Machine Learning Assignment-1

CS60050

Project Report

Group-number: 27

Pranav Mehrotra (20CS10085) and Saransh Sharma (20CS30065)

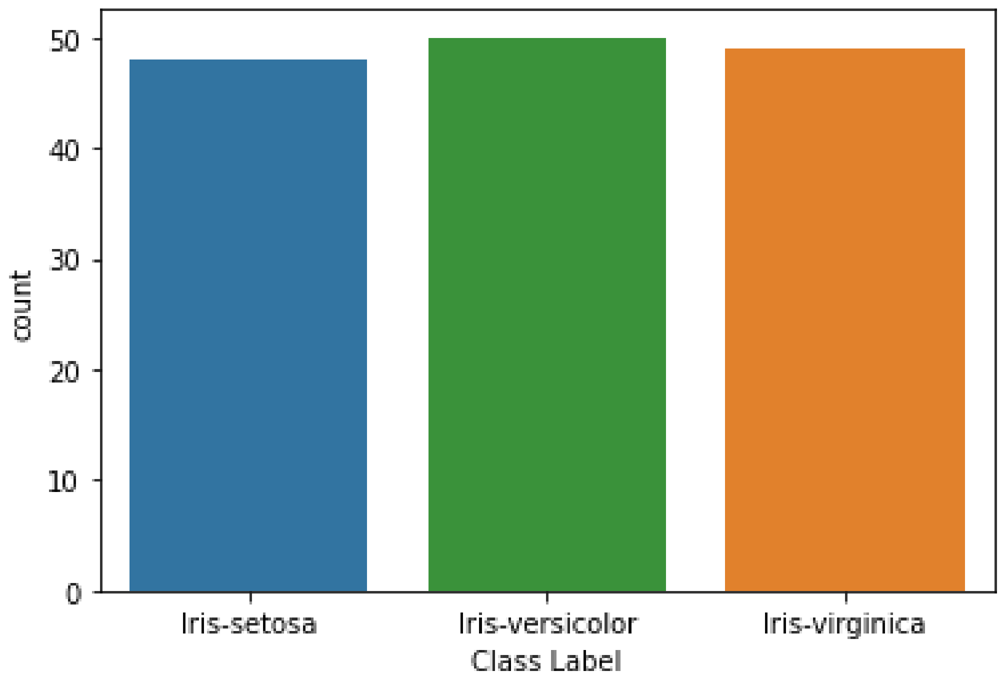
Question 1) Unsupervised Learning:

Source File: **q1.py**

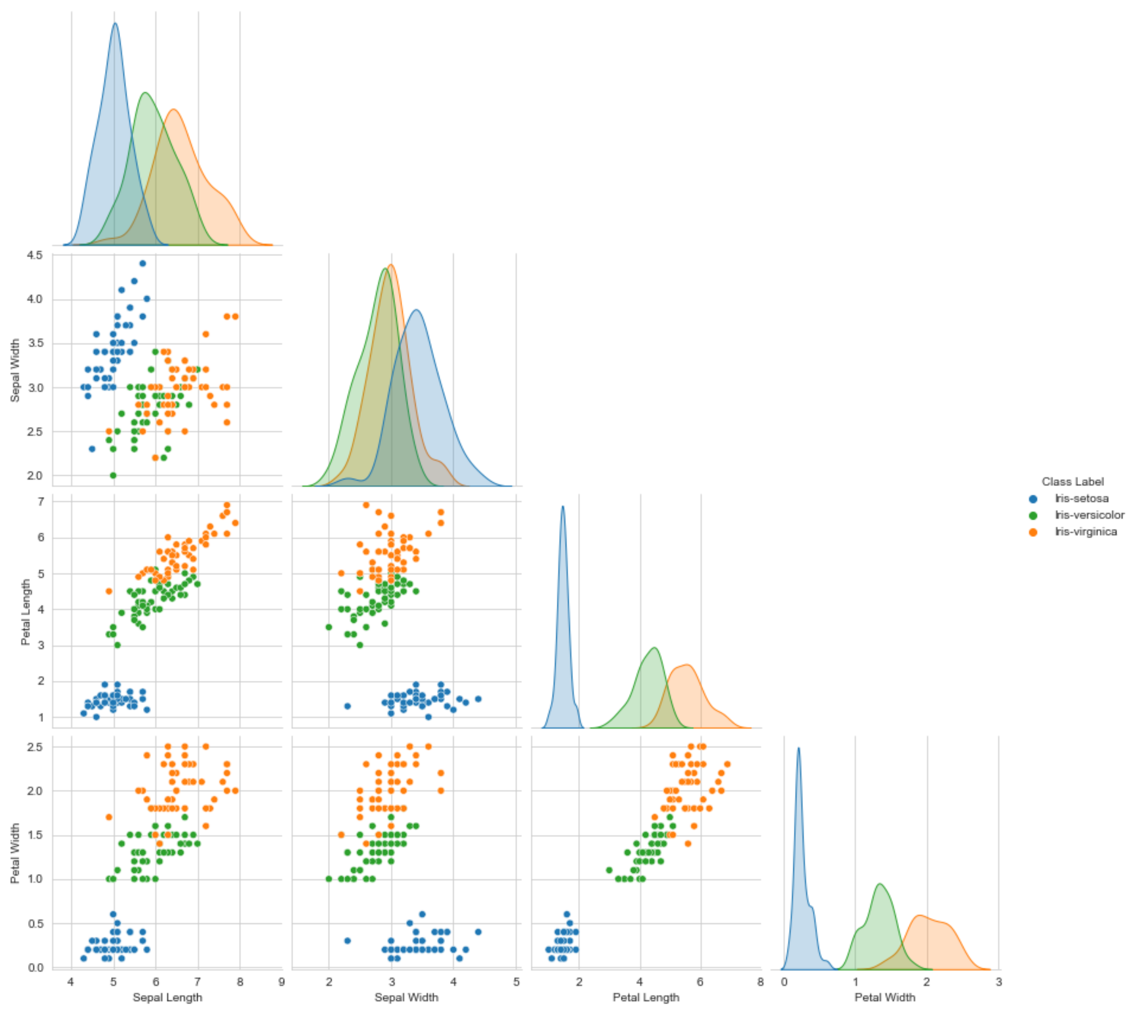
Exploratory Data Analysis and Data visualization:

This step is performed to gain insights from data. Data visualizations help us to understand data better and make conclusions about the distribution of data.

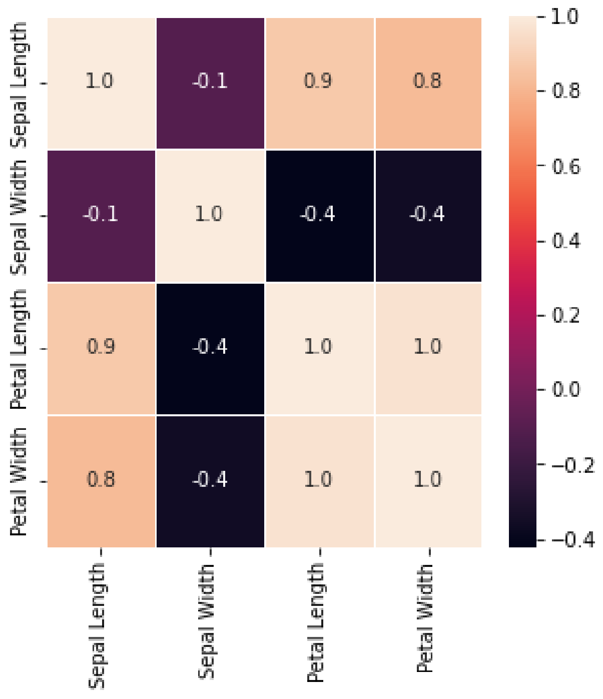
1. Count Plot: depicts the number of datapoints belonging to each class label.



2. Pair Plot: depicts the kernel density of different columns with different curves for different classes and how data is scattered when there two columns are plotted.



3. Heatmap: depicts the correlation i.e. extent of linear dependability between different features of the data.



	Sepal Length	Sepal Width	Petal Length	Petal Width
Sepal Length	1.0	-0.1	0.9	0.8
Sepal Width	-0.1	1.0	-0.4	-0.4
Petal Length	0.9	-0.4	1.0	1.0
Petal Width	0.8	-0.4	1.0	1.0

The heatmap clearly indicates high linear dependability between different columns suggesting that a few columns could be dropped to reduce the dimension.

Task-1 PCA:

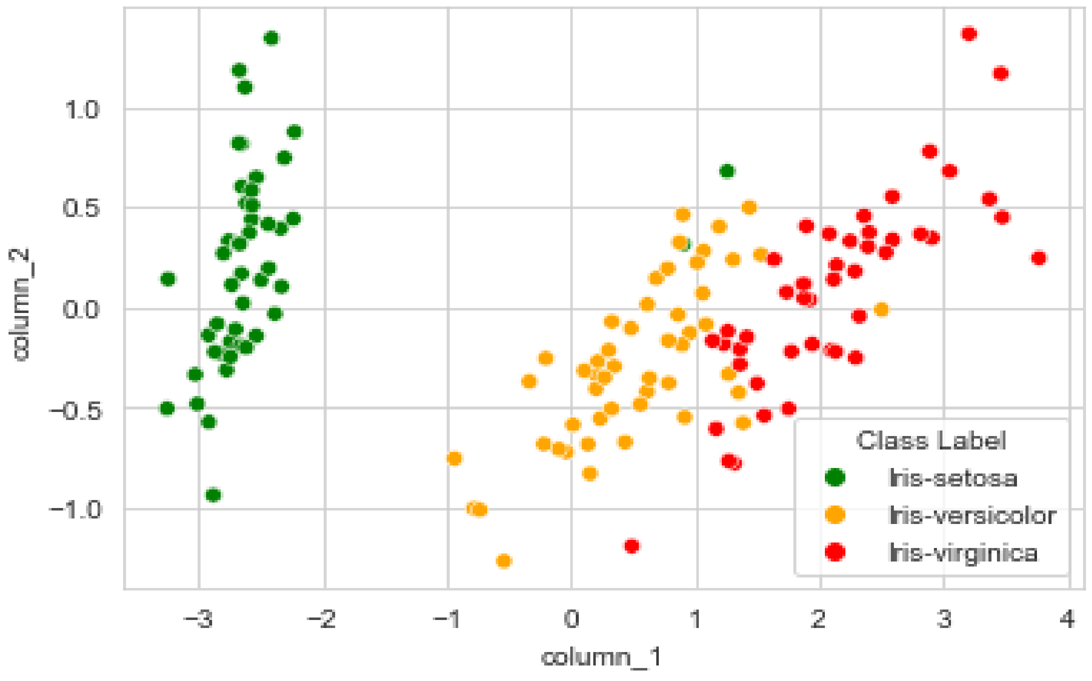
- Inbuilt function PCA from sklearn library is used.
- x and y are extracted from the original data frame where x is the dataset and y is the output label.
- PCA(n_components=0.95) creates an object of class PCA where in the number of components are selected by preserving **95%** of total variance.
- pca.fit_transform(x) fits the PCA model to datapoints in x and return components such that 95% of the variance is preserved.
- The output of PCA is a 147*2 dimensional that is 4 columns of original dataset are now projected to 2 columns.
- Original First five rows:

	Sepal Length	Sepal Width	Petal Length	Petal Width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

- First five rows after PCA:

	column_1	column_2
0	-2.710782	0.322125
1	-2.741763	-0.175061
2	-2.916691	-0.141509
3	-2.773363	-0.315205
4	-2.755418	0.330133

- Since the output dataset has two columns, we can plot a graph with x axis representing the first column and y axis representing the second column.



- Next task would be to convert categorical output labels to integers so that we can use them while calculating class entropy for NMI Calculation.

Task-2 K-means on data obtained from PCA:

Supporter functions:

1.computecluster_rep:

- **The function is used to update cluster locations depending upon cluster assignment.**
- The function takes as input dataset X, cluster assignment vector idx, number of clusters K and vector containing feature values corresponding to clusters i.e. coordinates of clusters.
- For every cluster i, find all the data points who have been assigned this cluster i, calculate the mean of features of these data points and update the location/coordinate/feature vector corresponding to cluster I with this mean vector.
- In case no dataset is assigned this cluster update the location of this cluster to all zeroes.
- Return the updated cluster locations.

2.findClosestcluster_rep:

- **The function is used to assign clusters to each datapoint.**
- Takes as input dataset X and cluster locations or feature vector of all clusters.
- cluster_rep is an array where in ith entry contains set of values corresponding to different features for ith cluster.
- For every datapoint, calculate its distances wrt all clusters and assign the index of cluster wrt which the distance is minimum.

np.random.seed(42) makes all the random process deterministic so that the results remain consistent and can be documented.

3.fromdatainitialise:

- **The function us used to initialize k clusters.**
- The function takes as input dataset X and number of clusters k.
- The function first generates a random permutation of indices that will effectively shuffle the data and then first k datapoints are picked from the randomly shuffled dataset. These k datapoints will n ow act as our initial cluster locations.

4.runkMeans:

- **The function is used to run K-mean clustering algorithm on dataset.**
- The function takes as input dataset X, cluster locations, and functions like findClosestcluster_rep and computecluster_rep.
- The function first assigns clusters to datapoints.
- Depending upon cluster assignment done in previous step, cluster locations are updated.
- The process is repeated until a sufficiently large number say 100 iterations are over or cluster locations doesn’t change. This will be our convergence criteria.
- The number of iterations are restricted to prevent infinite loop caused due to any reason.
- The function returns cluster locations after convergence and number of iterations needed.

5.ComputeCost:

- **The function is used to calculate cost to print final cost**
- The function takes as input dataset X , cluster locations cluster_rep and cluster_assignment vector idx.
- Find the sum of distances of datapoint with its assigned cluster.

6.class_entropy:

- The function returns entropy of class labels. This is calculated for the entire dataset and can be calculated prior to clustering.
- For every label i, calculate what fraction frac of datapoints have label i. calculate sum of frac * log2 frac.
- Return the negative of sum.
- H(Y)

7.cluster_entropy:

- The function returns entropy of cluster labels.
- For every cluster label i, calculate what frac of datapoints have cluster assignment I, calculate sum of frac * log2frac.
- Return the negative of sum.
- H(C)

8. mutual:

- The function returns entropy of class labels within each cluster.
- H(Y|C = 1) = -1 * P(C = 1) ∑ P(Y=y | C=1) log2 (P(Y=y | C=1)) for all output labels y.
- The function returns ∑H(Y|C=c) for all c

NMI = 2*(H(Y) – H(Y|C)) / (H(Y) + H(C))

H(Y) is class_entropy

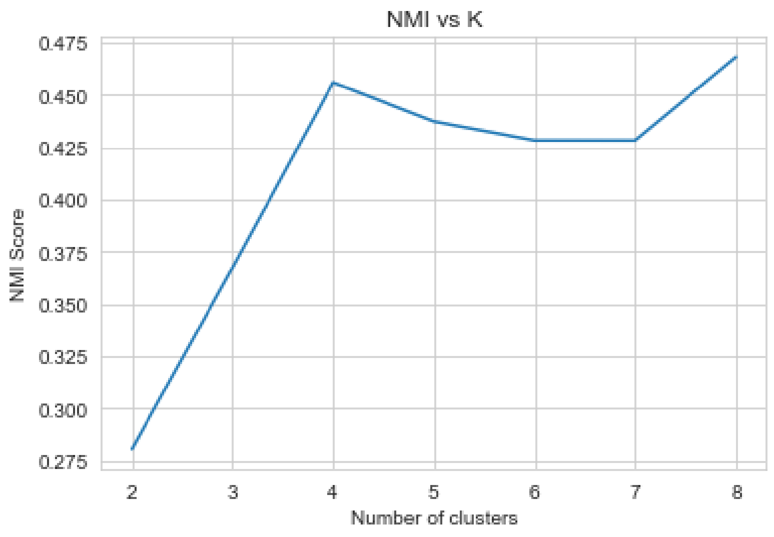
H(C) is cluster_entropy

H(Y) – H(Y|C) is mutual information

K-means on dataset obtained from PCA:

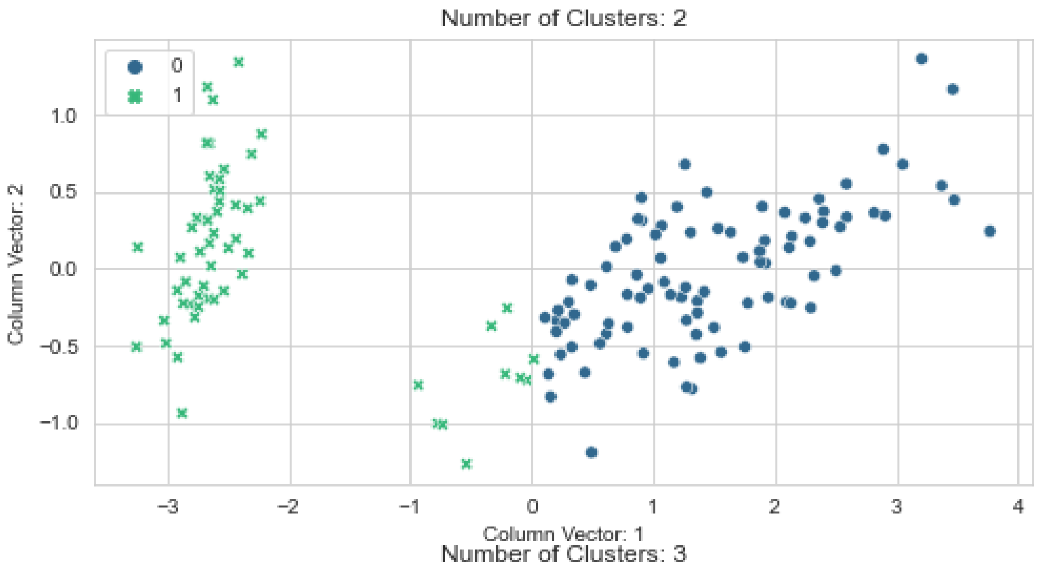
For cluster size from 2 to 8 run the runkmeans function and calculate nmi and cost for all instances. A list containing NMI values is created for plotting purpose. For every k, initialise cluster locations, run k means clustering algorithm, calculate NMI and cost.

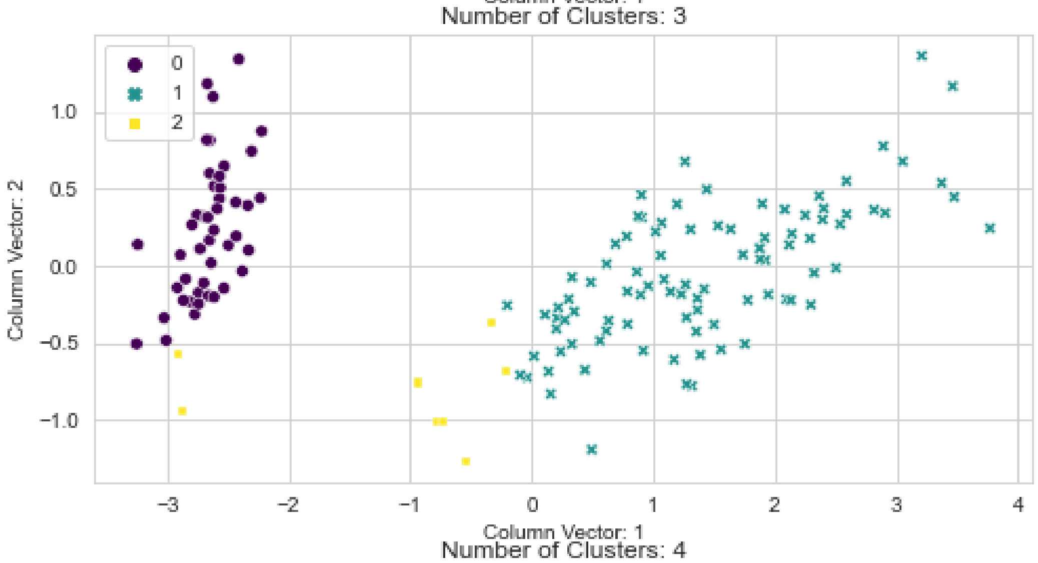
NMI vs K curve:

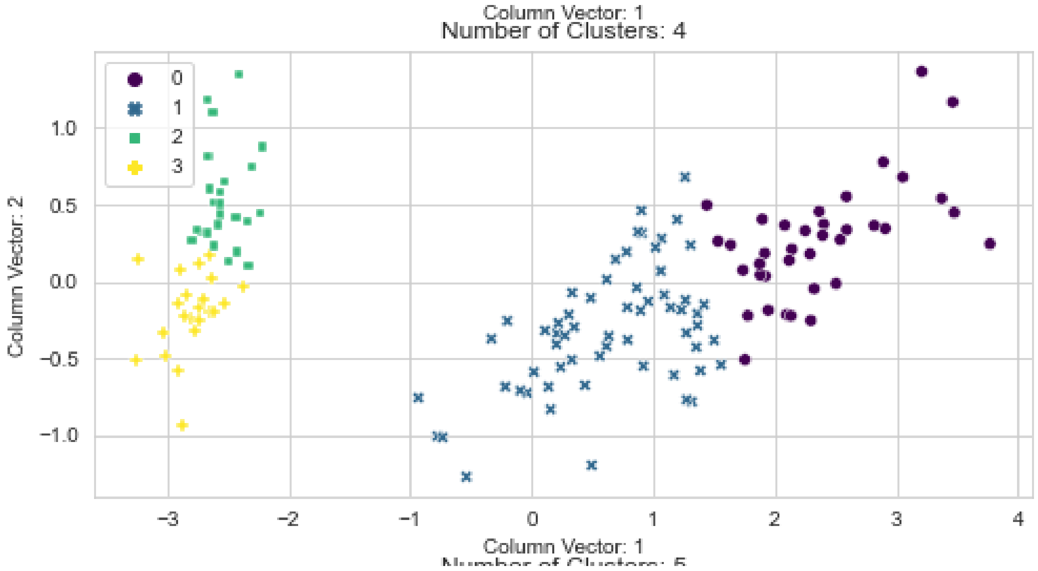


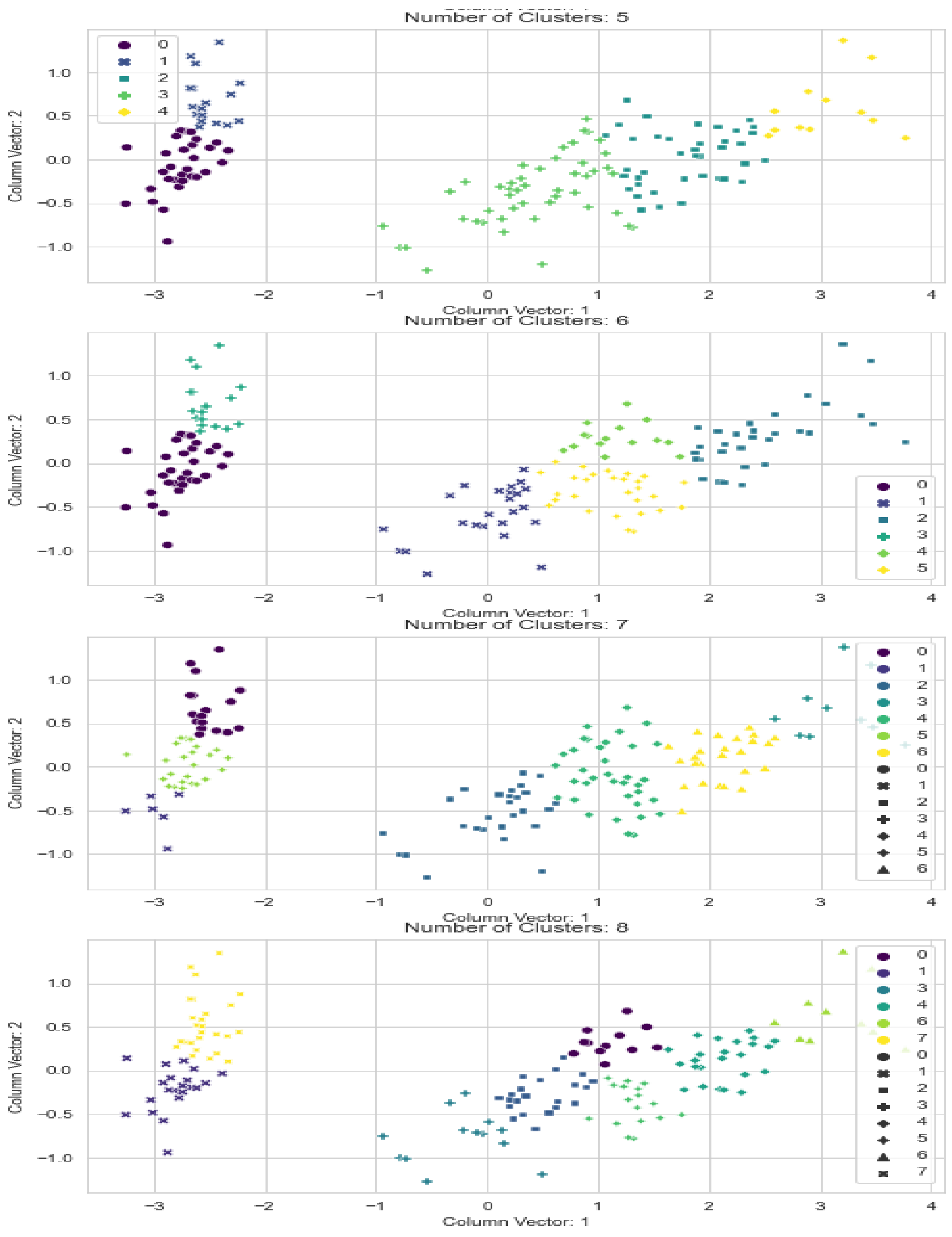
NMI is maximum for K = 8

How does clustering upon data look like?









Question 2) Supervised Learning

The files of the code: “**q2.py**”

This file contains the whole python code of the model and data evaluation. It mainly has mainly following functions:

Task-1: Standard Scalar Normalization:

Helper function:

1.normalize:

To normalize data, for all feature columns x, do x = x- mean(x)/standard_deviation(x). This process makes mean of all feature columns 0 and standard deviation of all feature columns 1.

The function helps in scaling features to same scale, so that no feature has a bias in the predictions.

```
d = df.sample(frac = 1,random_state=2) #returns a randomly jumbled data
div = int(0.8 * d.shape[0])#calculate 80 percent of the number of input datapoints

#categorical encoding
d.loc[data['Class Label']=="Iris-setosa","Class Label"]=0
d.loc[data['Class Label']=="Iris-versicolor","Class Label"]=1
d.loc[data['Class Label']=="Iris-virginica","Class Label"]=2

d_train, d_test = d.iloc[:div,:], d.iloc[div:,:]#split the data into test and train
```

Random division of dataset into test and train dataset and convert categorical output labels to integer.

Task-2 Binary SVM Classifier:

SVM Variants:

1. SVM with Linear kernel:

svm.SVC(kernel='linear',random_state=42) creates an object of SVM Classifier class where svm is imported from sklearn library. Random_state = 42 ensure deterministic results. Kernel = ‘linear’ creates an instance of SVM classifier with linear kernel.

clf.fit(d_train_x, d_train_y) fit is a member function of SVM class which trains the model taking in as input dataset and input output label. Training set is passed as parameter to the function.

y_pred = clf.predict(d_test_x) .predict is a member function of SVM class which makes prediction upon input dataset. The clf which is a SVM model is trained on d_train_x and d_train_y will now make predictions on d_test_x.

np.sum(y_pred == d_test_y.astype("int"))/d_test_y.shape) Calculates the accuracy of the classifier, i.e. fraction of dataset correctly labelled.

SVM with Linear kernel: Accuracy [0.9]

2. SVM with Quadratic kernel:
svm.SVC(kernel='poly’, degree = 2,random_state=42) creates an object of SVM Classifier class where svm is imported from sklearn library. Random_state = 42 ensure deterministic results. Kernel = ‘poly’ creates an instance of SVM classifier with polynomial kernel and degree = 2 specifies the degree of the polynomial i.e. SVM with quadratic kernel.

clf.fit(d_train_x, d_train_y) fit is a member function of SVM class which trains the model taking in as input dataset and input output label. Training set is passed as parameter to the function.

y_pred = clf.predict(d_test_x) .predict is a member function of SVM class which makes prediction upon input dataset. The clf which is a SVM model is trained on d_train_x and d_train_y will now make predictions on d_test_x.

np.sum(y_pred == d_test_y.astype("int"))/d_test_y.shape) Calculates the accuracy of the classifier, i.e. fraction of dataset correctly labelled.

SVM with quadratic kernel: Accuracy [0.83333333]

3. SVM with radial basis function kernel:

svm.SVC(kernel='rbf',random_state=42) creates an object of SVM Classifier class where svm is imported from sklearn library. Random_state = 42 ensure deterministic results. Kernel = 'rbf' creates an instance of SVM classifier with radial basis function kernel.

clf.fit(d_train_x, d_train_y) fit is a member function of SVM class which trains the model taking in as input dataset and input output label. Training set is passed as parameter to the function.

y_pred = clf.predict(d_test_x) .predict is a member function of SVM class which makes prediction upon input dataset. The clf which is a SVM model is trained on d_train_x and d_train_y will now make predictions on d_test_x.

np.sum(y_pred == d_test_y.astype("int"))/d_test_y.shape) Calculates the accuracy of the classifier, i.e. fraction of dataset correctly labelled.

SVM with radial basis function kernel: Accuracy [0.93333333]

Task-3 MLP Classifier:

Module neural network is imported from sklearn library.

neural_network.MLPClassifier(hidden_layer_sizes =(),solver='sgd',batch_size=32, learning_rate_init = 0.001,max_iter=4000,shuffle=False,random_state=2)

hidden_layers_size: type - tuple
(16) for first variant wherein, we have one hidden layer with 16 nodes. (256,16) for second variant wherein we have 2 hidden layers of 256 and 16 nodes respectively. ith (from i = 0) element of tuple indicates number of nodes in i+1th hidden layer.

solver = 'sgd' specifies the type of optimizer i.e., Stochastic Gradient Descent in this case.

batch_size: Size of minibatches for stochastic gradient descent optimizers.

learning_rate_init: specifies the learning rate of the stochastic gradient descent optimizer.

max_iter: specifies the maximum number of iterations for the stochastic gradient descent.

shuffle: False prevent random shuffling of data by MLP Classifier

random_state: 2 makes the output of MLP Classifier deterministic.

Accuracy of One 16 node hidden layer MLP Classifier: [0.93333333]

Accuracy of 256,16 node hidden layer MLP Classifier: [0.9]

16 node hidden layer MLP classifier has higher accuracy

Varying learning rate :

The best model of the two variants is chosen by comparing their accuracy. Accuracy of variant 1 is better than variant 2 therefore, 16 node hidden layer MLP is chosen.

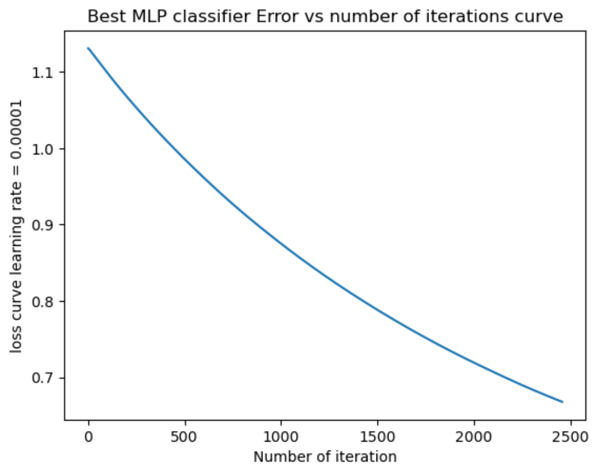
The best model is then trained with varying learning rates. The learning rate is varied from [0.00001,0.0001,0.001,0.01,0.1]

The model is then trained on d_train_x and d_train_y. the number of iterations and loss curve i.e. loss at intermediate steps are saved in list y.

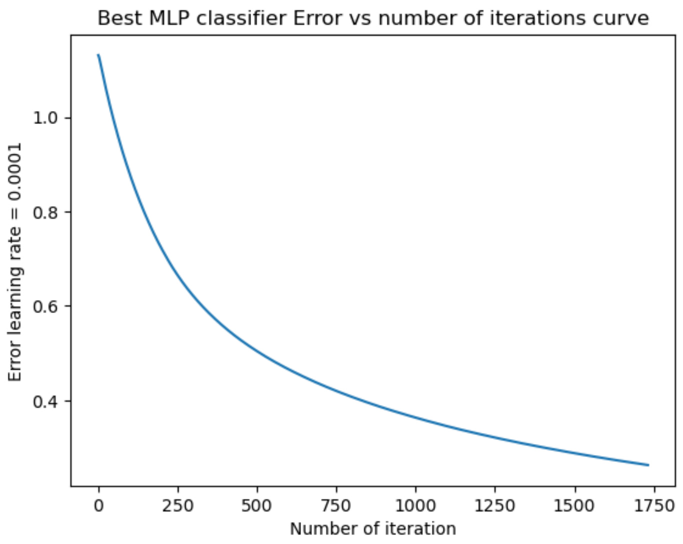
n_iter_: indicates the number of iterations taken by the classifier to converge

loss_curve_: is a list containing loss at every iteration.

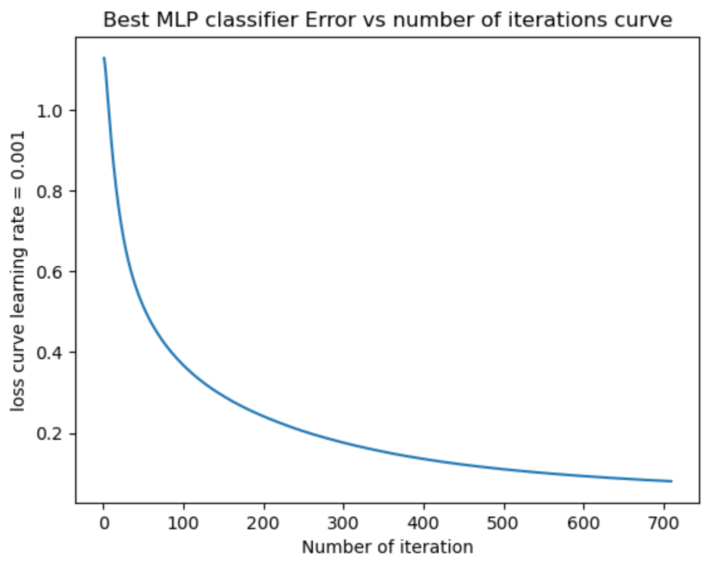
Learning_rate: 0.00001



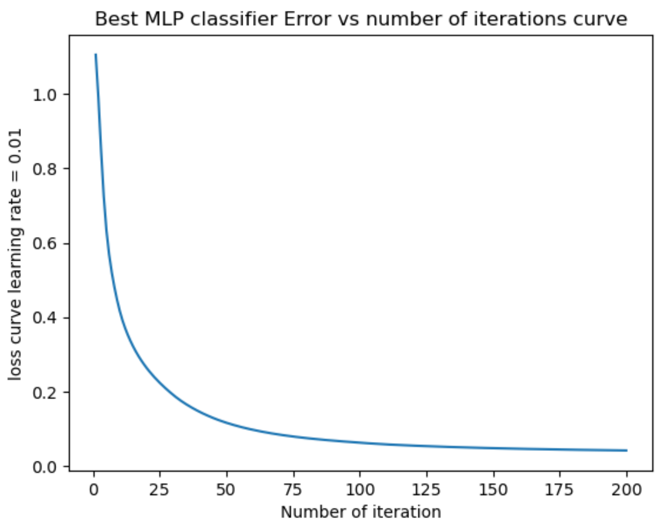
Leaning_rate: 0.0001



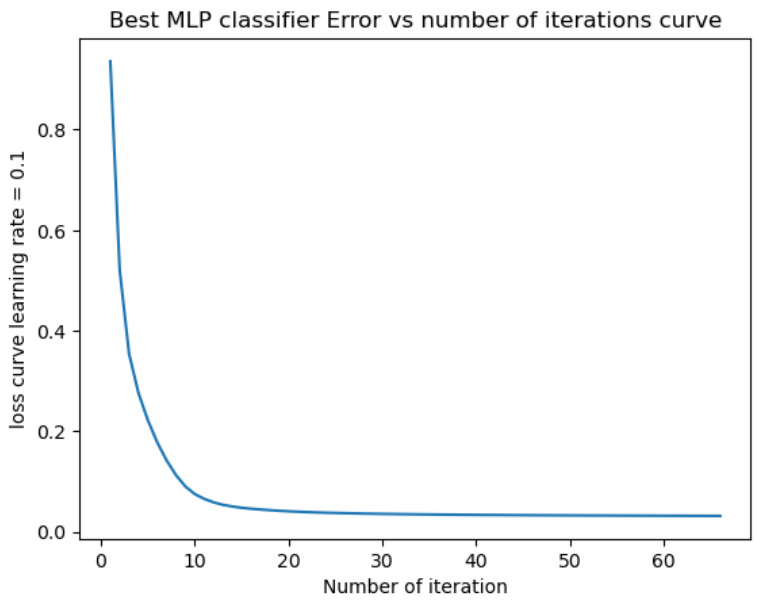
Learning_rate:0.001



Learning_rate: 0.01

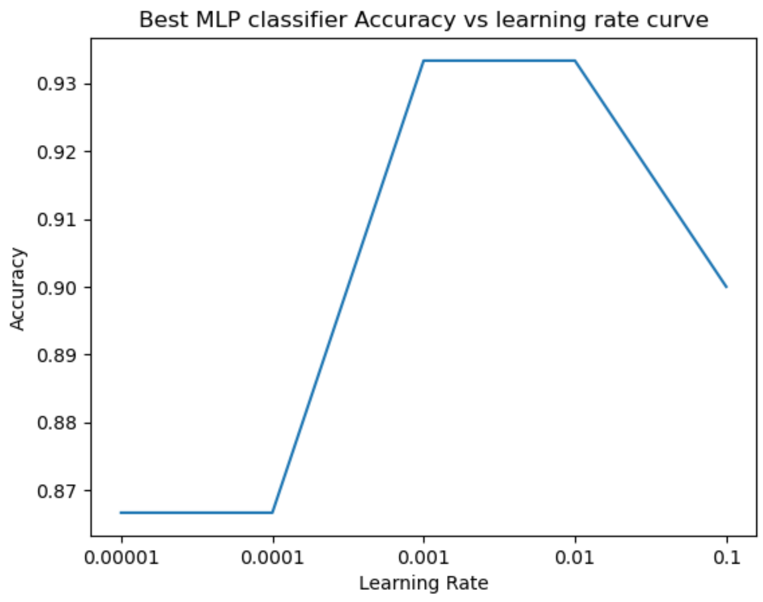


Learning_rate: 0.1



As learning rate is increased from 0.00001 to 0.1, the number of iterations decrease tremendously while the stochastic gradient descent optimizer converges faster.

Accuracy vs Learning_rate curve:



Even though stochastic gradient descent optimizes faster and number of iterations decreases as learning rate is increased, the accuracy doesn't improve that linearly. The maximum accuracy is achieved for learning rate 0.001 and 0.01. The accuracy first increases and then again decreases.

Task-4 Backward elimination method:

accuracy variable stores the accuracy of the best model of the two variants of MLP Classifier.

While we have more than or equal to two features i.e. there is a feature which can be removed and checked, repeat the following:

Find the feature which when dropped from training dataset gives highest accuracy on test set. That is iterate over all features, remove the feature from the data temporarily, check the accuracy of the newly trained data and find the max of all these accuracies.

Then check if the max accuracy obtained after removing a feature from data is greater than or equal to the accuracy of the original model.

If removing the feature doesn’t harm the accuracy, remove the feature completely, update the value of accuracy variable to the accuracy of model after the feature is removed and update the feature list.

Iterate over the new set of features.

If the max accuracy obtained after removing a feature from data isn’t better than original model’s accuracy, exit the process. This indicates no feature can be further removed.

Backward Elimination in progress:

Accuracy before backward elimination: [0.93333333]

Number of features before backward elimination: 4

Initial features: ['Sepal Length', 'Sepal Width', 'Petal Length', 'Petal Width']

Features left in each iteration:

Index(['Sepal Width', 'Petal Length', 'Petal Width'], dtype='object')

Index(['Sepal Width', 'Petal Width'], dtype='object')

Index(['Petal Width'], dtype='object')

Accuracy after backward elimination: [0.93333333]

Number of features after backward elimination: **1**

Columns after backward Elimination: **['Petal Width']**

Task–5 Ensemble Learning:

SVM with radial basis function kernel Accuracy: [0.93333333]

SVM with quadratic kernel Accuracy: [0.83333333]

MLP Classifier Accuracy: [0.93333333] (16 node hidden layer one)

To perform ensemble learning upon the three models shown above, We first make predictions using all the three models and stack the prediction vector one above another.

After stacking:

```
[[1 0 1 1 1 1 1 0 2 0 2 1 2 0 2 0 2 2 0 1 0 2 1 0 1 2 0 0 1 0]
 [1 0 1 1 1 1 1 0 0 0 0 1 2 0 2 0 0 2 0 1 0 2 1 0 1 1 2 0 1 0]
 [1 0 2 1 1 1 1 0 2 0 2 1 2 0 2 0 2 2 0 1 0 2 1 0 2 2 0 0 1 0]]
```

Where each row i represents predictions made by model i on test dataset.

To execute max-voting criteria, we take mode along every column to find the label predicted by maximum models. If for a dataset two models predict 2 and one model predict 1, we assign the final class 2 to the dataset.

Accuracy after ensemble learning: [0.93333333]