Data Mining Project Report

As a first step, I have imported all the libraries required to do the assignment tasks at the top.

Part I

To do this task of building a Neural Network Classifier and finetuning it, following steps are followed:

Step 1: Importing the data from csv file

Step 2: Data Pre-Processing: Data has duplicated records and NAs, which need to be dealt with. Also, we need to do encoding for categoricals.

- Check for duplicated records:
  After checking the existence of duplicated records, a total of 73 records were removed.
- Check for NAs:
  After checking for NAs, four columns "Buy", "Maint_costs", "person" and "Boot" came up with missing values.
  - For "Person" column there are 554 attributes with missing values, which is about 33% of the total records. Imputing so much values can create a bias in the data. So, it is better to remove the whole column from the analysis.
  - For "Buy", "Main_cost" and "Boot" columns, it is better to remove the corresponding records with empty values. As such imputing categoricals can result in misrepresentation, resulting in bad predictions.

  After Dealing with NAs, the final dataframe is of shape: (1558, 6)

- Transforming the Categoricals:
  As “Quality” column is of interest and class label, transforming it manually. (Moreover, transforming to (0, 1, 2, 3) matches with indexes of Cost Matrix)

After Data Preprocessing is complete, the pre-processed data is exported to a csv file.

Step 3: Preparing data for use in the model

- Defining target(class label) and input variables
- Splitting the data into training and testing sets in ratio 75% to 25%
- Defining Cost Matrix

Step 4: Defining and evaluating model

- Model 1:
  Model 1 is defined with following setting:
  - Hidden layer and neurons: (120)
  - Maximum iterations: 200
  - Learning Rate: 0.001
  - Solver: ‘sgd’
  - random state: 1

  Total Cost is the primary evaluation criteria. So the target for fine tuning is to minimize this total cost of misclassification.
  After running the model Total Cost = 234.

- Model 2: Finetuning No. of Neurons
  Testing 4 different settings for Neurons as 30, 60, 180, and 240; and compare it with model 1.
  After testing, changing the no. of neurons to 30, decreases the cost.
  So, we can take 30 as neurons for further fine tuning.
  Model 2 is defined with following setting:
  - Hidden layer and neurons: (30)
  - Maximum iterations: 200
  - Learning Rate: 0.001
  - Solver: ‘sgd’
  - random state: 1

  After running the model Total Cost = 233.

- Model 3: Finetuning No. of hidden layers
  Testing 4 different settings for layers as (30, 30), (60, 30), (60, 60) and (120, 60, 30); and comparing it with model 2.
  After testing, changing the layers doesn't improve our primary evaluation metric of total cost. So, we will continue with model 2 for further finetuning.
  For illustration, a model with different layer is shown with following setting:
  - Hidden layer and neurons: (60, 30)
  - Maximum iterations: 200
  - Learning Rate: 0.001
  - Solver: ‘sgd’
  - random state: 1

  After running the model Total Cost = 234. As it is more than model 2, our preferred model is 2.

- Model 4: Finetuning No. of max iterations on model 2

  Testing 4 different settings for max iterations as 200, 400, 600, 800; and comparing it with model 2.

  After testing, changing the max iterations to 800, decreases our primary evaluation metric total cost t. So, we will consider settings with 800 maximum iterations for further finetuning.

  Model 4 is defined with following setting:
  - Hidden layer and neurons: (30)
  - Maximum iterations: 800
  - Learning Rate: 0.001
  - Solver: ‘sgd’
  - random state: 1

  After running the model Total Cost = 210

- Model 5: Finetuning learning rate on model 4

  Testing 4 different settings for learning rate as 0.002, 0.004, 0.006, 0.008; and comparing it with model 4.

  After testing, changing the learning rate to 0.002 decreases our primary evaluation criteria total cost. So, we will consider settings with learning rate 0.002 for further fine tuning.

  Model 5 is defined with following setting:
  - Hidden layer and neurons: (30)
  - Maximum iterations: 800
  - Learning Rate: 0.002
  - Solver: ‘sgd’
  - random state: 1

  After running the model Total Cost = 197

- Model 6: Finetuning optimizer

  Testing the other optimizer setting of 'adam' and comparing it with model 5 which is the best model till now.

  After testing, changing the optimizer doesn't decrease our primary evaluation metric total cost. So, we will consider the settings of model 5 as the best.

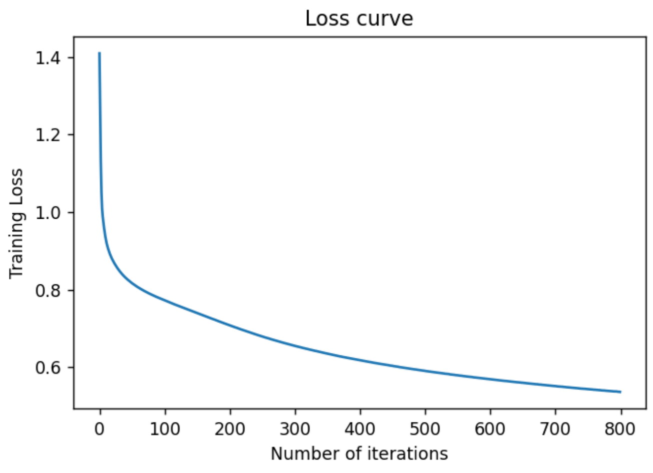
  Model 5 is defined with following setting:
  - Hidden layer and neurons: (30)
  - Maximum iterations: 800
  - Learning Rate: 0.002
  - Solver: ‘adam’
  - random state: 1

  After running the model Total Cost = 241. As it is more than model 5, our preferred model is 5.

Step 5: Describing the best model: Model 5

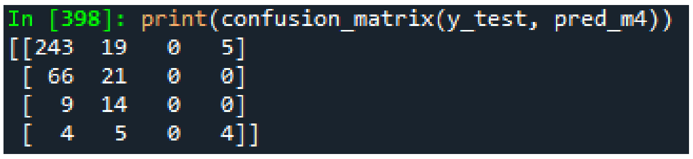
The selection of Model 5 is based only on the primary evaluation method of total cost.

- Loss Curve:



  Loss curve shows how the model loss decreases as the iteration increases. At 800th iteration, the loss for model 5 is below 0.6.

- Confusion Matrix:



```
In [398]: print(confusion_matrix(y_test, pred_m4))
[[243  19   0   5]
 [ 66  21   0   0]
 [  9  14   0   0]
 [  4   5   0   4]]
```

  Based on the confusion matrix, the model has predicted a total of 268 values (243+21+0+6) correctly.

Comparative analysis Evaluation Results

Model	**Cost**	**Loss**	**Accuracy**	**Senstivity**	**UAR**	**AUC**
Model 1	234	0.768	0.684	0.25	0.171	0.627
Model 2	233	0.771	0.687	0.253	0.422	0.648
Model 3	234	0.764	0.685	0.25	0.171	0.644
Model 4	210	0.617	0.687	0.365	0.389	0.788
Model 5	**197**	0.536	0.7	0.431	0.414	0.853
Model 6	241	**0.412**	**0.762**	**0.696**	**0.635**	**0.932**

If we see a comparative analysis of the models, just on the basis of total cost of misclassification, we can say that Model 5 is the best.
But all other evaluation metrics suggest Model 6 to be the best.
This may be primarily due to the fact that Model 6 misclassifies the higher quality cars, which have high cost of misclassification.

Part II next Page

Part II

To do this task, I have followed the following steps:

Step 1: Imported the dataset

Step 2: Defining labels and features

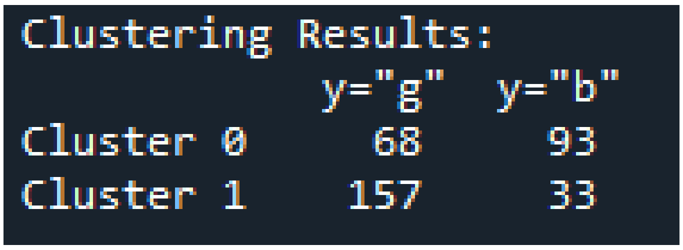
Step 3: Transform the labels to 0 for "g" and 1 for "b"

Step 4: Defining clusters

- Defining the model with 2 clusters
- Training the model with features
- Predicting the labels

Step 5: Counting and printing Clustering Results

- Added the predicted values as new column to the dataframe
- Printed the required results in the required format



```
Clustering Results:
             y="g"  y="b"
Cluster 0      68     93
Cluster 1     157     33
```