 6

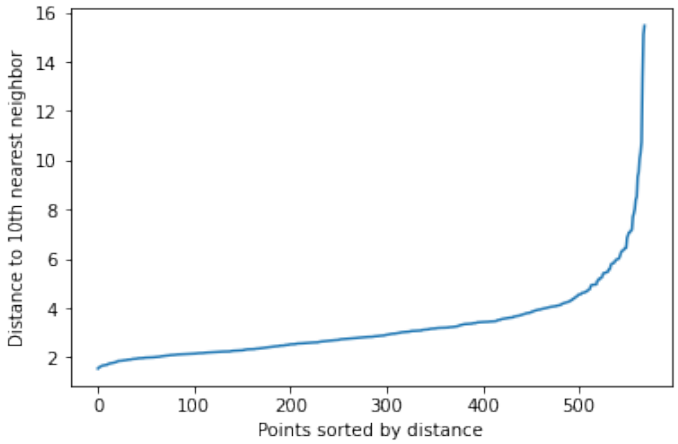


Fig. 6. Distance Plot

(d) The metrics that we use for evaluating the correctness and quality of our clusters will be the Silhouette Coefficient Mean and Calinski Harabasz Score together. We will fix the value of $min_samples$ to 6 and try to look at how the silhoutte and calinski scores vary upon running DBScan ten times with eps starting from 2 incremented by 1 each time. We can see how the values vary in fig 7

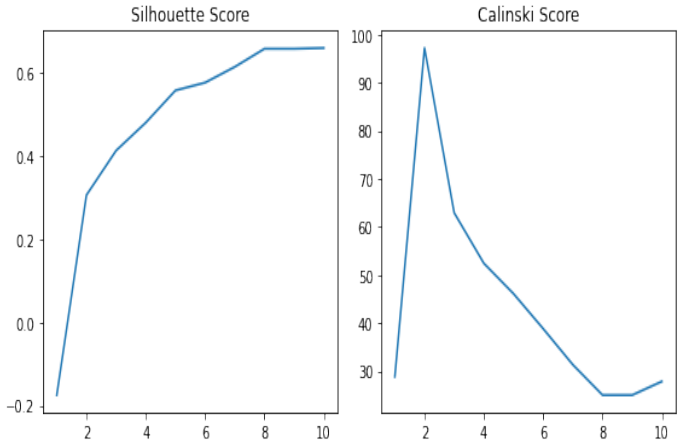


Fig. 7. Silhouette and Calinski Scores Outlook

(e) Now we want our silhouette coefficient to be as close as possible to 1 and our calinski scores as high as possible whose ideal value is considered as 100. But in order to maintain the quality of the cluster, we will select the ideal value where the silhouette coefficient is on the higher side and the calinski score has just begun to drop. $SilhouetteCoefficient = 0.365$ and $Calinski - Harabaszscore = 83.122$ The error bars for $eps = 2.9$ for Radius Mean vs Texture Mean and Perimeter Mean vs Area Mean are given in fig 8

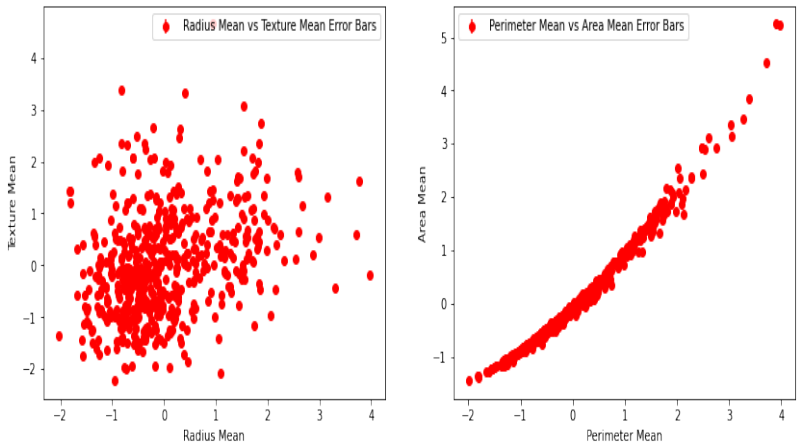


Fig. 8. Errorbars for Four Features

(5) **Results**:

(a) Since in the above errorbar plots are with respect to the standard deviation means, it means that we can say that the quality of clusters given will be consistent. So we now produce outputs of the clusters generated with the labels created with our assumed optimal eps and min_values across radius mean vs texture mean and perimeter mean vs area mean(Shown in fig 9)

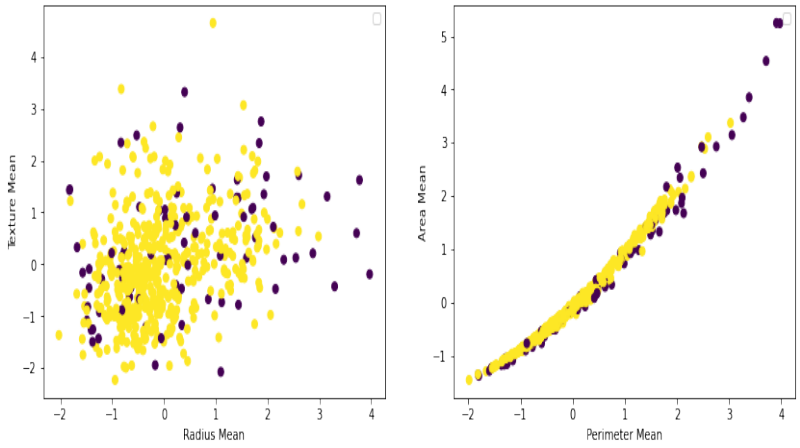


Fig. 9. DBScan Clustering

(b) So for values $eps = 2.9$, $min_samples = 6$, we can say that the clusters produced are consistent. This means that upon looking at fig 9 we can say that the density of the features radius mean vs texture mean lies in about the same areas as Perimeter mean vs Area mean meaning that the importance of a sample being classified as benign or malignant might lies heavy in those areas

2.5 Preliminary results: Spectral Clustering (Austin Lee 862084022)

(a) **Off-the-shelf baseline**: The baseline classifier used here is scikit-learn's Spectral Clustering. The following links are used for reference

(i) SciKit Learn Spectral Clustering

I ran Spectral Clustering twice, one time with a Radial Basis Function affinity matrix and a second time with a Nearest Neighbor affinity matrix. Table 2.5.1 shows the input settings of SpectralClustering. Values in green are modified, while values in black are the default values specified by the library.

Spectral Clustering Parameters			
Parameter	Value	Parameter	Value
n_clusters	2	eigen_solver	None
n_components	none	random_state	None
n_init	1	gamma	1.0
affinity	rbf/nearest_neighbors	n_neighbors	10
eigen_tol	auto	assign_labels	kmeans
degree	3	coef0	1

Table 2.5.1. Spectral Cluster Parameters

(b) **Potential modifications to the existing dataset and task**: Nothing to report.

(c) **Preprocessing**: See section 2.4.3

(d) **Experimental setting**: I evaluated my clusterer by measuring the Silhouette coefficient averaged over all data points and the Normalized Mutual Information Score given the class labels of the dataset. For the class label of the dataset, I used the column "diagnosis" except I transformed the data such that 'M' = 1 and 'B' = 0. I ran my clusterer ten times with random initial centroids. I then compiled all my results in order to create a bar graph showcasing the Silhouette Coefficient and Normalized Mutual Information Score for Spectral Clustering with Radial Basis Function affinity matrix and Spectral Clustering with Nearest Neighbor affinity matrix. Although, I set the parameters assign_labels = ḱmeansánd n_init = 1 which should ensure a random initial centroid, I get very insignificant differences between the different runs. I also want to mention that I tested different cluster sizes ranging from 2-15 and found that a cluster size of 2 gave the best performance so that is the cluster size I chose when displaying the final results for my clusterer. Fig. 10 will display a line graph showing the correlation between the Silhouette Coefficient and Normalized Mutual Score, and the number of clusters.

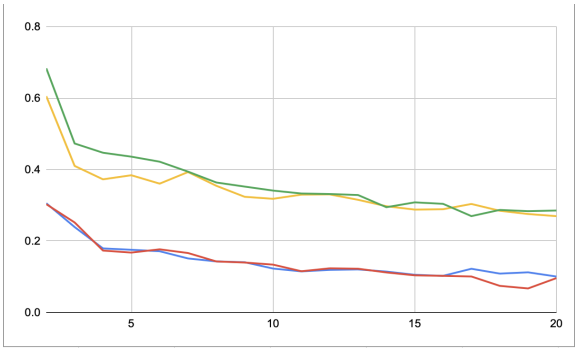


Fig. 10. Line graph of Results as # of cluster increases. Blue = Silhouette RBF, Red = Silhouette NN, Yellow = NMI RBF, Green = NMI NN

(e) **Results**: As you can see in Fig. 12, I achieve a score of $\tilde{0}.3$ with 0 or an insignificant amount of standard deviation. This is a relatively low score which means that the data is not very suitable for Spectral Clustering or my data may require different preprocessing. I achieved a Normalized Mutual Information Score of $\tilde{0}$.6-0.67 with 0 or an insignificant amount of standard deviation. Having a high Normalized Mutual Information score means that there is a high correlation between my clustering and the groundtruth/diagnosis column. Fig. 11 is a bar graph of my results for easy visualization.

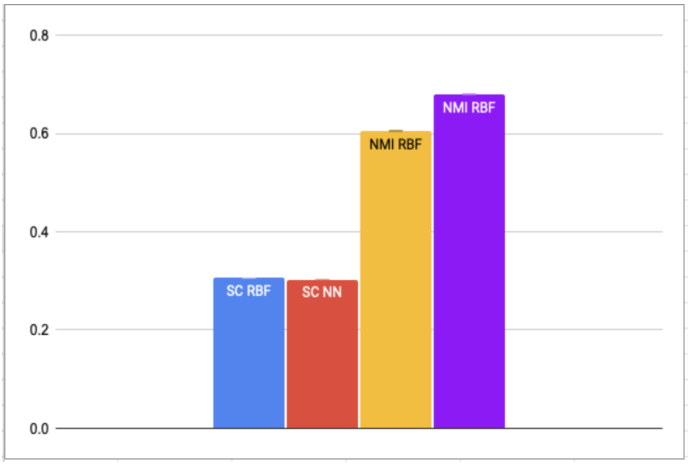


Fig. 11. Bar Graph of Silhouette Coefficient and Normalized Mutual Information Score of Spectral Clustering with Radial Basis Function and Nearest Neighbor

Silhouette Coefficient with Radial Basis Function	0.305617734±0
Silhouette Coefficient with Nearest Neighbor	0.3029471168 ± 0.00005566545357
Normalized Mutual Information Score with Radial Basis Function	0.6051728777 ± 0
Normalized Mutual Information Score with Nearest Neighbor	0.6793804003 ± 0.004105894783

Fig. 12. Table of Silhouette Coefficient and Normalized Mutual Information Score of Spectral Clustering with Radial Basis Function and Nearest Neighbor

2.6 Preliminary results: Agglomerative Clustering with Single Linkage (Avinash Lakhmawad 862254041)

(a) **Off-the-shelf baseline**: This method uses following module from sklearn Agglomerative Clustering This is bottom-up approach, initially each observation is treated as it's own cluster. Linkage used for clustering is single-linkage. It's minimum distance between members of the two clusters. The method automatically forms two clusters. However, user can specify the number of clusters using method n_clusters.

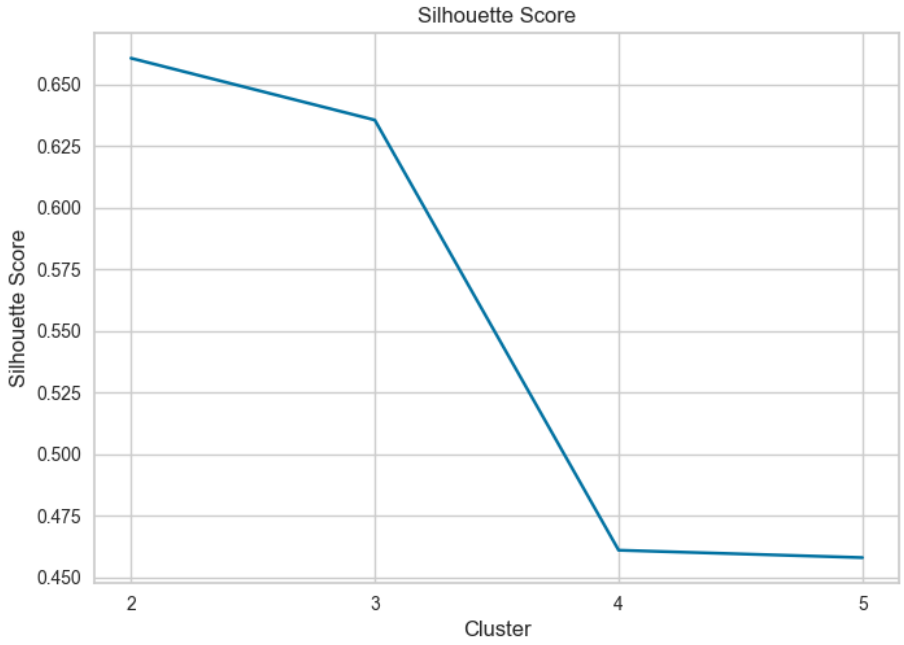
(b) **Potential modifications to the existing dataset and task**: None.

(c) **Preprocessing**:

- Drop the 'ID' column
- Convert the categorical values of 'diagnosis' column to numeric values by mapping B: 0 and M: 1
- Make a copy of the dataframe where 'diagnosis' column is dropped - 'dff'.
- Use the copied dataframe - 'dff' for clustering purpose.

(d) **Evaluation**

- Intrinsic - Calculate the Silhouette score using scikit-learn library function. Distance method used is 'euclidean'. Repeat for different size of clusters 2, 3, 4, 5



- Extrinsic - Calculate the Normalized Mutual Information(NMI) using the predicted column vs the ground truth - the diagnosis column from dataset. Standard library function is used for this calculation as well.