Machine Learning

CS60050
Assignment-2 Report

Group 016

Ashwani Kumar Kamal (20CS10011)
Prerit Paliwal (20CS10046)

November 6, 2022

1 Problem Statement

1.1 Unsupervised Learning

1. Apply **PCA** (select number of components by preserving 95% of total variance). (in-built function allowed for PCA).
2. Plot the graph for PCA.
3. Using the features extracted from PCA, apply **K-Means Clustering**. Vary the value of K from 2 to 8. Plot the graph of K vs normalised mutual information (NMI). Report the value of K for which the **NMI is maximum**. (in-built function not allowed for K-Means).
4. Prepare a report including all your results.

1.2 Supervised Learning

1. Normalise the data using **Standard Scalar Normalisation**. Randomly divide the dataset into 80% for training and 20% for testing. Encode categorical variables using appropriate encoding method (in-built function not allowed for normalization, sampling and encoding).
2. Implement the **Binary SVM classifier** using the following kernels: Linear, Quadratic, Radial Basis function. Report the accuracy for each. (in-built function allowed).

3. Build an **MLP classifier** (in-built function allowed). for the given dataset. Use **stochastic gradient descent** optimiser. Keep learning rate as 0.001 and batch size of 32. Vary the number of hidden layers and number of nodes in each hidden layer as follows and report the accuracy of each:
   - hidden layer with 16 nodes
   - 2 hidden layers with 256 and 16 nodes respectively.
4. Using the best accuracy model from part 3, vary the learning rate as 0.1, 0.01, 0.001, 0.0001 and 0.00001. Plot the learning rate vs accuracy graph.
5. Use **forward selection method** on the best model found in part 3 to select the best set of features. Print the features.
6. Apply **ensemble learning** (**max voting** technique) using SVM with quadratic, SVM with radial basis function and the best accuracy model from part 3. Report the accuracy.
7. Prepare a report including all your results.

2 Solution

2.1 Directory structure

- **data_processing.py** contains helper functions for like **read_data()**: for reading dataset and returning a pandas dataframe object, **trainTestSplit()**: for splitting the dataset according to train and test set, **normalize()** and **forward_feature_selection()**
- **Kmeans.py** contains the code for implemented K-Means Clustering.
- **Q1.py** is the solution file for first part of assignment
- **Models.py** contains the code for implemented Binary SVM and MLP
- **Q2.py** is the solution file for second part of assignment
- **requirements.txt** contains the required packages for running the solution files.

2.2 Unsupervised Learning

2.2.1 Procedure

- In order to achieve our aim of reducing the Feature Dimension using PCA such that it preserves atleast 95% variance in the data, we used an instance of the PCA class provided by **sklearn.decomposition**.
- After Obtaining the Principal Components of the data, to visualize to resulting components and how is the data spread across them. A scatter plot was drawn with the X and Y axis represented by components with maximum and second maximum variance. Also, plotted a line graph of the cumulative variance in data with number of principal components.

- Implemented K-Means Clustering Algorithm in a separate file, **KMeans.py** which has basic helper function like **getInitialRep()** to select initial cluster representative, **NMI_calculator()** to calculate Normalised Mutual Information which uses **mutual_information()** and **entropy()** function to calculate the basic metrics and a final function **KMeans()** which takes a parameter the initial cluster representatives, data and target label (used just to calculate NMI after termination of the algorithm). Here, we simply classify every data point to some cluster based on the which is closest and then update the cluster representative as mean of all data points in that cluster.

2.2.2 Results

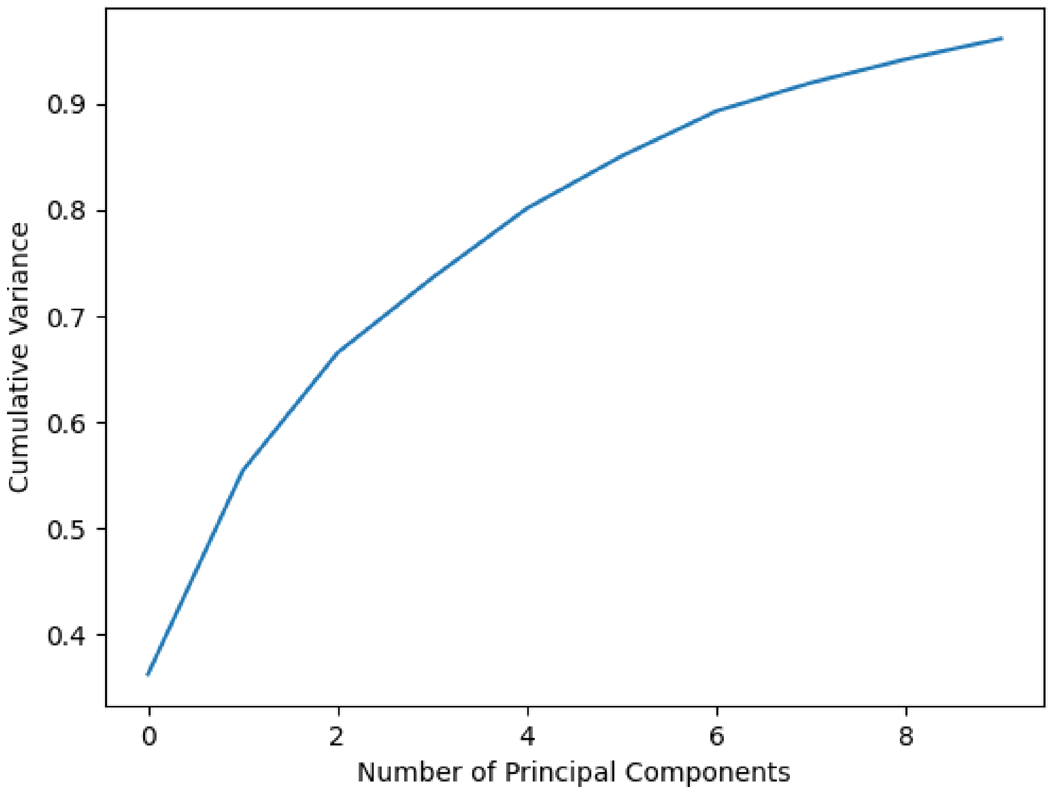

Figure 1: Plot of Variance in Data VS Number of Components

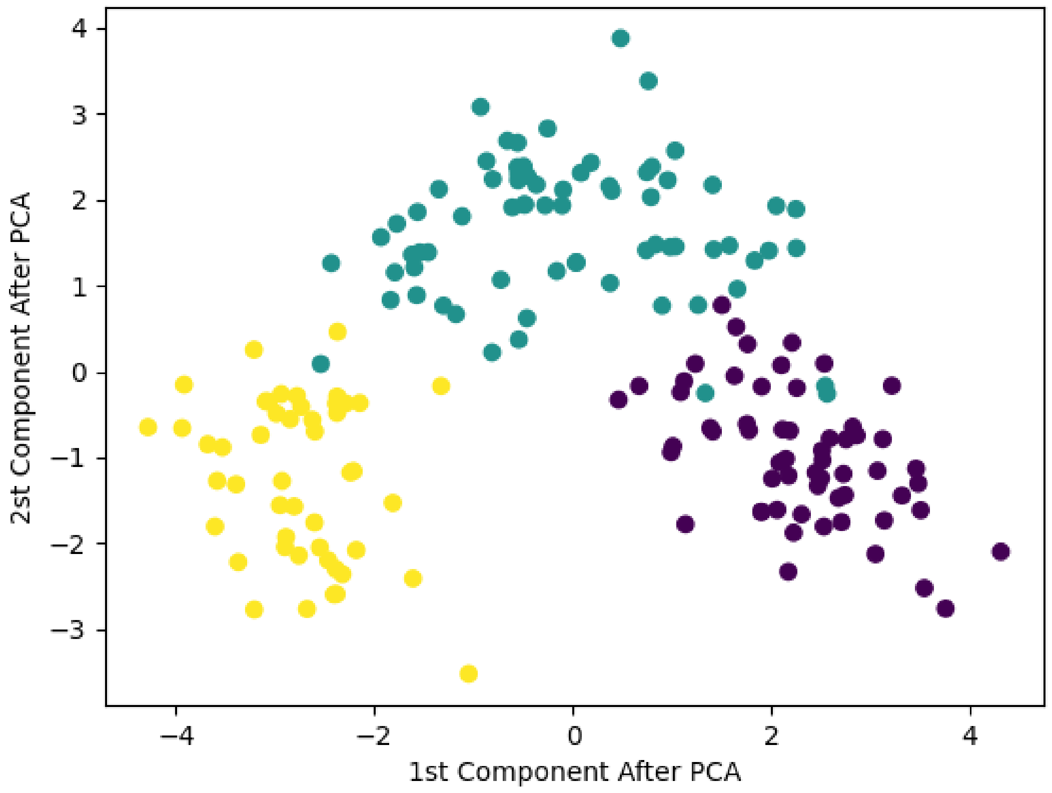

Figure 2: Plot of Visualization of first 2 Principal Components after PCA

- Upon Principal Component Analysis of the given Wine data, it turned out that to preserve 95% of variance we have to use atleast **10 components**. The list of **cumulative variance with number of components** is given as follows : 0.36 0.55 0.66 0.73 0.80 0.85 0.89 0.92 0.94 0.96. (Figure 1)
- The value of **K** which gave the highest value of **Normalized Mutual Information** is 3 with NMI = 0.8758935341223072
- List of **NMI with K from 2-8** is as follows : 0.48, 0.87, 0.81, 0.70, 0.69, 0.68, 0.65 (Figure 3)

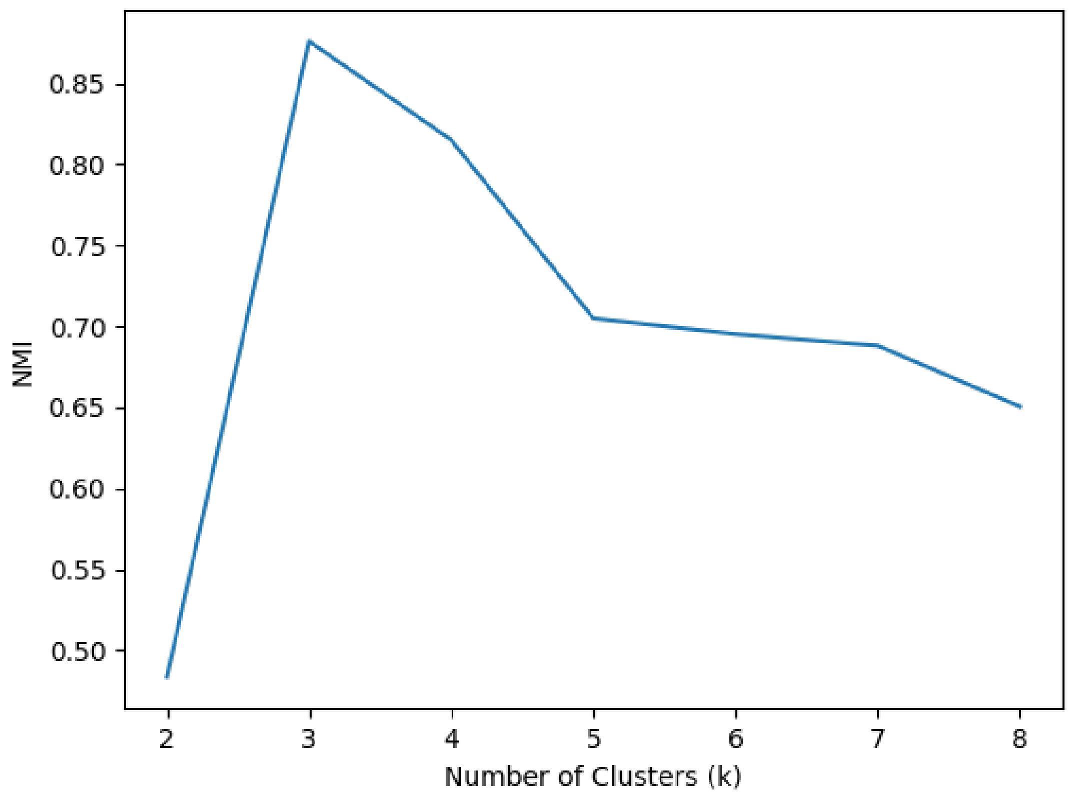

Figure 3: Plot of Normalized Mutual Information VS Number of Clusters (K)

2.3 Supervised Learning

2.3.1 Procedure

Model classes and related functions are written in **Models.py**. The BinarySVM class has the following attributes-

- xTrain: Numpy matrix of the attributes and their values accross instances
- yTrain: Corresponding labels vector for training model
- kernel: type of kernel to be used in the Binary SVM
- yTest: Prediction label vector given a test set
- __model: <sklearn.svm.SVC> class (private attribute)

The MLP class has the following attributes-

- xTrain: Numpy matrix of the attributes and their values accross instances
- yTrain: Corresponding labels vector for training model

- yTest: Prediction label vector given a test set
- hidden_layer_sizes: tuple having values for layer wise information
- learning_rate: default value 1e-3
- solver: default value 'sgd'
- batch_size: default value 32
- __model: <sklearn.neural_network.MLPClassifier> class (private attribute)

1. Divide the dataset in 80:20 ratio after shuffling to get train and test datasets respectively. Using the **normalize(xTrainSet, xTestSet)** function obtain the standardized train and test sets. Instantiate classes for binary SVM using <sklearn.svm.SVC> according the kernels given in problem statement and report accuracy. Related helper class: **BinarySVM**. Report the best accuracy recorded.
2. Instantiate classes for MLP Classifier using <from sklearn.neural_network.MLPClassifier> according the *hidden_layer_sizes* mentioned in problem statement. Related helper class: **MLP**. Report the best accuracy recorded.
3. Instantiate classes for MLP Classifier using <from sklearn.neural_network.MLPClassifier> according the *learning_rates* mentioned in problem statement. Related helper class: **MLP**. Report the best accuracy recorded. Plot the results for *learning_rate* vs *accuracy_list*
4. Using *sequential forward feature selection* get set of best features and report them. Related function: **forward_feature_selection(model, xTest, yTestTrue)**. It returns the list of best feature columns (indices) and prints the final accuracy using those features.
5. Using the prediction labels for models MPL Classifier, Quadratic SVM and Radial SVM, obtain the **mode** of prediction labels instance wise (**max voting**) and report the new accuracy thus obtained.

2.3.2 Results

Since the size of dataset is very small, the models are tending to overfit and some values obtained are very close in range.
Also **trainTestSplit()** function splits the dataset randomly in each run and may result in slightly different values.

- The accuracies obtained on varying *kernel* for Binary SVM are-
  - linear: 0.9444444444444444
  - polynomial degree 2: 0.8333333333333334
  - radial polynomial function: 0.5833333333333334

  Best accuracy recorded for **linear SVM**

- The accuracies obtained on varying *hidden_layer_sizes* for MLP Classifier are-
  - 1 layer of 16 nodes: 0.9722222222222222
  - 2 layers of 256 and 16 nodes: 0.9722222222222222
- The accuracies obtained on varying *learning_rate* for best MLP Classifier obtained in last part are-
  - learning_rate -> 0.1: 0.9444444444444444
  - learning_rate -> 0.01: 0.9722222222222222
  - learning_rate -> 0.001: 0.9722222222222222
  - learning_rate -> 0.0001: 0.9722222222222222
  - learning_rate -> 0.00001: 0.9722222222222222

  Note: Since the size of dataset is small, in some iterations of the run, plot may not show a consistent trend (Also in case of *learning_rate*=1e-5, iterations may reach their maximum limit (2000) and without the optimizer converging).

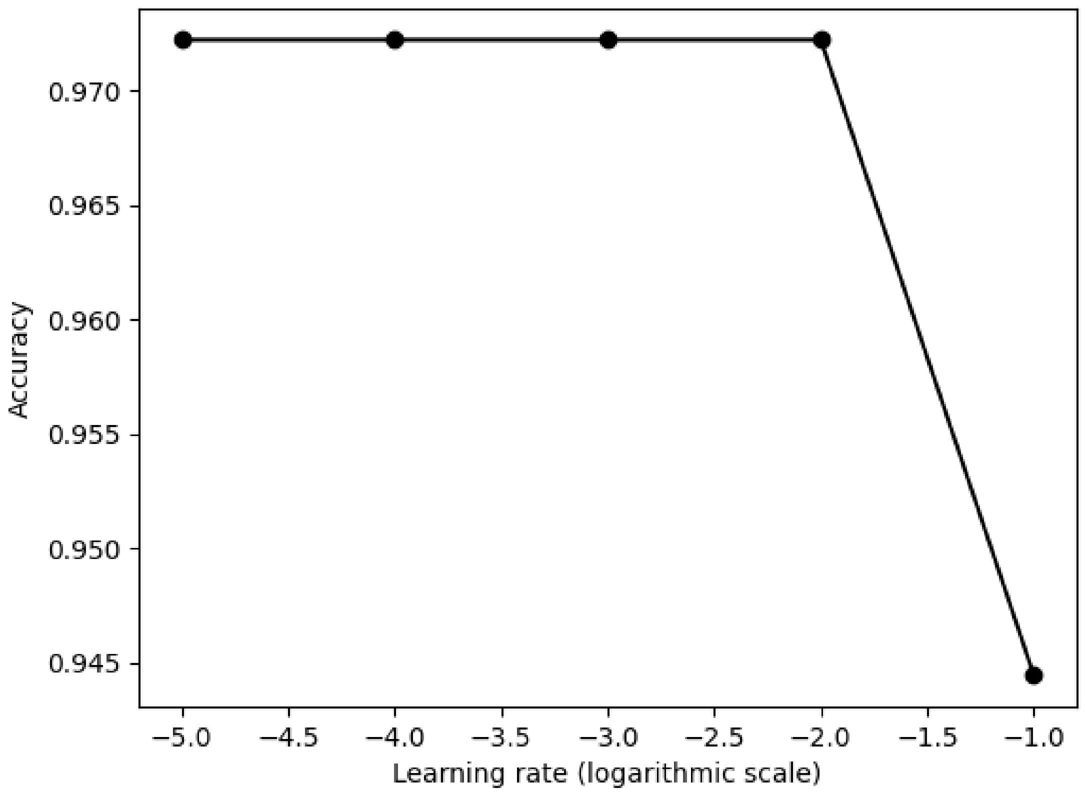

Figure 4: Plot of Learning rate VS Accuracy for MLP Classifier

- Features found in *forward_feature_selection* algorithm-
  - Feature at index 1: Malic acid
  - Feature at index 12: Proline
  - Feature at index 6: Flavanoids
  - Feature at index 10: Hue

  Final accuracy after forward feature selection: 1.0
- **Ensemble learning**: Prediction labels for **MLP Classifier**, **Quadratic SVM** and **Radial SVM** were used to obtain the final prediction label set. Final accuracy reported is- **0.9444444444444444**