Green University of Bangladesh
Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering
Semester: (Spring, Year: 2024), B.Sc. in CSE (Day)

Lab Report NO 04
Course Title: Artificial Intelligence Lab
Course Code: CSE 316 Section: 213_D4

Lab Experiment Name: Run Neural Network classification algorithm and evaluate the results of the method on a data file. Also make training model and use the training model to evaluate a testing file and compare accuracy values for different parameters.

Student Details

	Name	ID
1.	Md.Rabby Khan	213902037

Lab Date : 14-05-24
Submission Date : 12-06-24
Course Teacher's Name : Md.Fahimul Islam

[For Teachers use only: Don't Write Anything inside this box]

Lab Report Status
Marks: …… **Signature:.....**
Comments:..... **Date:.....**

1. TITLE OF THE LAB EXPERIMENT

Run Neural Network classification algorithm and evaluate the results of the method on a data file. Also make training model and use the training model to evaluate a testing file and compare accuracy values for different parameters.

2. OBJECTIVES/AIM

The objective of the experiment is,

- To understand WEKA tool usage to analysis training data.
- To perform data analysis in WEKA.
- Implement a Neural Network classification algorithm to train and evaluate a model on a given dataset.
- To learn about dataset splitting method and apply Neural Network classify.
- Compare accuracy values of the trained model on a testing dataset with varying parameters to determine optimal performance.

3. PROCEDURE / ANALYSIS / DESIGN

Running Neural Network classification algorithm and evaluation consists of following procedure.

Step 1: Open WEKA
Step 2: Click on explorer button
Step 3: Click on choose file button
Step 4: Select a dataset and open in WEKA
Step 5: Switch the UI from preprocess to classify by clicking on classify
Step 6: Choose MultilayerPercenption algorithm.
Step 7: Click on start button to evaluate the algorithm.
Step 8: Click on the selected algorithm to make changes for showing GUI
Step 9: Set the GUI value to True

Also making the training model and use the training model to evaluate a testing file consists of following steps.

Step 1: Change the tab into preprocess
Step 2: Click on choose button to select a randomizer algorithm
Step 3: Select randomize algorithm
Step 4: Click on apply
Step 5: Click on choose to select splitter method *RemovePercentage*

Step 6: Click on the input field and set the value of percentage to 80

Step 7: Click on apply to remove 80% of the dataset

Step 8: Click on save as a testing set.

Step 9: Repeat the steps 1 to 5 for making the training set.

Step 10: Click on input field and change the invertSelection value to True.

Step 11: Click ok and apply the changes.

Step 12: Observe the changes in the details for instances.

Step 13: Change the tab into classify.

Step 14: Selecting the use training set click on start to evaluate the function.

Step 15: Click on supplied testing set and click on the set button.

Step 16: Select the testing set created earlier and click on start button to evaluate.

4. IMPLEMENTATION

4.1 Task 01: Run Neural Network classification algorithm and evaluate the results of the method on a data file.

Figure 4.1.1: Click on the explorer button in WEKA.

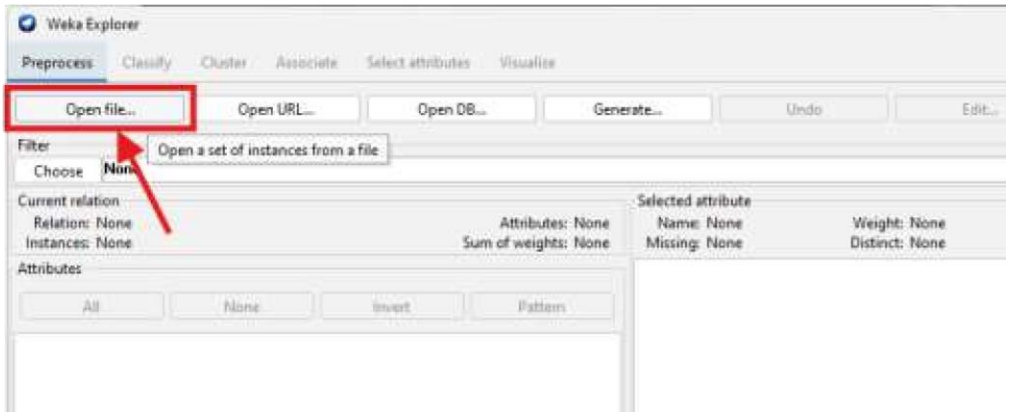

Figure 4.1.2: Click on the open file button to select a dataset.

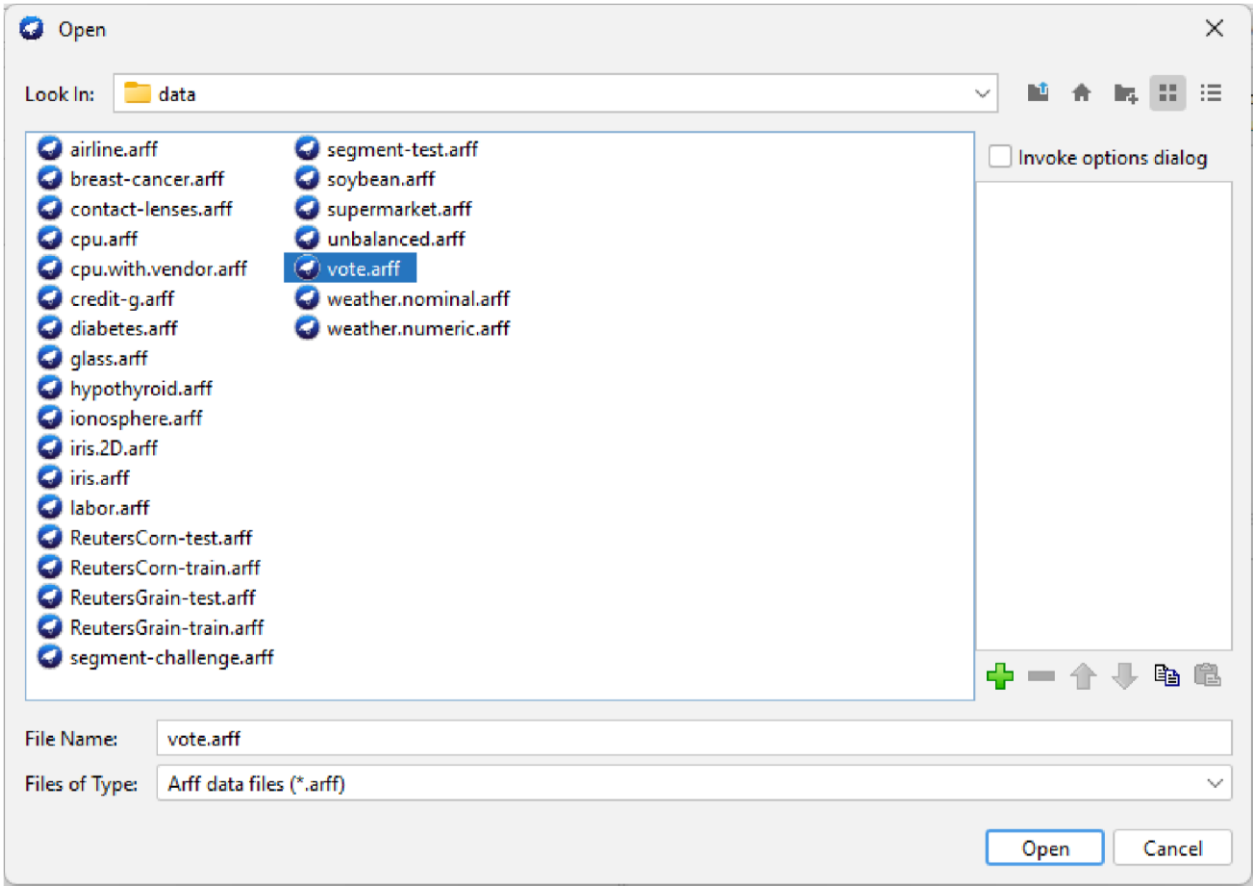

Figure 4.1.3: Select a dataset from directory and click open.

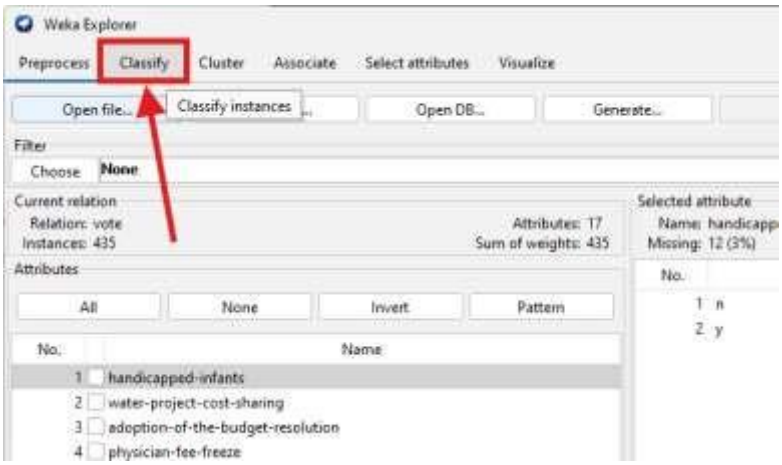

Figure 4.1.4: Click on classify button to apply the classification algorithms.

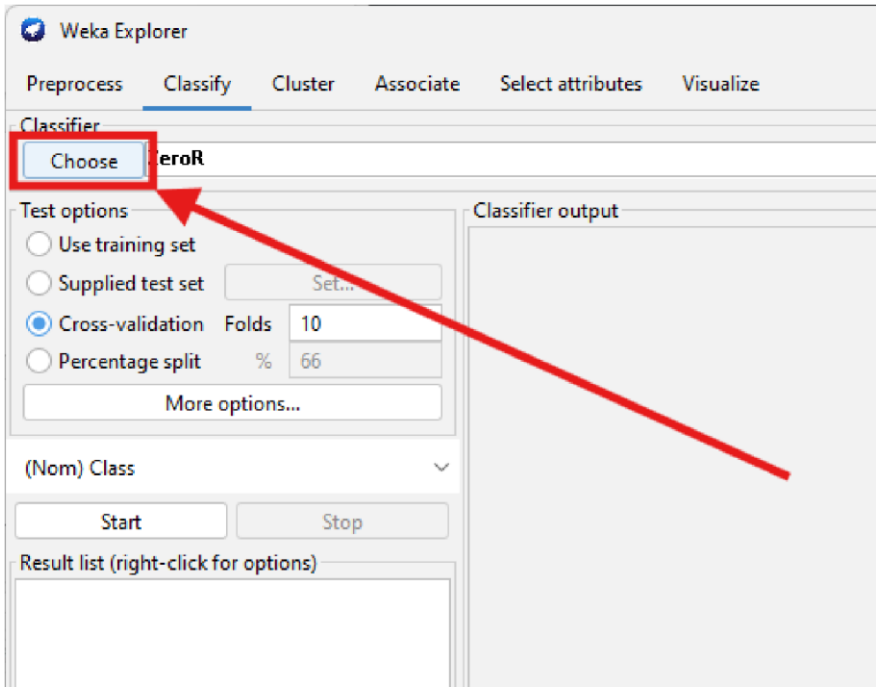

Figure 4.1.5: Click on choose button to choose the classification algorithm.

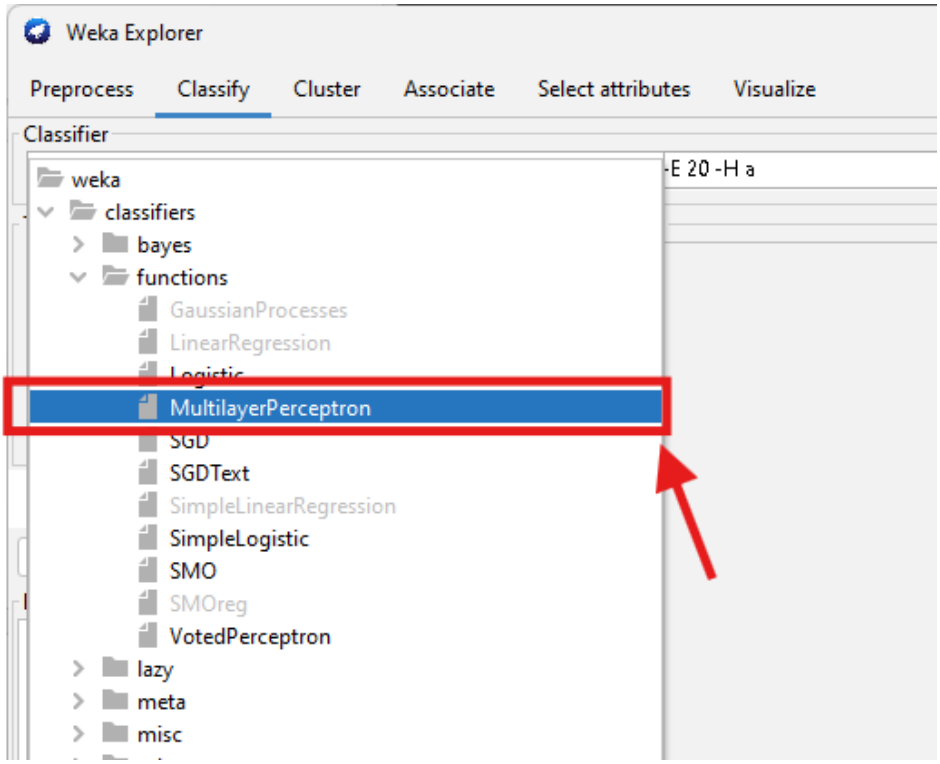

Figure 4.1.6: Select MultilayerPerception algorithm.

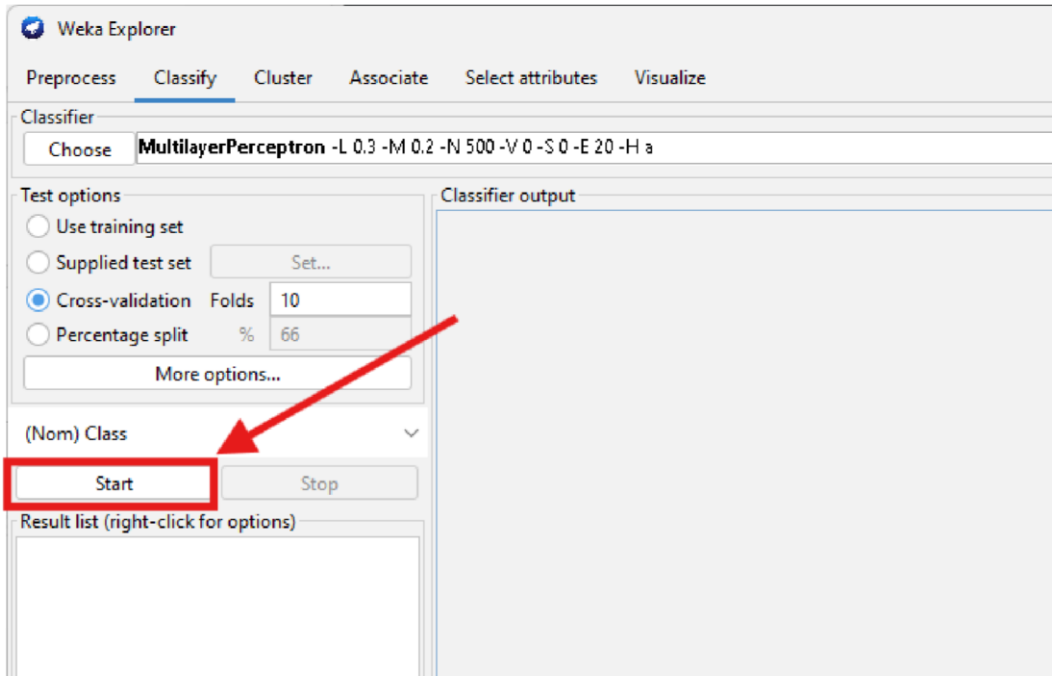

Figure 4.1.7: Click on start button to apply multilayer perception algorithm.

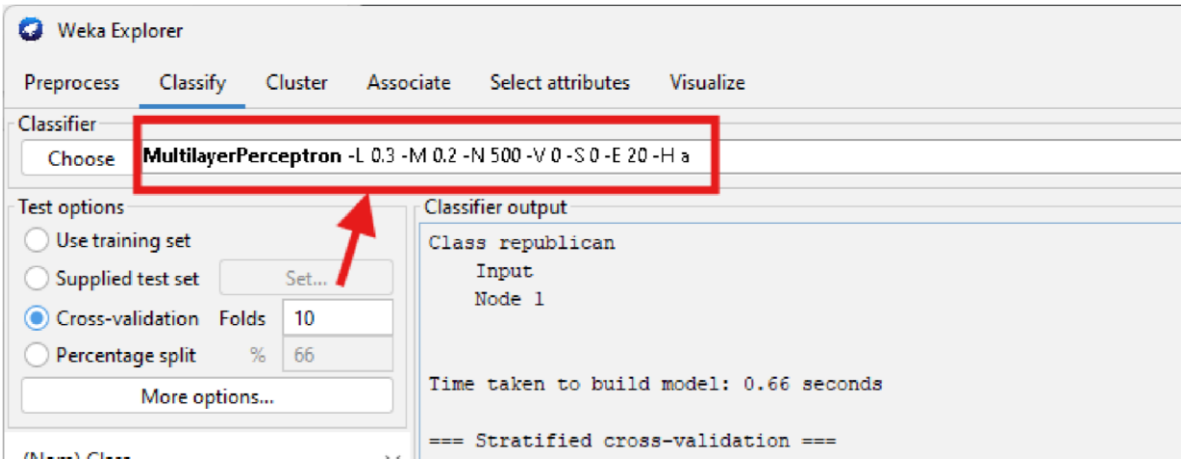

Figure 4.1.8: Click on the input field besides the choose button.

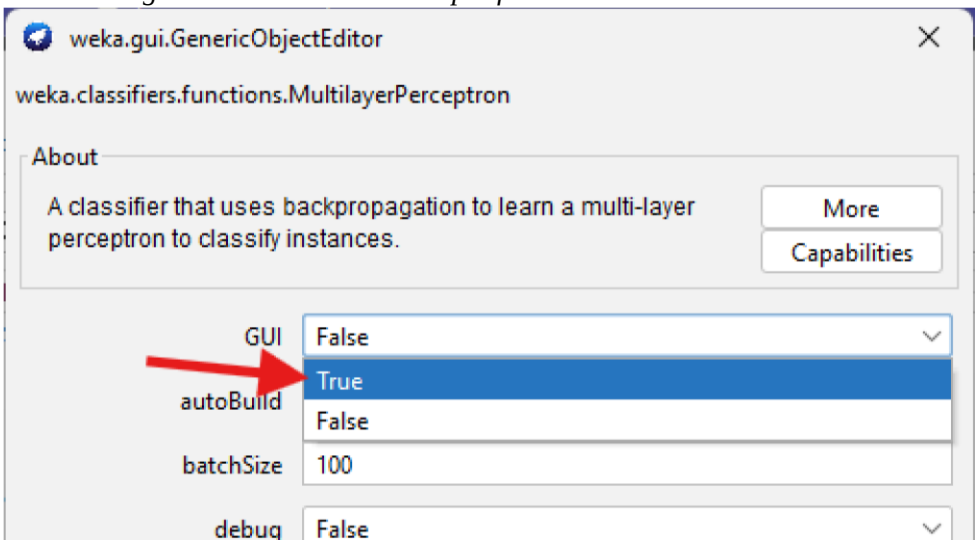

Figure 4.1.9: Change the GUI checkbox from false to true.

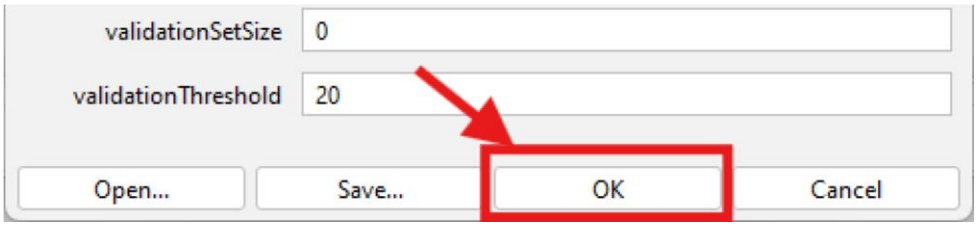

Figure 4.1.10: Click on ok to make the changes to be applied in WEKA.

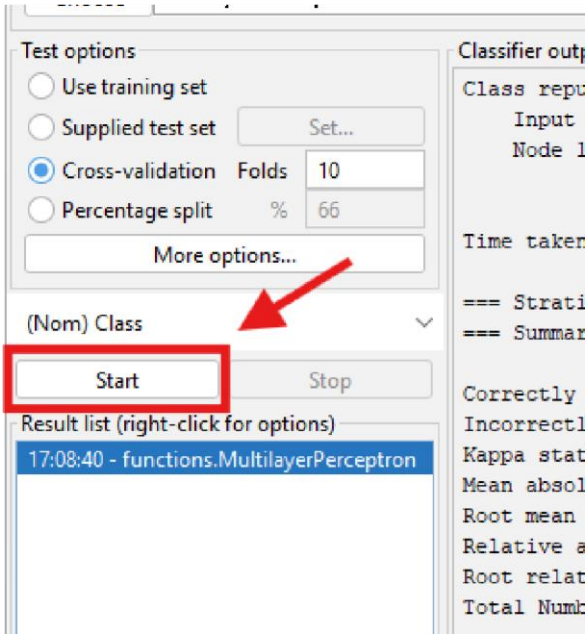

Figure 4.1.11: Click on start button to run the Neural Network.

4.2 Task 02: make training model and use the training model to evaluate a testing file and compare accuracy values for different parameters

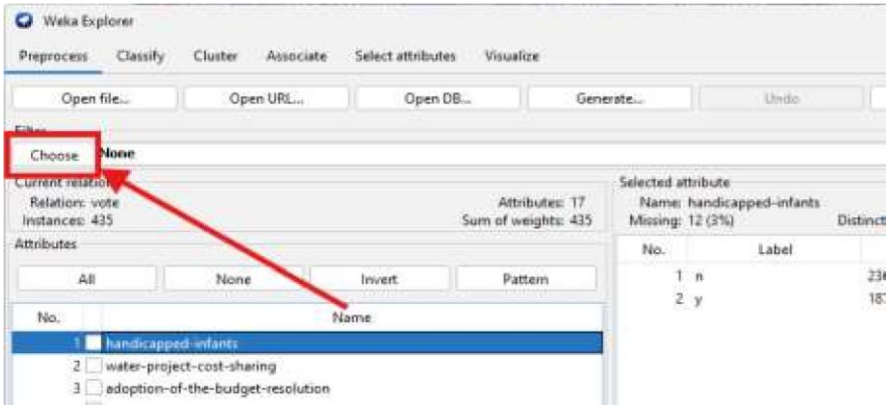

Figure 4.2.1: Click on choose button to apply algorithm.

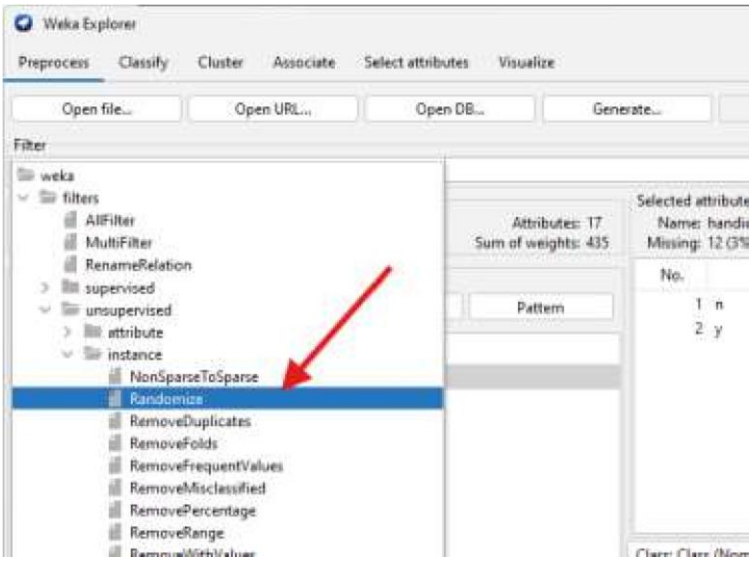

Figure 4.2.2: Select the unsupervised > instance > randomize to make the dataset random.

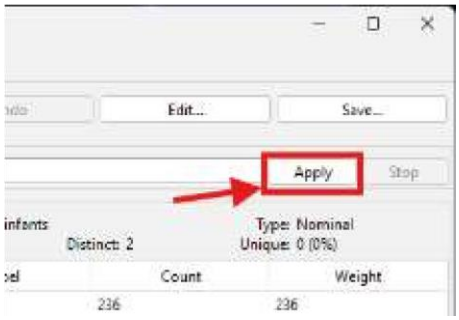

Figure 4.2.3: Click on apply to make changes.

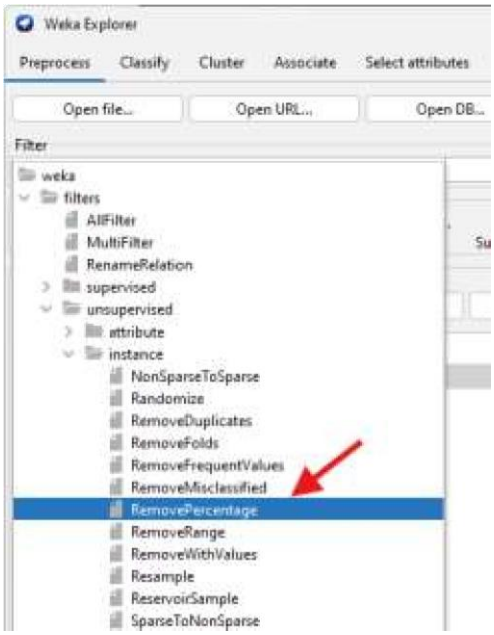

Figure 4.2.4: Select RemovePercentage algrotihm.

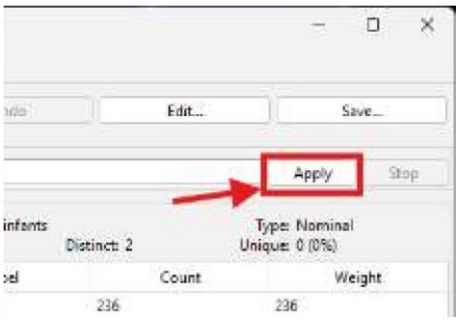

Figure 4.2.5: Click on apply to make changes.

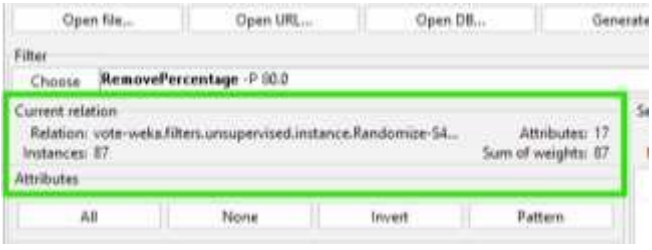

Figure 4.2.6: Check the changes in the instances.

Figure 4.2.7: Click on save button to save the testing set.

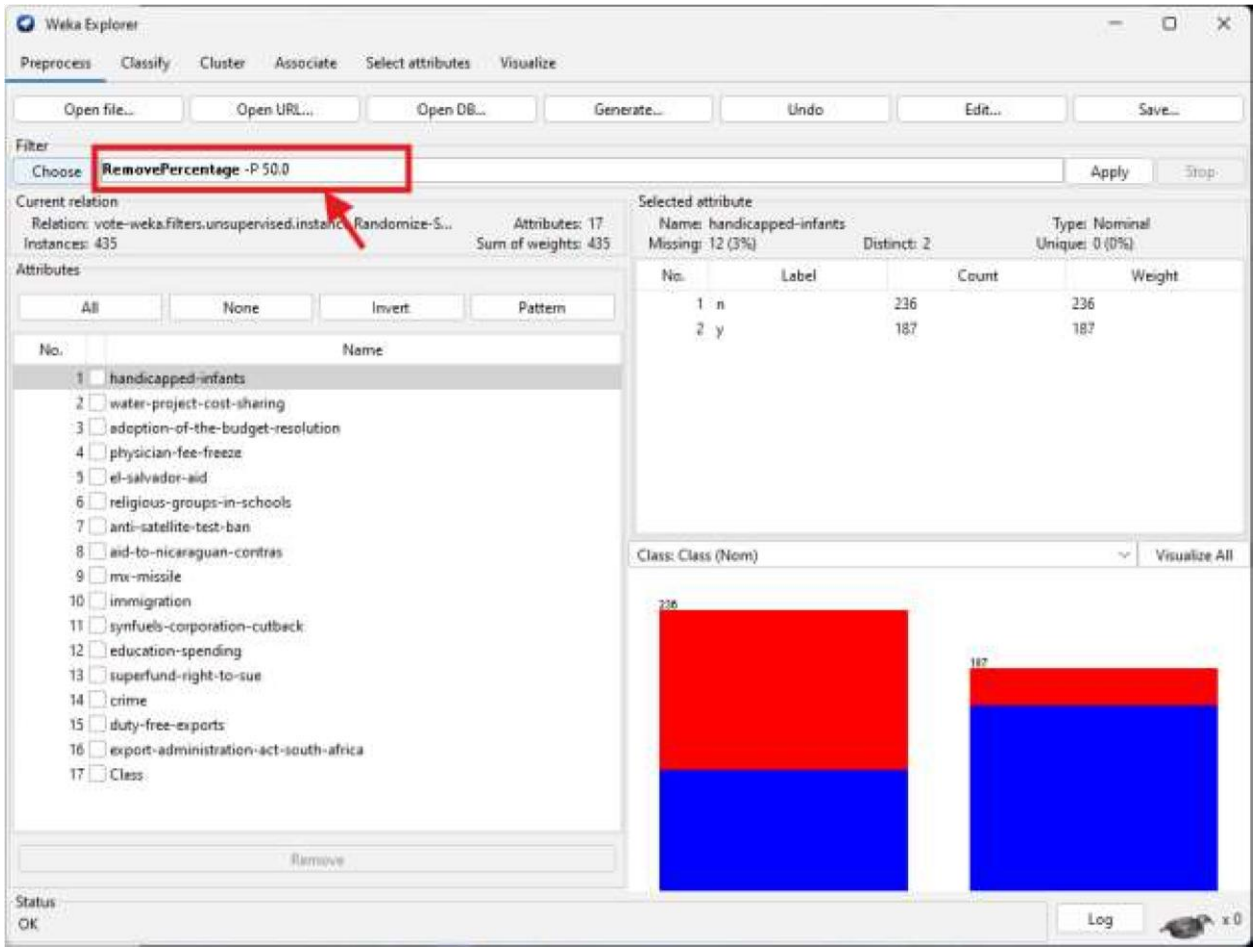

Figure 4.2.8: Click on the input field.

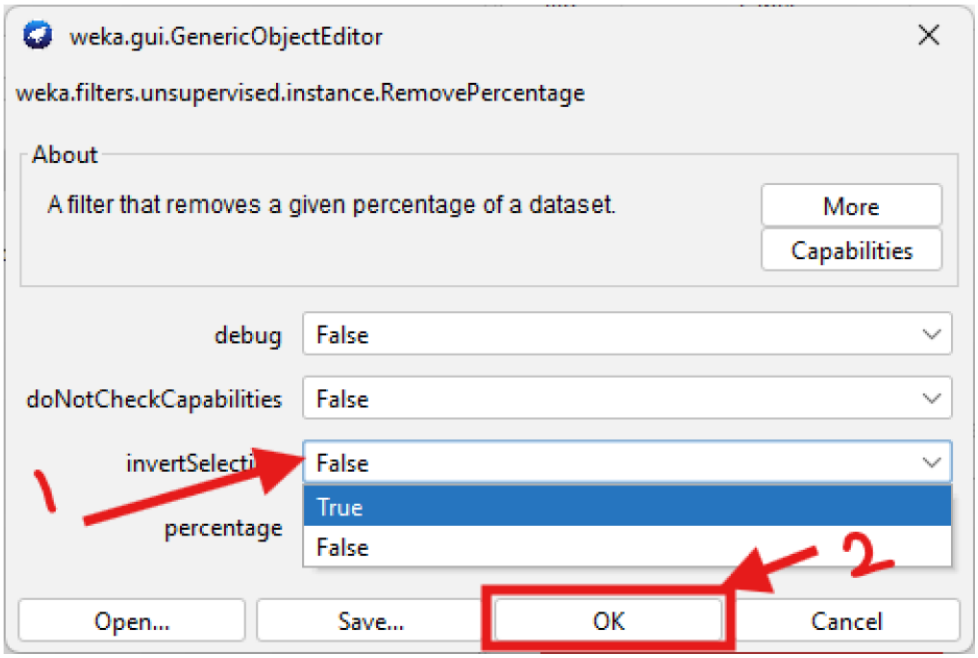

Figure 4.2.9: Change the invert selection into True.

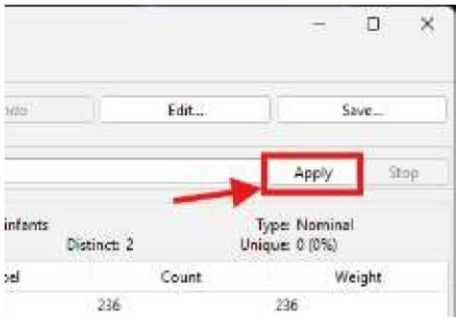

Figure 4.2.10: Click on apply to make changes.

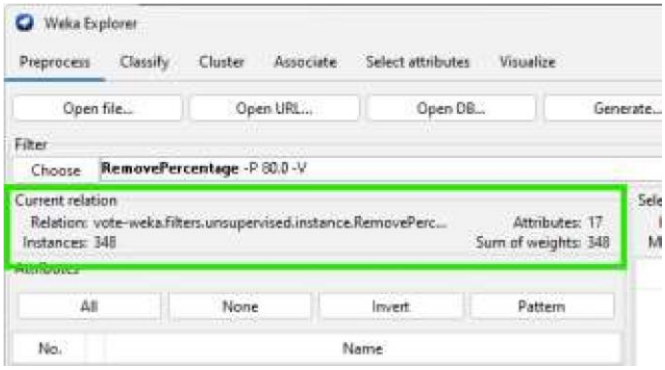

Figure 4.2.11: Check the changes in the instances.

Figure 4.2.12: Click on save button to save the training set.

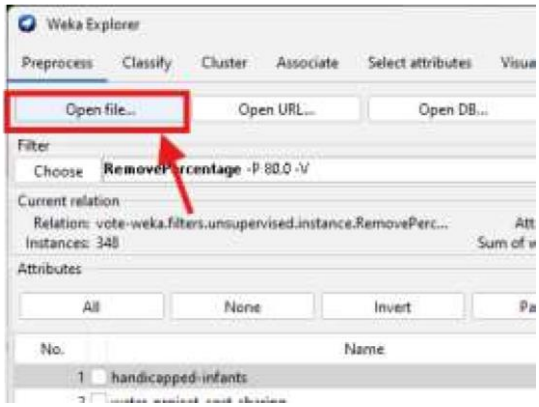

Figure 4.2.13: Click on open file button to open the training set.

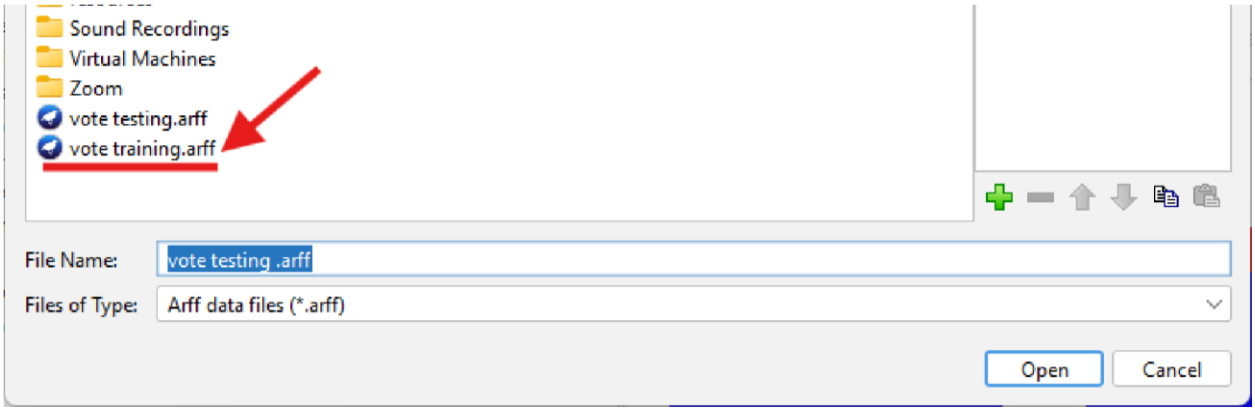

Figure 4.2.14: Select training set and open in WEKA.

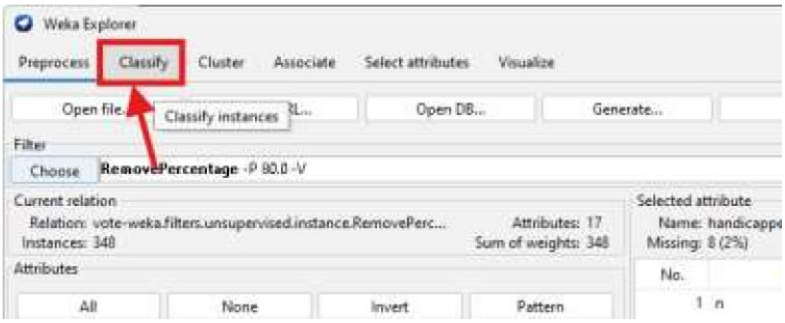

Figure 4.2.15: Change the tab to classify.

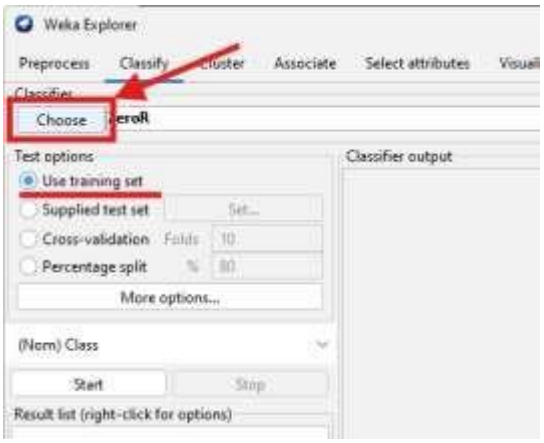

Figure 4.2.16: Click on choose to select the algorithm.

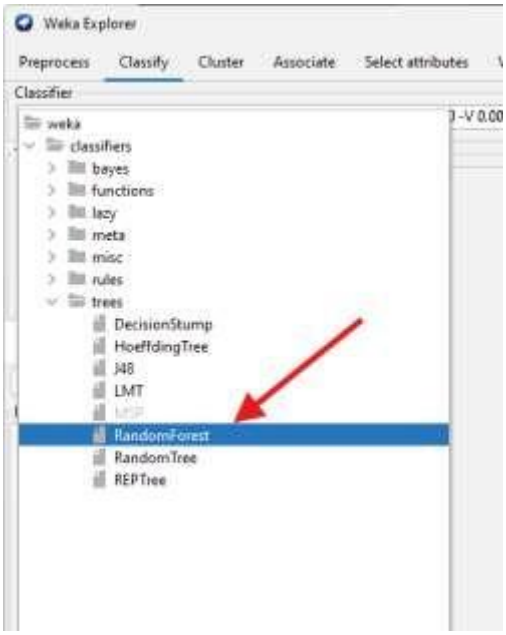

Figure 4.2.17: Select the RandomForest algorithm.

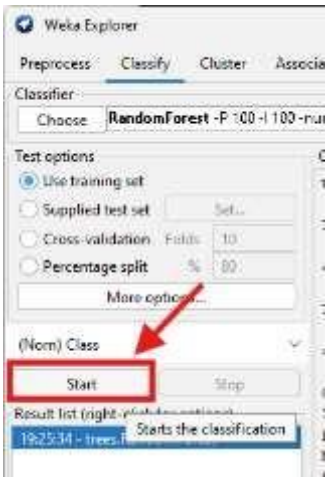

Figure 4.2.18: Click on start button to evaluate the training set.

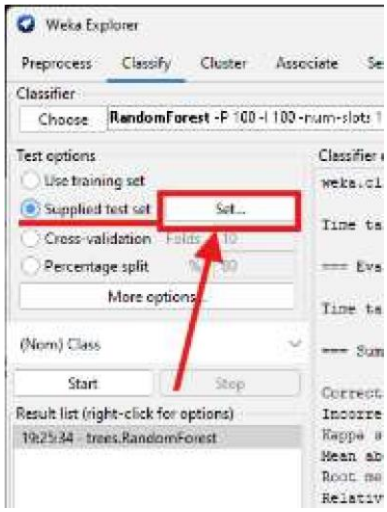

Figure 4.2.19: Select supplied test set and click on the set button.

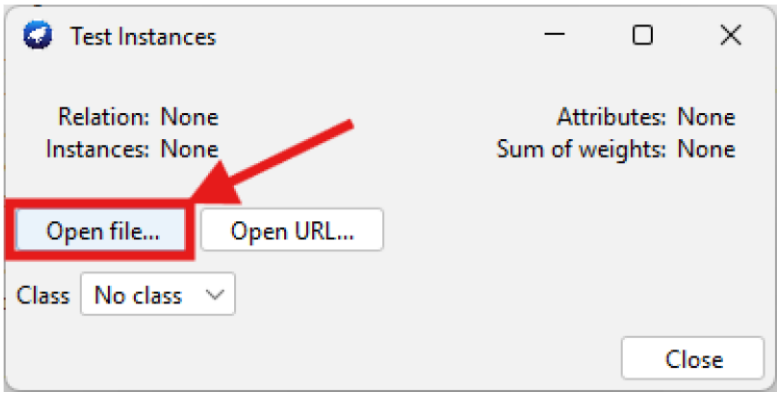

Figure 4.2.20: Click on open file button.

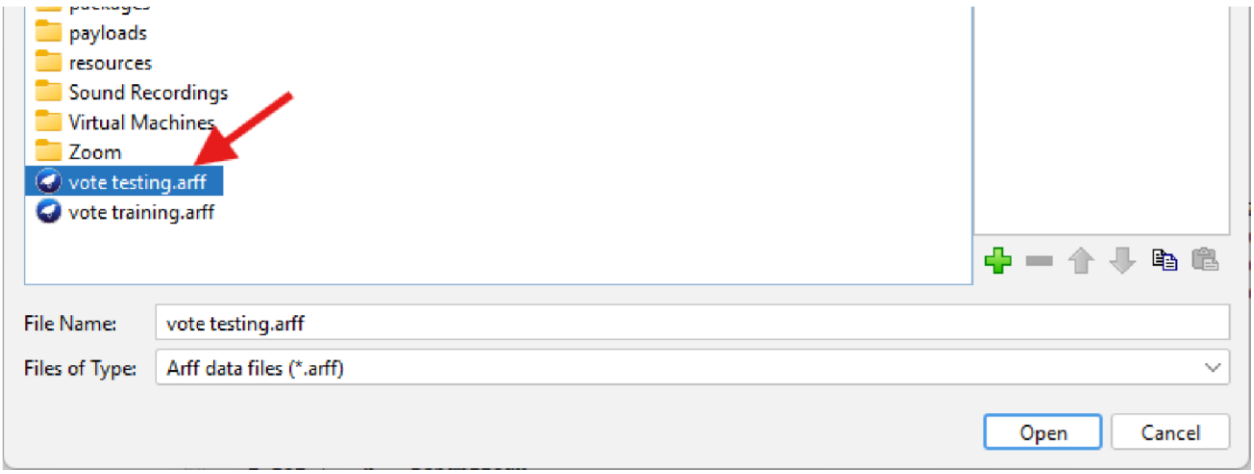

Figure 4.2.21: Select the testing set and click open.

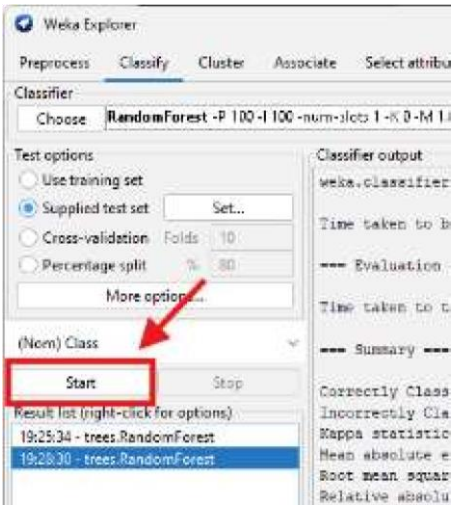

Figure 4.2.22: Click on start button to evaluate the function.

5. TEST RESULT/ OUTPUT

5.1 Task 1 Result/Output

```
Time taken to build model: 0.66 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         412               94.7126 %
Incorrectly Classified Instances        23                5.2874 %
Kappa statistic                          0.8888
Mean absolute error                      0.0528
Root mean squared error                  0.2078
Relative absolute error                 11.135  %
Root relative squared error             42.6788 %
Total Number of Instances              435

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.951    0.060    0.962      0.951    0.957      0.889    0.991     0.995     democrat
                 0.940    0.049    0.924      0.940    0.932      0.889    0.991     0.984     republican
Weighted Avg.    0.947    0.055    0.947      0.947    0.947      0.889    0.991     0.991

=== Confusion Matrix ===

   a   b   <-- classified as
 254  13 |   a = democrat
  10 158 |   b = republican
```

Figure 5.1.1: Output of the summary is shown as well as the confusion matrix for the dataset.

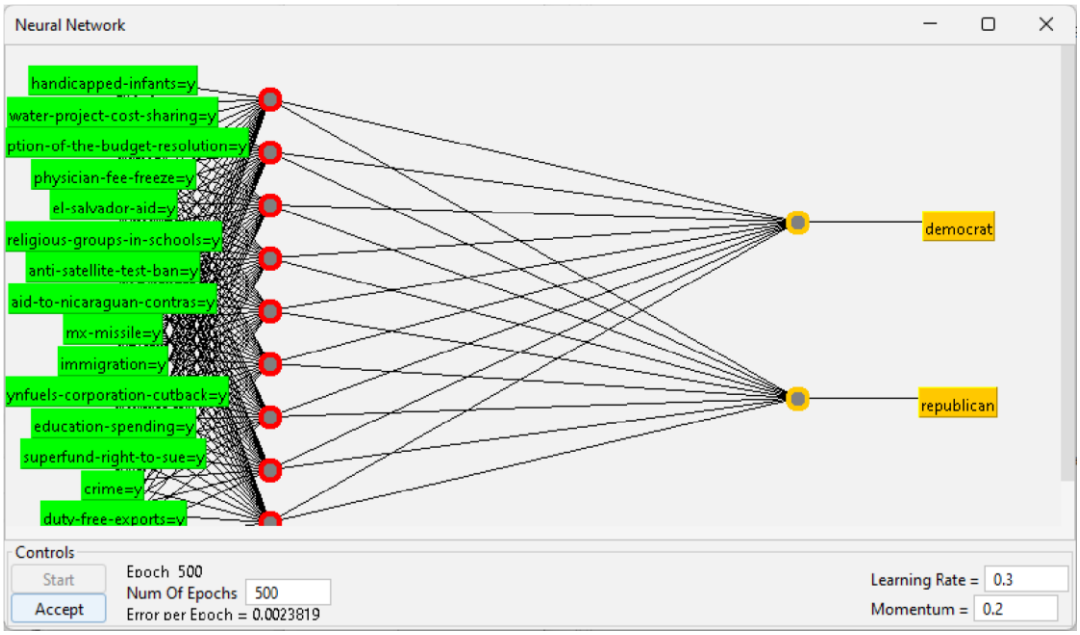

Figure 5.1.2: Output of the neural network.

5.2 Task 2 Result/Output

```
Time taken to build model: 0.11 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0.02 seconds

=== Summary ===

Correctly Classified Instances         346               99.4253 %
Incorrectly Classified Instances         2                0.5747 %
Kappa statistic                          0.9878
Mean absolute error                      0.0295
Root mean squared error                  0.0816
Relative absolute error                  6.2366 %
Root relative squared error             16.7739 %
Total Number of Instances              348

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.015    0.991      1.000    0.995      0.988    1.000     1.000     democrat
                 0.985    0.000    1.000      0.985    0.992      0.988    1.000     1.000     republican
Weighted Avg.    0.994    0.009    0.994      0.994    0.994      0.988    1.000     1.000

=== Confusion Matrix ===

   a   b   <-- classified as
 214   0 |   a = democrat
   2 132 |   b = republican
```

Figure 5.2.1: Click on save button to save the training set.

```
Time taken to build model: 0.08 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0 seconds

=== Summary ===

Correctly Classified Instances          82               94.2529 %
Incorrectly Classified Instances         5                5.7471 %
Kappa statistic                          0.8812
Mean absolute error                      0.1237
Root mean squared error                  0.2507
Relative absolute error                 26.0333 %
Root relative squared error             51.3768 %
Total Number of Instances               87

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.925    0.029    0.980      0.925    0.951      0.883    0.968     0.983     democrat
                 0.971    0.075    0.892      0.971    0.930      0.883    0.968     0.945     republican
Weighted Avg.    0.943    0.047    0.946      0.943    0.943      0.883    0.968     0.968

=== Confusion Matrix ===

  a  b   <-- classified as
 49  4 |  a = democrat
  1 33 |  b = republican
```

Figure 5.2.2: Click on save button to save the training set.

6. ANALYSIS AND DISCUSSION

From the result of the given task, we have utilized the algorithm of *MultilayerPerceptron* to build and evaluate a neural network classification model. Firstly, we trained the model on a given training dataset, tuning parameters such as learning rate, momentum, and the number of hidden layers. The model was evaluated using cross-validation, yielding an accuracy of 94.7126%.

Next, we saved the trained model and applied it to a separate testing dataset to assess its generalizability. The testing accuracy was 94.2529%, indicating how well the model performs on unseen data. We observed that adjusting parameters like the number of epochs and the network structure significantly impacted the model's performance. In the case of visualizing the neural network provided insights into its complexity and learning process.

Overall, this process demonstrated the importance of parameter tuning and validation techniques in developing robust neural network models.