Machine Learning (CS60050)
Assignment 2 Report

Group-5

Members:

- **Pranav Nyati (20CS30037)**
- **Shreyas Jena (20CS30049)**

Question 1:
Unsupervised Learning

Introduction:

- In the given problem, we perform *k-means clustering* on input data (dimensionality reduced using *PCA*) for different values of k and choose a suitable value based on *Normalized Mutual Information (NMI)*.

- The tasks carried out are :
    1. Apply PCA to given input data such that >95% of the total variance is preserved.
    2. Perform k-means clustering on the PCA-extracted features.
    3. For different values of k, compute the NMI value, and report the value of k for which NMI is maximum.

Theoretical Background:

- *Principal Components Analysis* is a method to reduce the dimensionality of data by representing it in terms of a set of principal components (a new set of orthonormal basis vectors) such that the information is preserved as much as possible.
- *K-means clustering* is an unsupervised learning technique that performs the grouping of observations into clusters, in which each observation is assigned to the cluster with the nearest mean.

- Initialization is done by randomly choosing k cluster representatives from the given dataset.
- For each iteration, we first assign each observation to the nearest cluster based on distance from the cluster representatives.
- This is followed by updating the values of the cluster representatives by taking the mean of all vectors in each cluster.

  $$c_i = argmin_j \left\|X_i - rep_j\right\|^2$$

  where,

  c_i = cluster assigned to ith observation

  rep_j = jth cluster representative

- We compute J_{clus}, the sum of squared distances of each observation from its cluster representative. The algorithm converges if J_{clus} stops decreasing beyond a certain threshold.

Dataset:

- The given dataset contains 178 samples with 13 features, each specifying an attribute related to wine quality.
- Data preprocessing was carried out in the following steps:

1. The data was normalized using Standard Scaler normalization, which normalized the values to unit variance.
2. The encoding step was not needed since all data values were continuous.
3. The dataset was randomly divided into 80% for training and 20% for testing.
    - **Train set -** 142 samples
    - **Test set -** 36 samples

Steps:

- The given input observations (with 13 features) was subjected to Principal Component Analysis using `sklearn.decomposition.PCA`, leading to a reduced dataset with n_components = 10, giving > 95% explained variance.
- The reduced dataset was then subjected to the k-means algorithm for values of k ranging from 2 to 8.
- The value of Normalized Mutual Information (NMI) for each k was computed, and the value of k for which NMI is maximum was found to be **k = 3**.

Helper functions: (in file **q1.py**)

- `preprocess` – function for data preprocessing.
- `PCA_fit` – function to apply dimensionality reduction using PCA.
- `PCA_plot_var_vs_PC` – function to plot the cumulative explained variance vs no. of Principal Components.
- `PCA_plot_scatter` – function for plotting reduced data w.r.t its first 2 PCs.
- `kmeans` – function to implement the K-means algorithm.
- `normalized_mutual_info` – function to compute NMI score.
- `mutual_information` – function to compute the mutual information for two random variables.
- `entropy` – function to compute the entropy of a random variable.
- `NMI_vs_k` – function to compute NMI values for different values of k.
- `NMI_vs_k_plot` – function to compute NMI vs k plot.

Results:

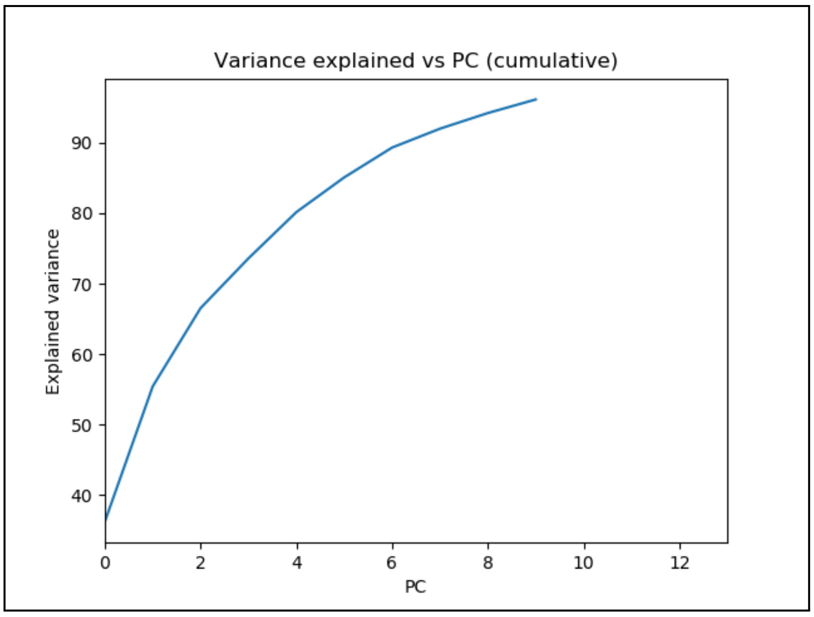
k (Number of clusters)	Normalized Mutual Information (NMI)
2	0.5013631823800656
3	**0.8472896471857924**
4	0.7569277351151601
5	0.7525772009894589
6	0.6339057263954603
7	0.632643641229236
8	0.633194337715482

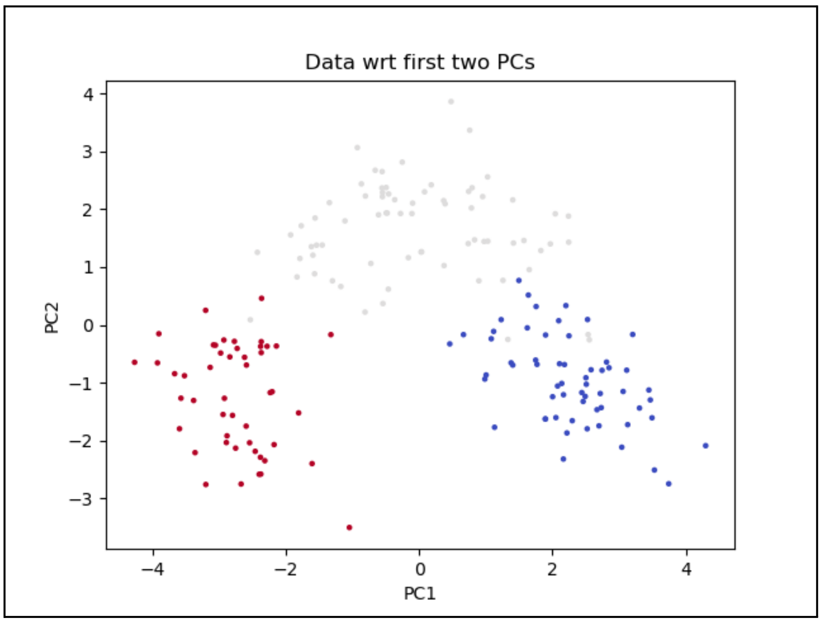
Given,

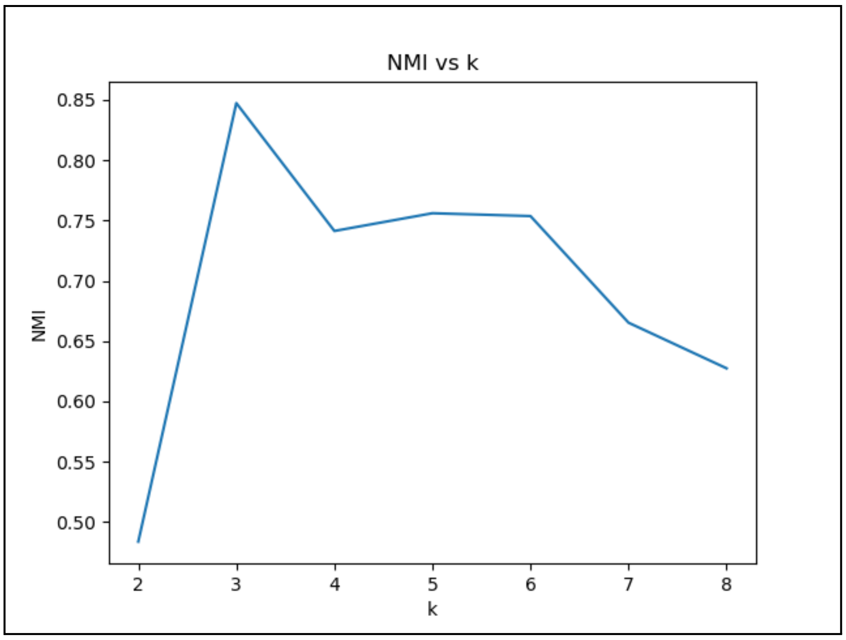
number of Principal Components = **10** (explained variance = **96.17%**),

Value of k for which NMI is maximum = 3

Plots:


Variance explained vs PC (cumulative)
Explained variance
PC
40
50
60
70
80
90
0
2
4
6
8
10
12


Data wrt first two PCs
PC2
PC1
−4
−2
0
2
4
−3
−2
−1
0
1
2
3
4


NMI vs k
NMI
k
0.50
0.55
0.60
0.65
0.70
0.75
0.80
0.85
2
3
4
5
6
7
8

Question 2:
Supervised Learning

Theoretical Background:

1) Support Vector Machine (SVM):

- Binary SVM Classifier is a linear discriminant classifier that makes use of Vapnik's Principle, i.e., to never solve a more complex problem as a first step before the actual problem.
- After training, we can write the weight vector in terms of training samples lying in class boundaries.
- The Primal problem for soft-margin hyperplanes and kernel function can be formulated in the following form:

  Minimize wrt '**w**':

  $$L_p = \frac{1}{2}||w||^2 + C \star \sum_t s^t - \sum_t \alpha^t [r^t (w^t \phi(x^t) + w_0) - 1 - s^t] - \sum_t \mu^t s^t$$
- We have used three different kernel functions and vary their hyperparameters to train the Support Vector Machine. The kernels we have used are as follows:
  – **Linear Kernel (linear): ⟨x, x ′⟩**, no hyperparameters
  – **Radial Basis Function Kernel (rbf)**: **exp(−γ // x − x ′ // ²)**, hyperparameter is gamma(γ)
  – **Polynomial Kernel (poly): (γ⟨x, x ′⟩) ᵈ** , hyperparameters are gamma(γ) and degree(d) : We have used a Quadratic kernel, i.e. we have taken the hyperparameter degree d = 2.

2) Multi-Layer Perceptron (MLP):

- It is a network of perceptrons arranged in a particular fashion, which constitutes the architecture of the model.
- We consider a multi-layered feed-forward network, which essentially implies that there isn't any feedback loop in the network.
- We define an error function, and we try to minimize w.r.t. **w** parameter using Stochastic Gradient descent (SGD).
- The backpropagation algorithm helps to find out the changes in each and every parameter at each level. It simply refers to the chain rule of partial derivatives.

Minimize

$$J_n(W) = \frac{1}{N}\sum_{i} \left\| O_i - F(X_i; W) \right\|^2$$

Starting with W_0, we update the weights in SGD using the following rule:

$$W_i = W_{i-1} + \eta(i) \sum_{k} (O_k - F(X_k; W_{i-1})) \nabla F(X_k; W_{i-1})$$

3) Feature Selection using Forward Selection Approach:

- Forward selection is a type of stepwise regression which starts with a model with no variables and adds in variables one by one by testing the addition of each variable using a chosen model fit criterion.

- In each forward step, the variable that gives the single best improvement to your model is added.

- This process is repeated until addition of a new attribute does not improve the accuracy of the model to a statistically significant extent.

4) Ensemble Learning using Max Voting Approach:

- Ensemble learning is a general approach to machine learning that seeks to achieve better predictive performance by combining the predictions from multiple models trained over the training dataset.
- The main idea of Ensemble Learning is that it aims to reduce the likelihood of a wrong prediction, as there is less probability that multiple models (trained for the same task to similar performance levels) - all make a wrong prediction for a test example. A poor performance from one model can be offset by a strong performance from other models.
- Other applications of ensemble learning include assigning a confidence to the decision made by the model, selecting optimal features, incremental learning, etc.
- There are various common methods of Ensemble Learning such as Bagging, Boosting, AdaBoost, and Voting based Methods.

Here we use a Voting Based Method called **Max Voting** to do ensemble learning on based on our previously trained SVM and MLP classifiers.

- **Max Voting:** In Max voting for Classification, we assign the class label which is predicted by most number of models (i.e. the class label with highest frequency of occurrence) for a particular test example as the output class label for that test example.
- Max voting is beneficial only when all the machine learning classifier models which are a part of the ensemble perform at similar levels. A max-voting ensemble built from models with contrasting levels of efficiency may perform erratically, and it may happen that the ensemble may have poorer accuracy than the accuracy of some of the individual models in the ensemble.

Procedure and Results:

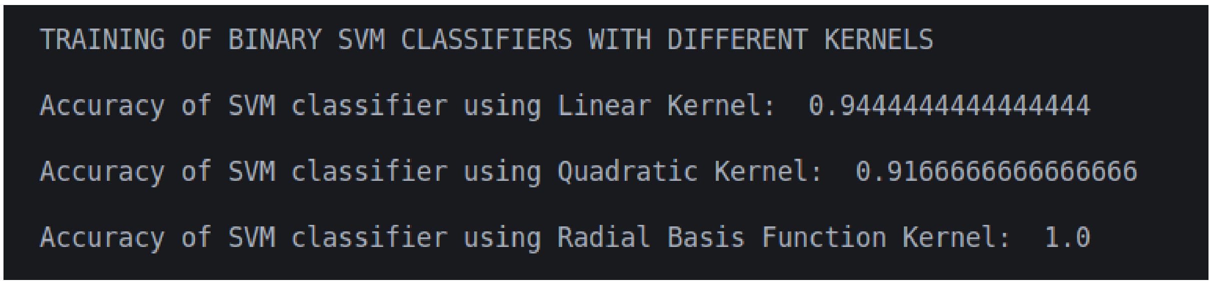
1) Data Reading and Train-Test Split:

- The wine dataset was read from the input file "wine.csv" using the **pandas.read_csv** function into a pandas dataframe.
- Standard Scaler Normalization for each dataset sample was performed by subtracting each of the attribute values by their mean across all samples and dividing by their standard deviation.
- There were no categorical attributes in our dataset (except the class labels which were already encoded in the dataset), hence we didn't require encoding of categorical variables.
- The train-test split of 80%-20% was done using two helper functions: **train_test_split** and **df_feature_label_split**

2) Implementation of the Binary SVM classifier:

- The SVM classifier was trained using the **svm.SVC** method of the **sklearn library** by setting the argument **kernel** for the **svm.SVC** method as follows:
  1. **kernel = 'linear'** for Linear Kernel
  2. **kernel = 'poly' , degree = 2** for Quadratic Kernel
  3. **kernel = 'rbf'** for Radial Basis Function Kernel

- **Results of the SVM classifier for a random train-test split:**

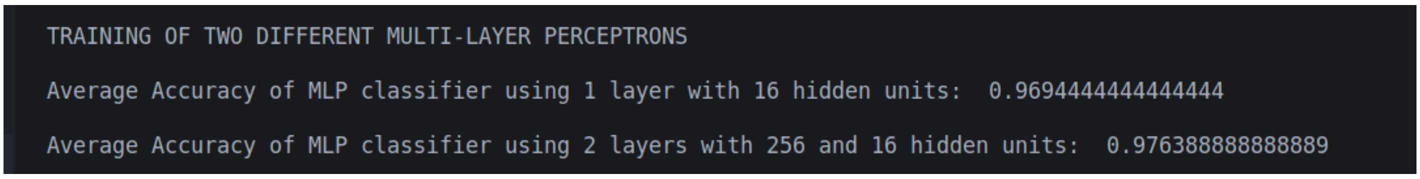


```
TRAINING OF BINARY SVM CLASSIFIERS WITH DIFFERENT KERNELS

Accuracy of SVM classifier using Linear Kernel:  0.9444444444444444

Accuracy of SVM classifier using Quadratic Kernel:  0.9166666666666666

Accuracy of SVM classifier using Radial Basis Function Kernel:  1.0
```

3) Implementation of the Multi-Layer Perceptron classifier:

- Two different MLP classifiers: (a) one hidden layer with 16 nodes and (b) two hidden layers with 256 and 16 nodes respectively were trained using **sklearn.neural_network.MLPClassifier** method.
- The hyperparameters are as follows:
    1. Max number of iterations: 3000
    2. Initial learning rate: 0.001
    3. Batch size: 32
    4. Optimizer: Stochastic gradient descent (‘sgd’)
- The training was repeatedly done for 20 times for both the MLP classifiers , with a different train-test split in each iteration. The MLP model which had higher accuracy than the other MLP model in maximum no of iterations out of these 20 iterations was considered to be the best MLP classifier. This was done since based on just one train-test split, it was difficult to judge which model has better accuracy as sometimes both had same accuracy for just one particular train-test split.
- However, over the 20 iterations , the two-hiddel layer MLP classifier performed better more no of times than the one hidden layer model.
- Based on these, the MLP model with two hidden layers with 256 and 16 nodes respectively is found to be the best model.

- **Results (Average Accuracy of the two MLP classifiers over the 20 iterations):**

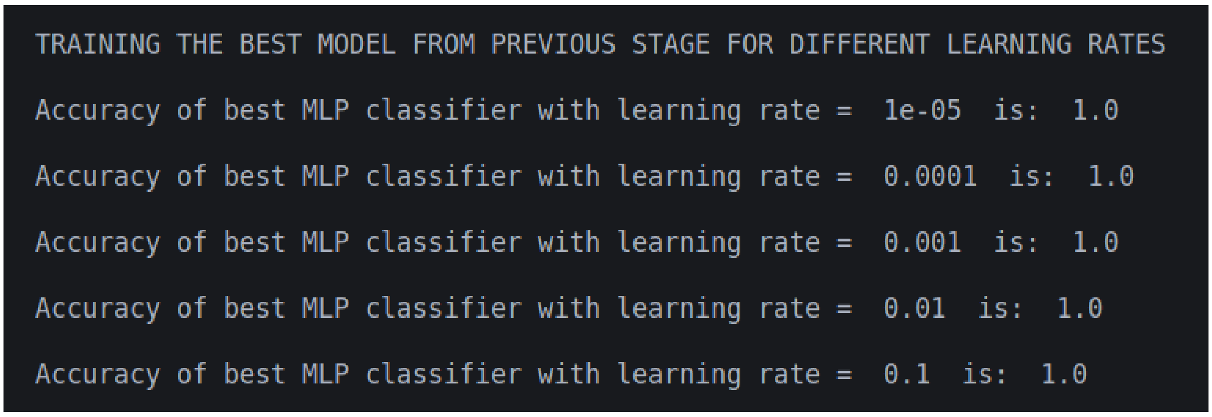


```
TRAINING OF TWO DIFFERENT MULTI-LAYER PERCEPTRONS

Average Accuracy of MLP classifier using 1 layer with 16 hidden units:  0.9694444444444444

Average Accuracy of MLP classifier using 2 layers with 256 and 16 hidden units:  0.976388888888889
```

4) Training the best MLP model for different learning rates:

- The best MLP model (with 2 hidden layers of 256 and 16 nodes) was trained by varying the initial rates as follows:
  Learning rates: [0.00001 0.0001 0.001 0.01 0.1]

Results:



```
TRAINING THE BEST MODEL FROM PREVIOUS STAGE FOR DIFFERENT LEARNING RATES

Accuracy of best MLP classifier with learning rate =  1e-05  is:  1.0

Accuracy of best MLP classifier with learning rate =  0.0001  is:  1.0

Accuracy of best MLP classifier with learning rate =  0.001  is:  1.0

Accuracy of best MLP classifier with learning rate =  0.01  is:  1.0

Accuracy of best MLP classifier with learning rate =  0.1  is:  1.0
```

- **Plots of loss versus no of iterations of the optimizer during training for the different learning rates:**

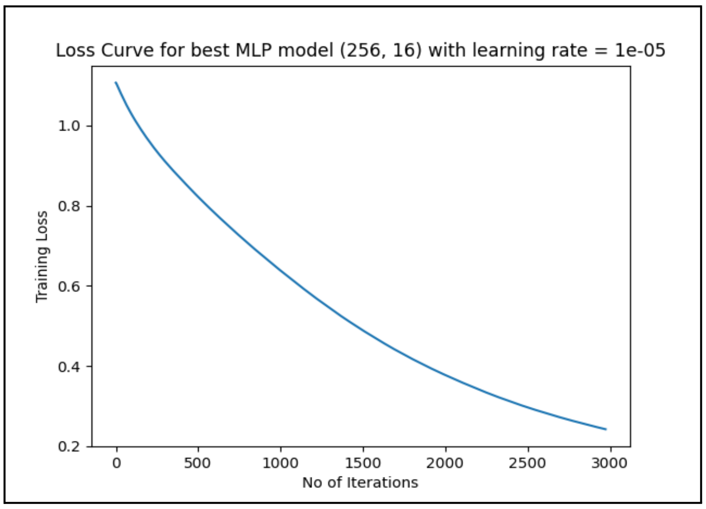


Fig 1. Loss with no of iterations of SGD for initial learning rate = 0.00001

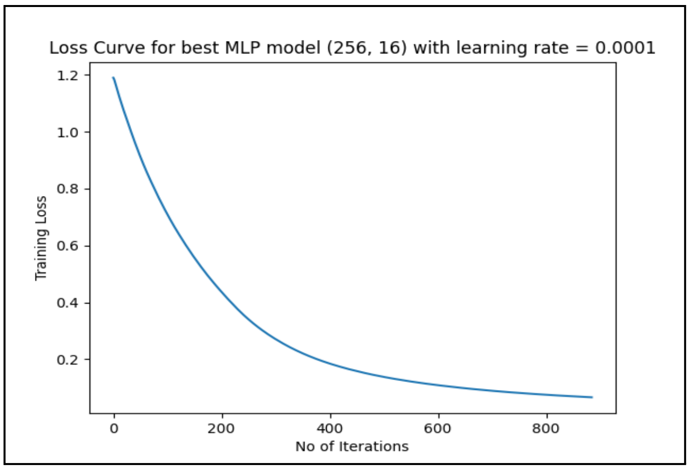


Fig 2. Loss with no of iterations of SGD for initial learning rate = 0.0001

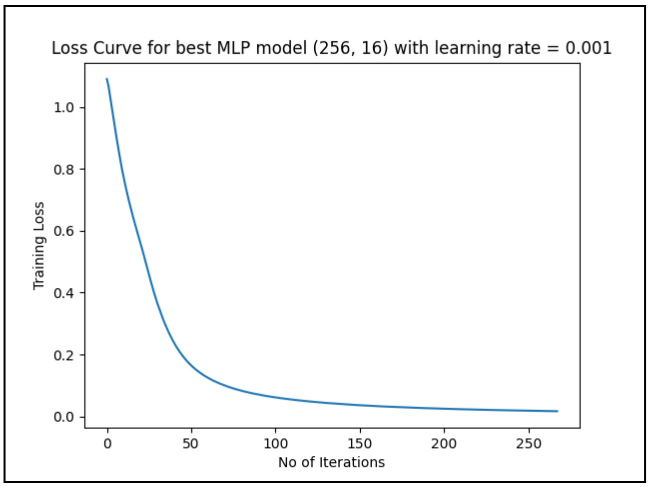


Fig 3. Loss with no of iterations of SGD for initial learning rate = 0.001

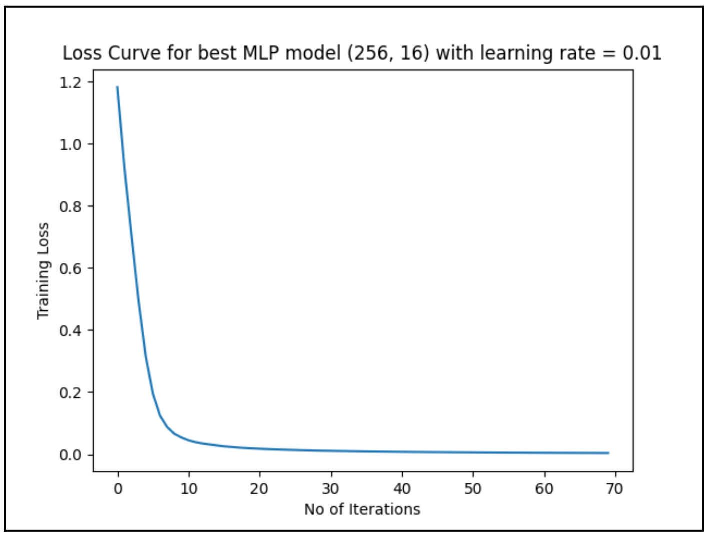


Fig 4. Loss with no of iterations of SGD for initial learning rate = 0.01

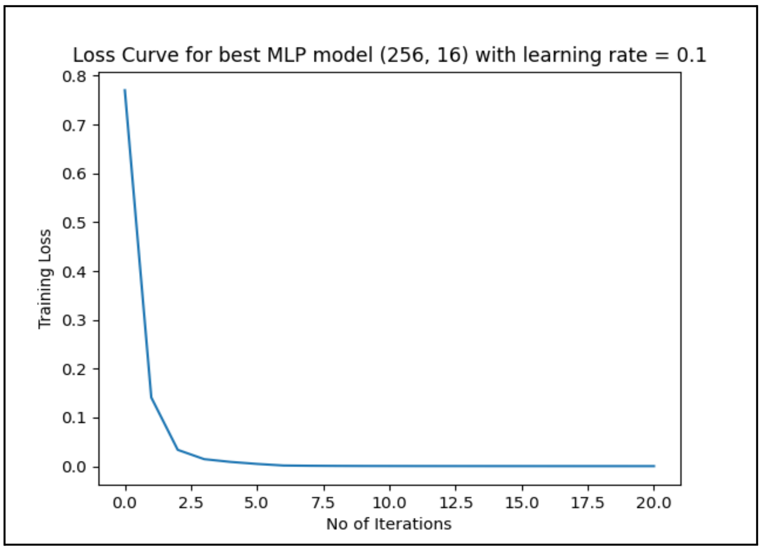


Fig 5. Loss with no of iterations of SGD for initial learning rate = 0.1

- **Plot comparing the final test accuracies of the (256, 16) MLP for the different initial learning rates**

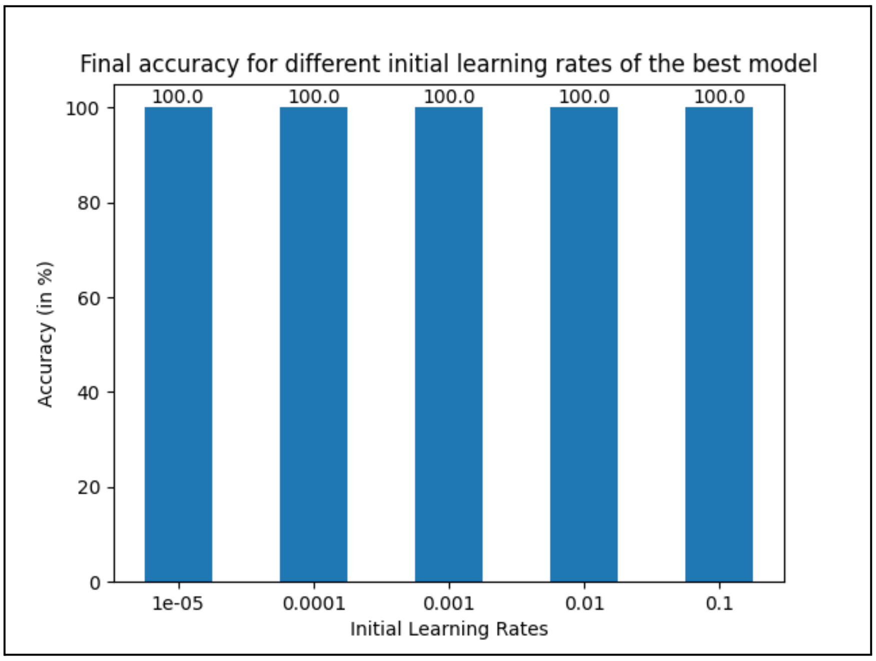
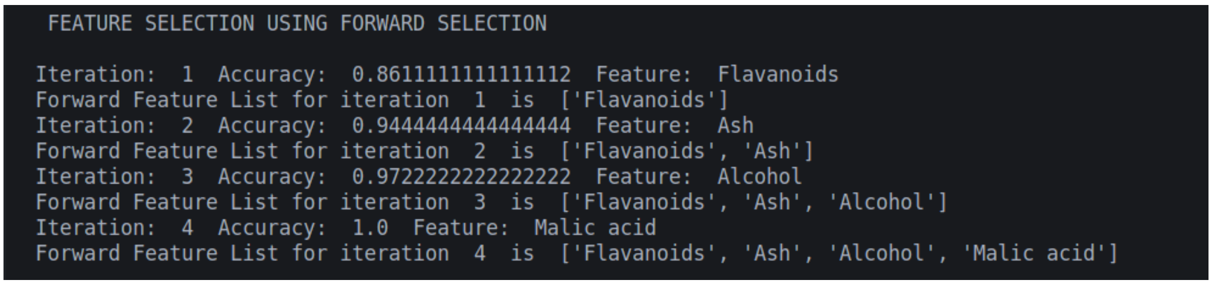


Fig 6. Bar plot comparing the final test accuracy (in %) of the best MLP model (256, 16) for different initial learning rates

5) Feature Selection using Forward Selection Approach:

- The forward selection was implemented from scratch without using any inbuilt library function. The process stopped if the addition of a new attribute didn't increase the accuracy significantly. Also, the process is stopped once the addition of a new attribute leads to an accuracy of 100% over the test set.

- **Results of Forward Selection:**



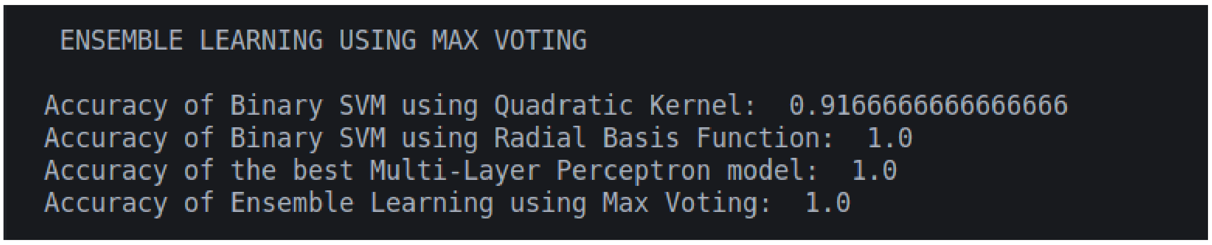
```
 FEATURE SELECTION USING FORWARD SELECTION

Iteration:  1  Accuracy:  0.8611111111111112  Feature:  Flavanoids
Forward Feature List for iteration  1  is  ['Flavanoids']
Iteration:  2  Accuracy:  0.9444444444444444  Feature:  Ash
Forward Feature List for iteration  2  is  ['Flavanoids', 'Ash']
Iteration:  3  Accuracy:  0.9722222222222222  Feature:  Alcohol
Forward Feature List for iteration  3  is  ['Flavanoids', 'Ash', 'Alcohol']
Iteration:  4  Accuracy:  1.0  Feature:  Malic acid
Forward Feature List for iteration  4  is  ['Flavanoids', 'Ash', 'Alcohol', 'Malic acid']
```

6) Ensemble Learning using Max Voting:

- The ensemble learning with **Max Voting** was done for the following classifier models trained previously:
    1.) SVM Classifier using Quadratic Kernel
    2.) SVM Classifier using Radial Basis Function Kernel
    3.) Best MLP Classifier from part 3: MLP classifier with 256 nodes in the first hidden layer and 16 nodes in the 2nd hidden layer

- Since all three models have comparable performance over the dataset, Max Voting is a viable choice here.

- **Results of Max Voting Ensemble Learning:**



```
 ENSEMBLE LEARNING USING MAX VOTING

Accuracy of Binary SVM using Quadratic Kernel:  0.9166666666666666
Accuracy of Binary SVM using Radial Basis Function:  1.0
Accuracy of the best Multi-Layer Perceptron model:  1.0
Accuracy of Ensemble Learning using Max Voting:  1.0
```

- For this particular run of the code, we got an accuracy of 100 % in the Ensemble Learning via Max Voting, however, quite often we also got an accuracy of Max Voting lower than that of the MLP classifier individually in many runs of code.