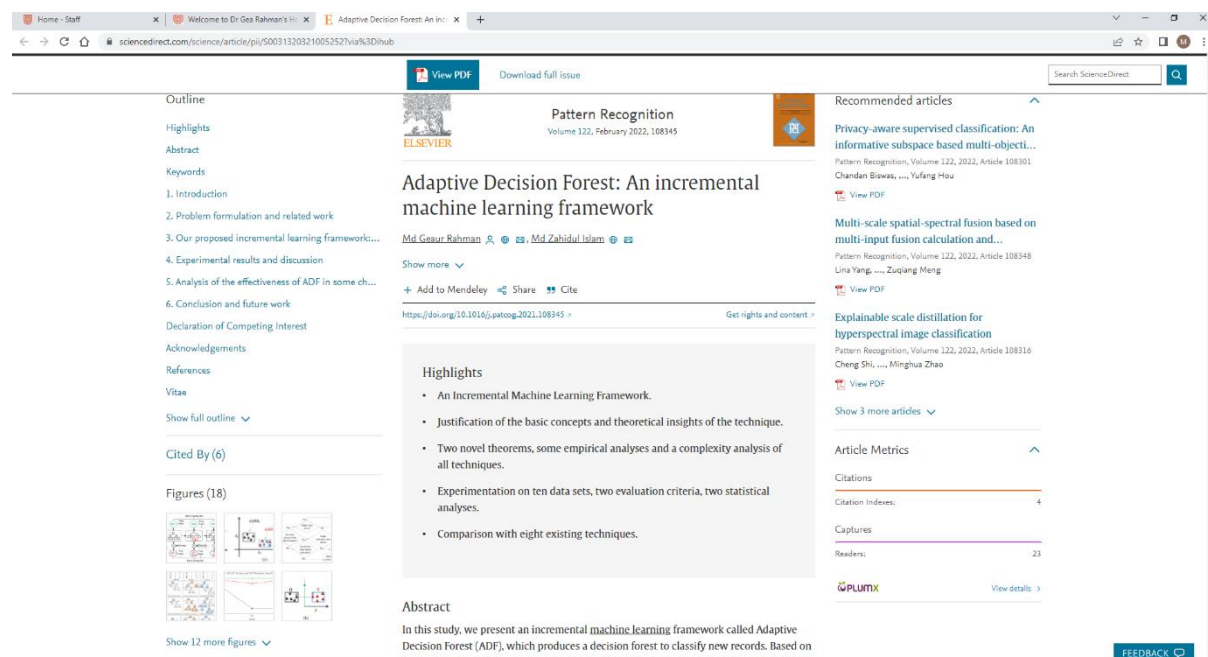


Abstract

In this study, we present an incremental machine learning framework called Adaptive Decision Forest (ADF), which produces a decision forest to classify new records. Based on our two novel theorems, we introduce a new splitting strategy called iSAT, which allows ADF to classify new records even if they are associated with previously unseen classes. ADF is capable of identifying and handling concept drift; it, however, does not forget previously gained knowledge. Moreover, ADF is capable of handling big data if the data can be divided into batches. We evaluate ADF on nine publicly available natural datasets and one synthetic dataset, and compare the performance of ADF against the performance of eight state-of-the-art techniques. We also examine the effectiveness of ADF in some challenging situations. Our experimental results, including statistical sign test and Nemenyi test analyses, indicate a clear superiority of the proposed framework over the state-of-the-art techniques.

Article Information

Rahman, M. G. and Islam, M. Z. (2022): Adaptive Decision Forest: An Incremental Machine Learning Framework. Pattern Recognition, 122, 108345. <https://doi.org/10.1016/j.patcog.2021.108345>

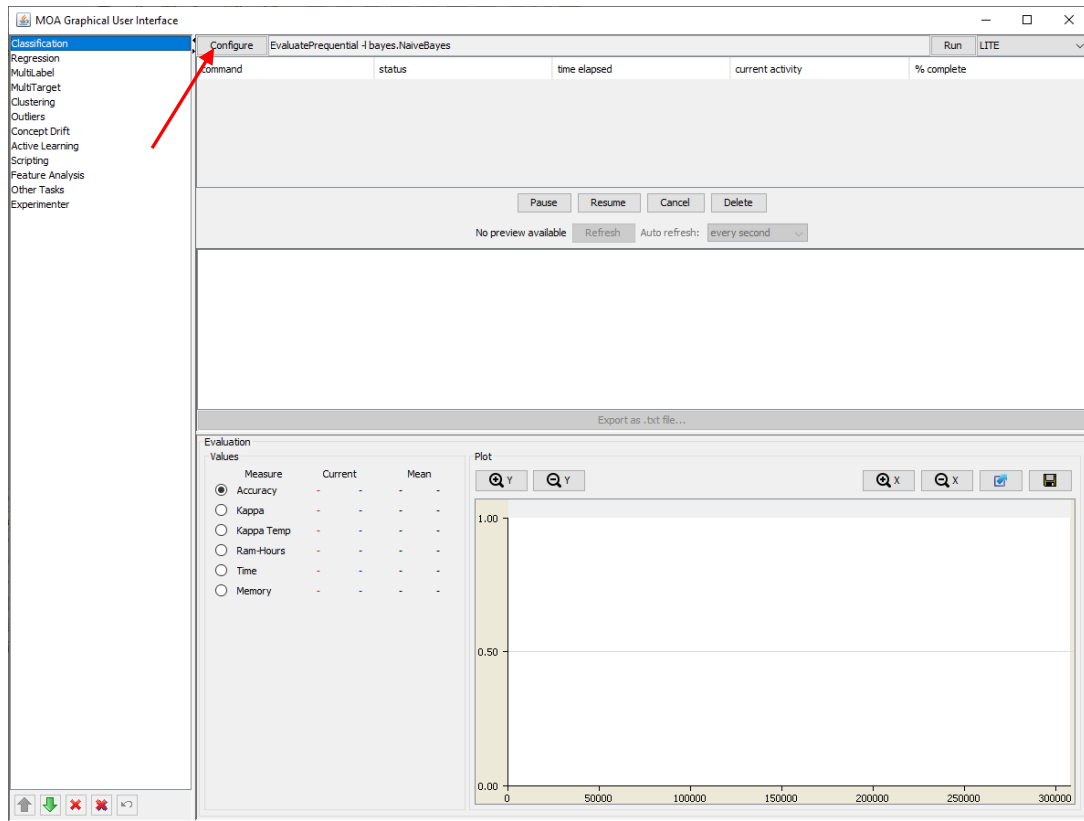


Download the source code of ADF available at <https://github.com/grahman20/ADF>

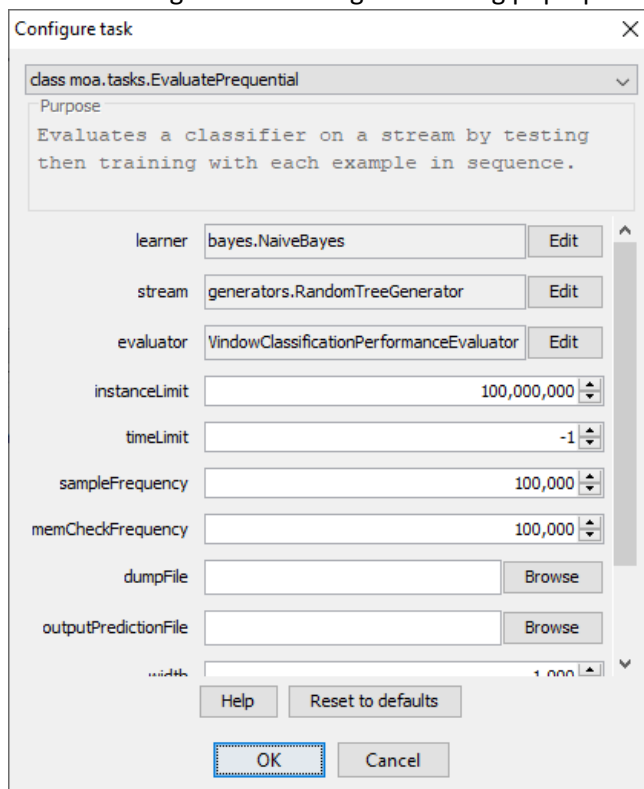
It contains two folders:

- ADF_master
 - AdaptiveDecisionForest.java
 - EvaluateModelsForBatchData.java
- SampleData
 - Contains sample train and test batch data (5 sets) and a log file (batch-log.txt) as follows

- Run the project and make sure that you got the following two screens (Note: you may need to setup the GUI.class as the main class):

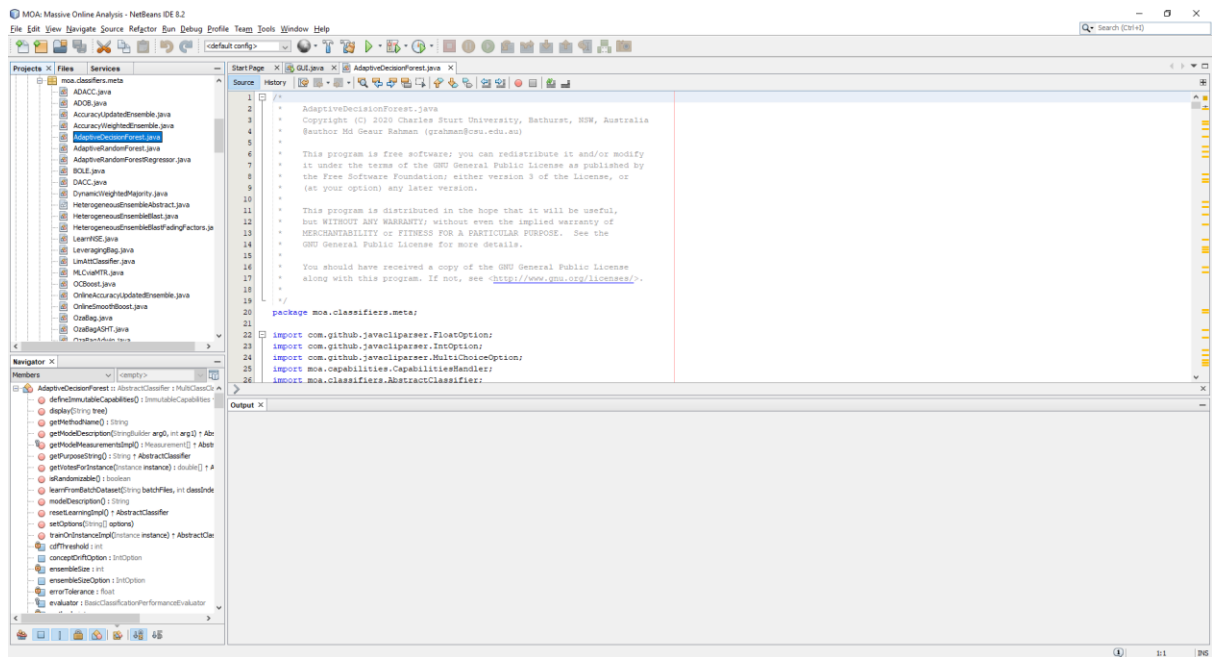


Click on Configure button to get following pop-up window:

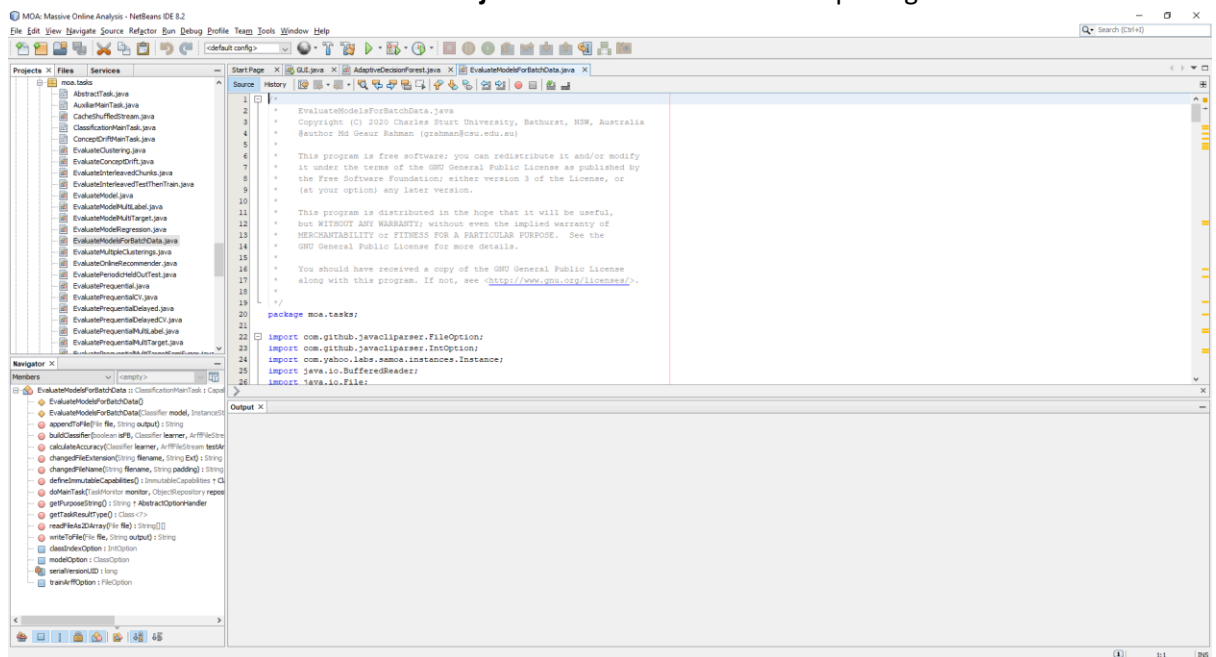


Click ok/cancel to close the pop-up window. Also close the “MOA Graphical User Interface” window.

4. Add the **AdaptiveDecisionForest.java** file under the **moa.classifiers.meta** package as follows:

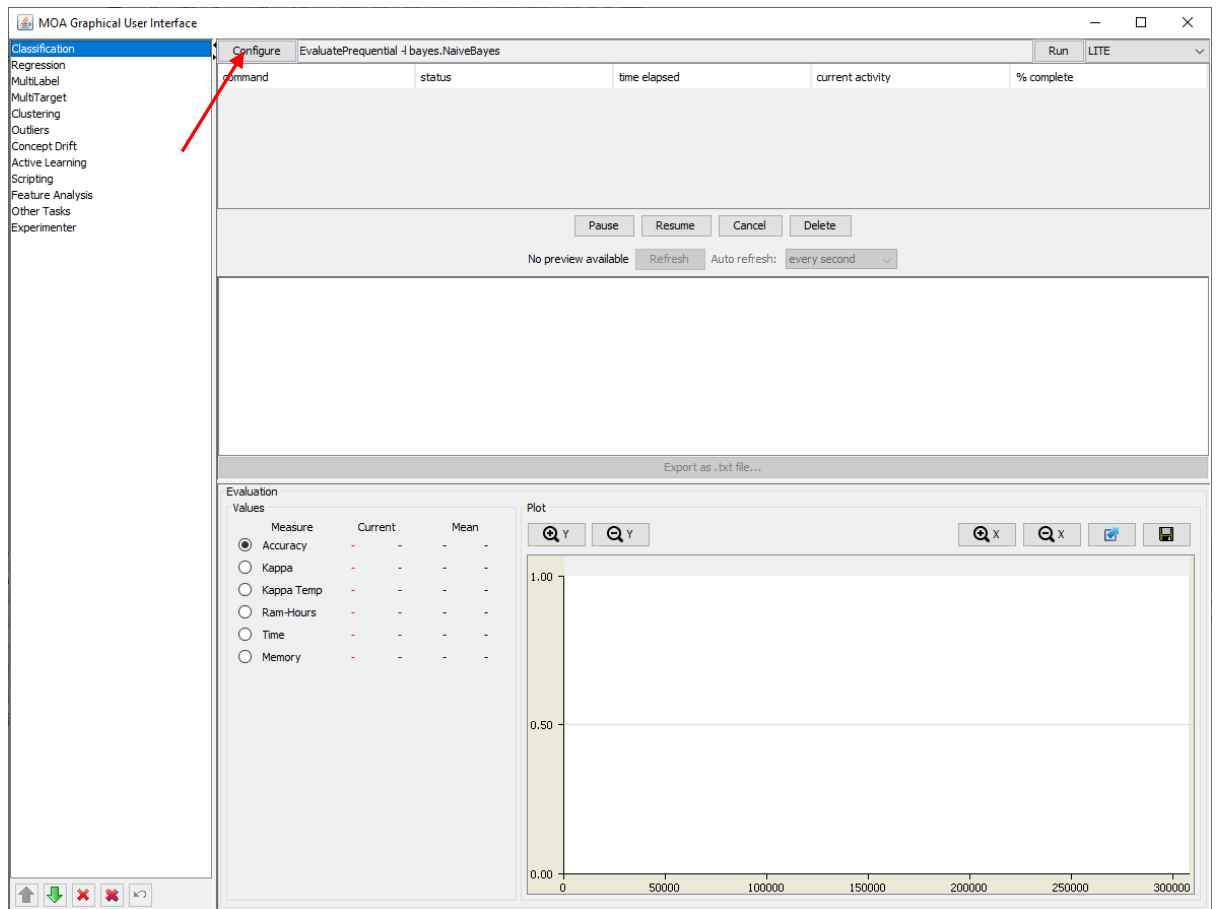


5. Add the **EvaluateModelsForBatchData.java** file under the **moa.tasks** package as follows:

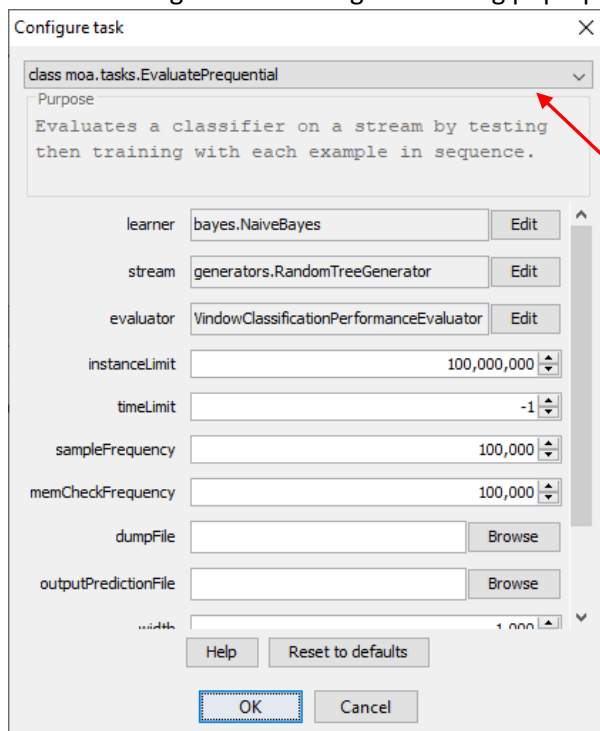


Run and Test ADF code into MOA framework

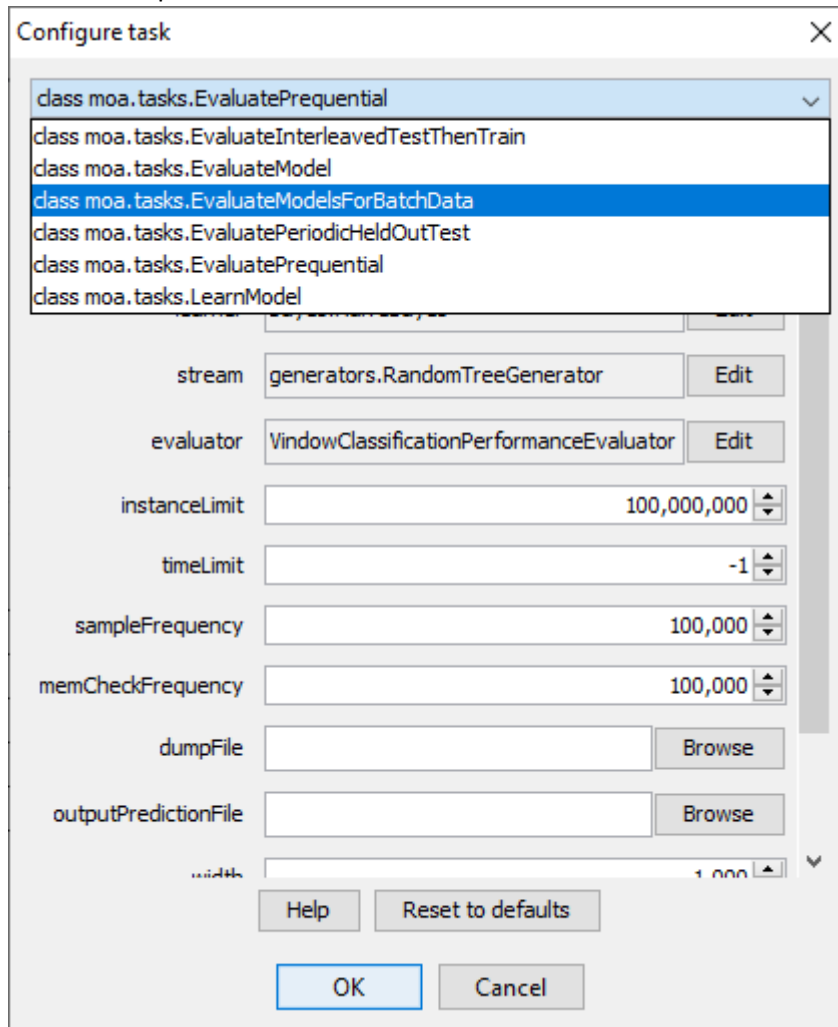
- Now Run the MOA project to open the “MOA Graphical User Interface”



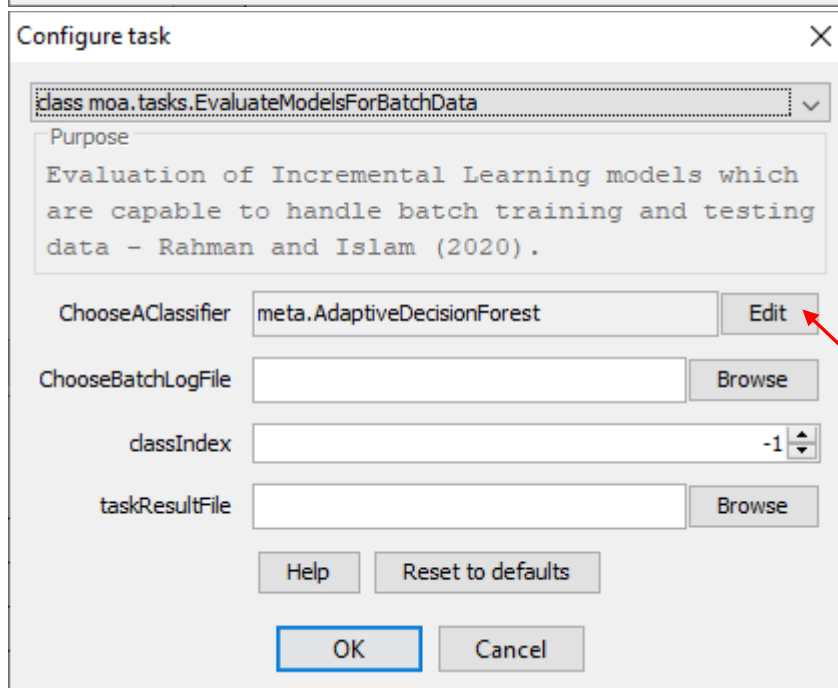
- Click on Configure button to get following pop-up window:



8. Click the drop-down menu and select the EvaluateModelsForBatchData as follows:



The 'Configure task' dialog box is shown with the 'class' dropdown menu open. The menu lists several classes, with 'class moa.tasks.EvaluateModelsForBatchData' highlighted in blue. Below the dropdown, the 'stream' is set to 'generators.RandomTreeGenerator', the 'evaluator' is 'VindowClassificationPerformanceEvaluator', and 'instanceLimit' is '100,000,000'. Other fields like 'timeLimit', 'sampleFrequency', 'memCheckFrequency', 'dumpFile', and 'outputPredictionFile' are also visible, along with 'Help', 'Reset to defaults', 'OK', and 'Cancel' buttons.



The 'Configure task' dialog box is shown with 'class moa.tasks.EvaluateModelsForBatchData' selected in the dropdown. The 'Purpose' text area contains the text: 'Evaluation of Incremental Learning models which are capable to handle batch training and testing data - Rahman and Islam (2020)'. Below this, 'ChooseAClassifier' is set to 'meta.AdaptiveDecisionForest', 'ChooseBatchLogFile' is empty, 'classIndex' is '-1', and 'taskResultFile' is empty. A red arrow points to the 'Edit' button next to 'ChooseAClassifier'. At the bottom are 'Help', 'Reset to defaults', 'OK', and 'Cancel' buttons.

9. You can now select a base learner by clicking the Edit button (shown above):

Editing option: ChooseAClassifier

class moa.classifiers.meta.AdaptiveDecisionForest

Purpose
Rahman, M. G., and Islam, M. Z. (2022):
Adaptive Decision Forest: An Incremental
Machine Learning Framework, Pattern
Recognition, pp. 100245, vol. 122, TSCN

baseLearner RF

ensembleSize 1

minimumRecords 100

repairableThreshold 0.2

pertubedETThreshold 0.01

windowSize 3

conceptDriftThreshold 3

Help Reset to defaults

OK Cancel

Editing option: ChooseAClassifier

class moa.classifiers.meta.AdaptiveDecisionForest

Purpose
Rahman, M. G., and Islam, M. Z. (2022):
Adaptive Decision Forest: An Incremental
Machine Learning Framework, Pattern
Recognition, pp. 100245, vol. 122, TSCN

baseLearner RF

ensembleSize SysFor

minimumRecords HT

repairableThreshold 0.2

pertubedETThreshold 0.01

windowSize 3

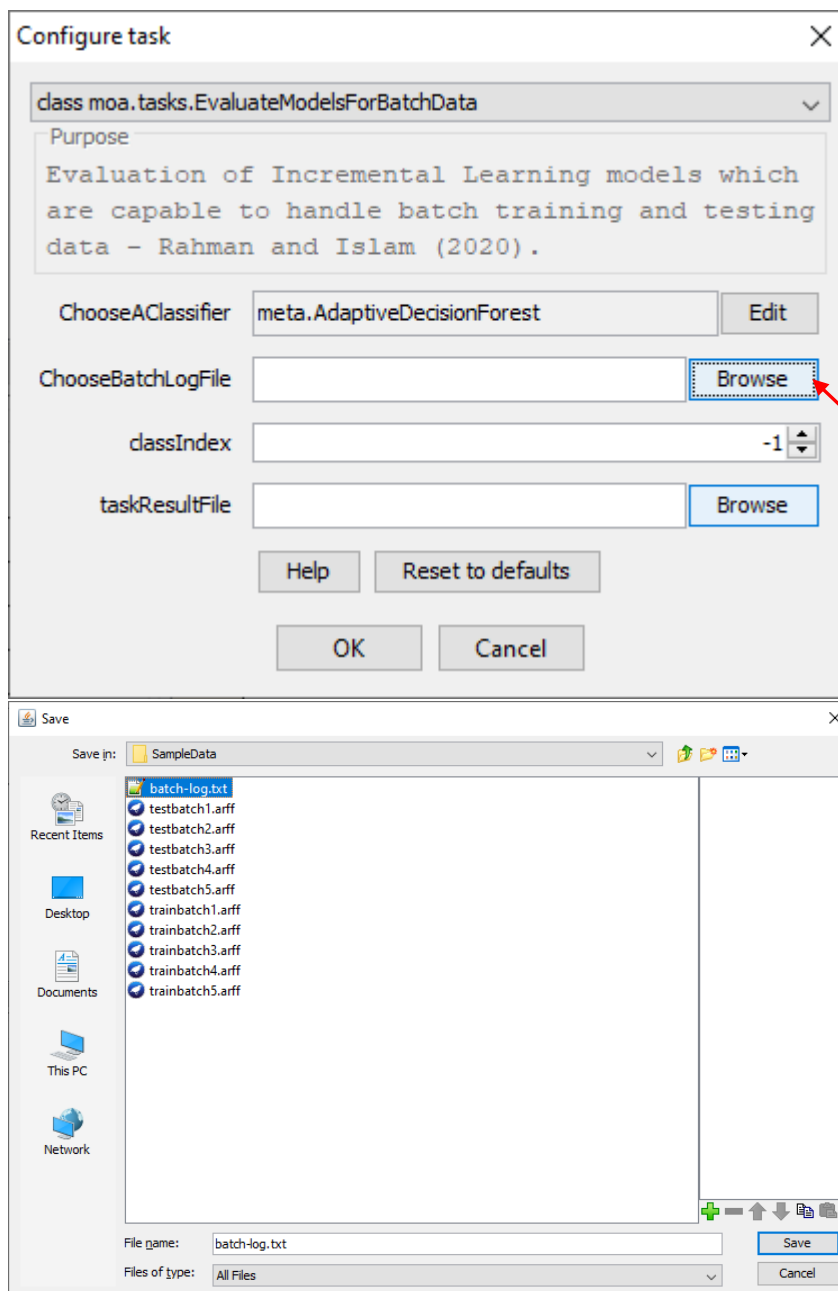
conceptDriftThreshold 3

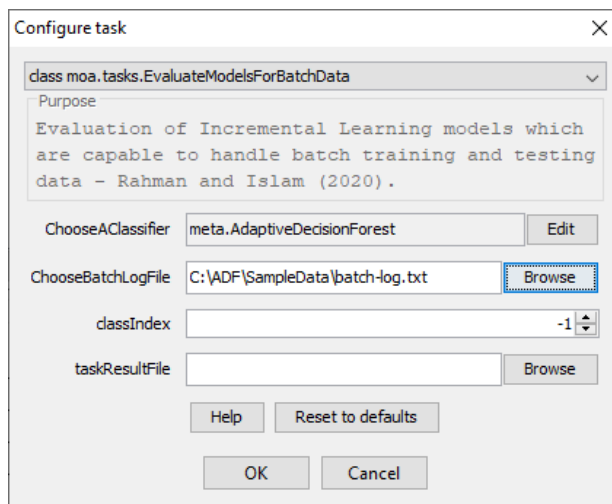
Help Reset to defaults

OK Cancel

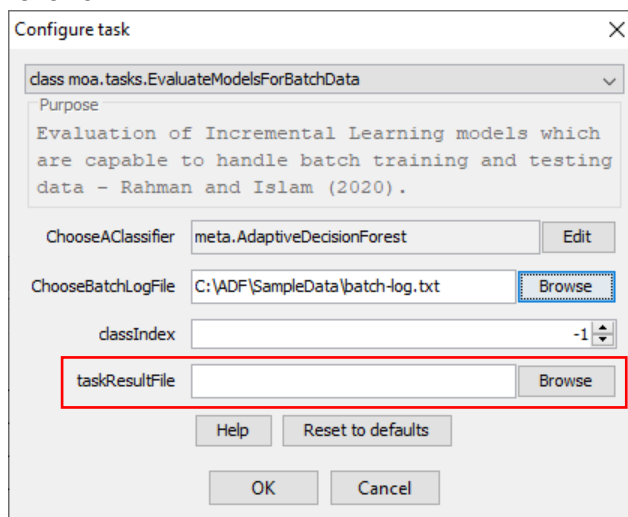
Once you are done with the base learner and parameters (which are discussed in the paper), click OK button and back to the Configure task window.

10. Choose batch log file:



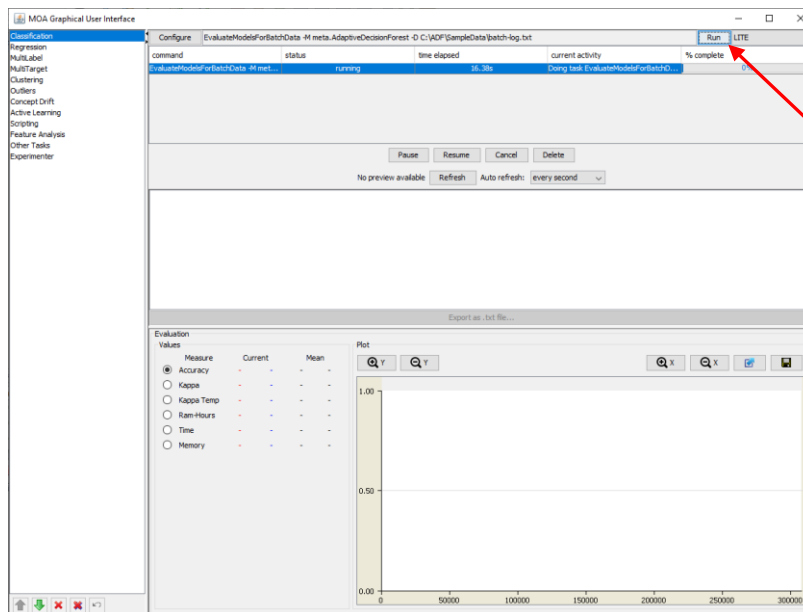


Note: you do not need to provide any value to the taskResultFile (i.e. keep it empty) as follows:

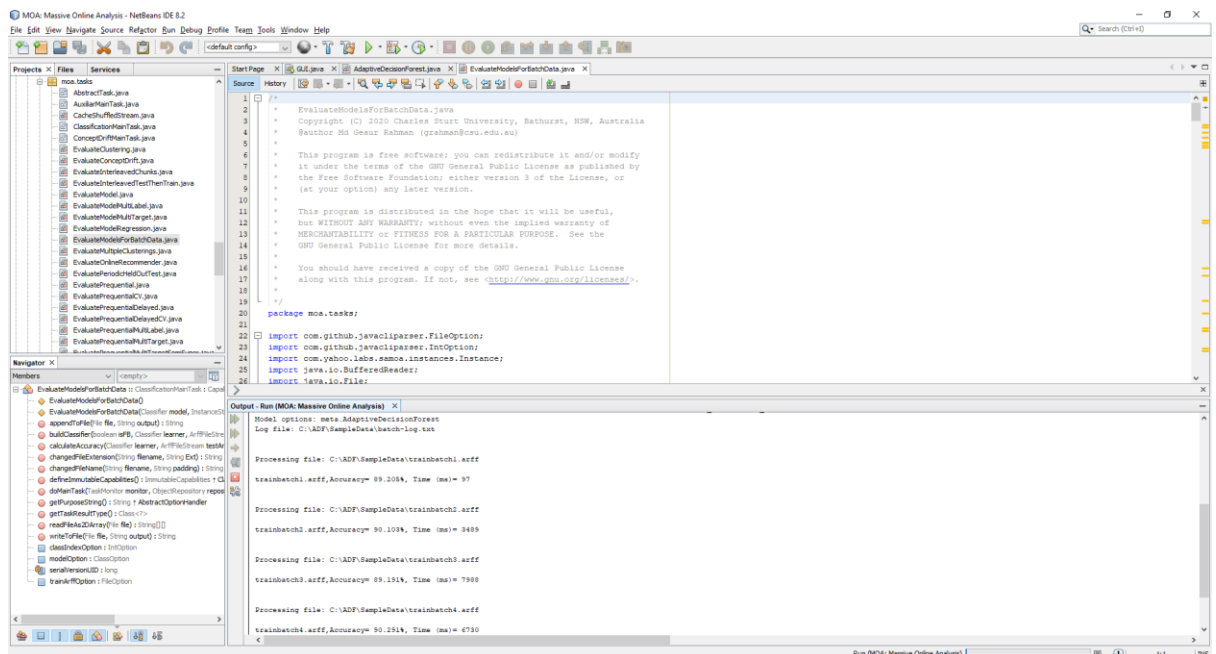


11. Click Ok to Save and close the Configure task window.

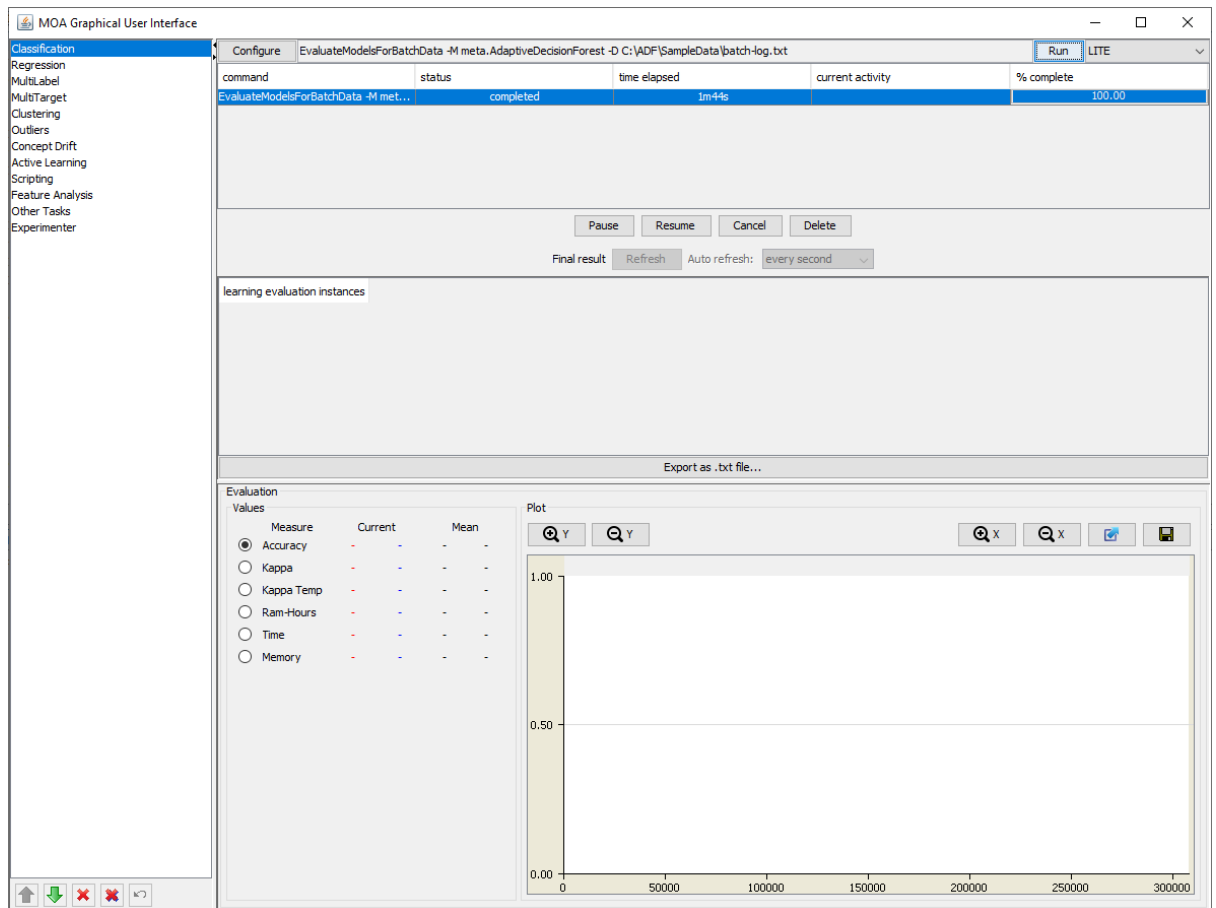
12. Now click on the Run button as shown in below:



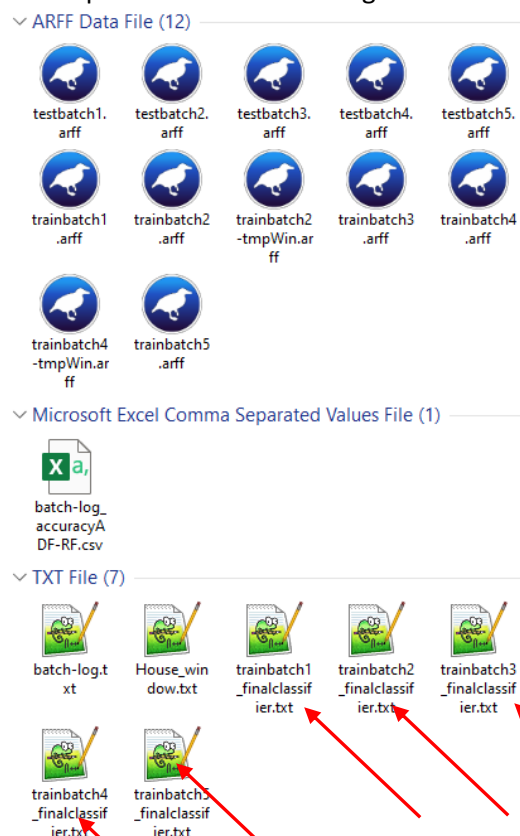
13. While the task is running you will be able to see accuracy in the console output screen as follows:



14. Once the task is done, you will see the following screens



15. Now open the folder of the log file. You will find the classifiers as follows:



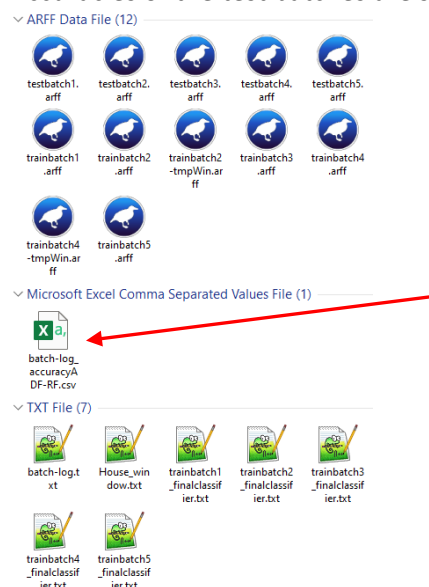
The classifier for trainbatch1 is as follows:

```

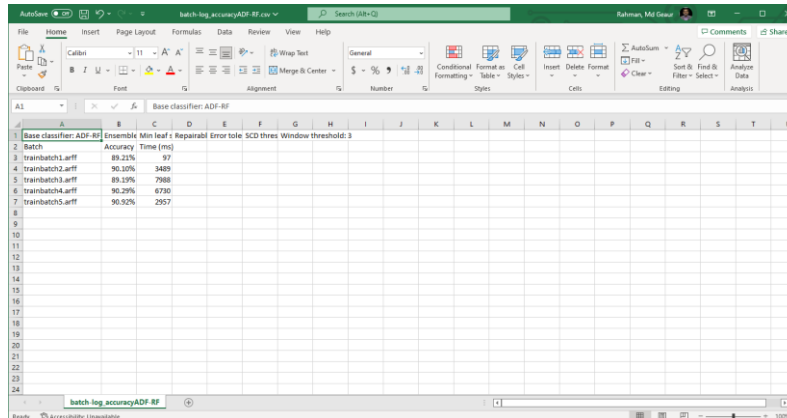
trainbatch1_finalclassifier.txt
1
2 Tree: 1, Total nodes:61, Total leaves:31, Tree depth:6
3
4 Beds < 2.5
5 | Area < 198.5
6 | | CTC = n
7 | | | View = y: UN {UN:145, LN:0, N:0, LN672:0}
8 | | | View = n: LN {UN:0, LN:92, N:0, LN672:0}
9 | | | CTC = y
10 | | | Area < 186.5
11 | | | | CTS = y: UN {UN:115, LN:0, N:0, LN672:0}
12 | | | | CTS = n: N {UN:32, LN:0, N:144, LN672:0}
13 | | | | Area >= 186.5: UN {UN:31, LN:0, N:17, LN672:0}
14 | | | Area >= 198.5
15 | | | CTS = y: UN {UN:1018, LN:0, N:0, LN672:0}
16 | | | CTS = n
17 | | | CTC = n: UN {UN:150, LN:0, N:0, LN672:0}
18 | | | CTC = y
19 | | | | Area < 269.5: N {UN:22, LN:0, N:95, LN672:0}
20 | | | | Area >= 269.5
21 | | | | | Area < 272.5: UN {UN:2, LN:0, N:1, LN672:0}
22 | | | | | Area >= 272.5
23 | | | | | | Beds < 1.5
24 | | | | | | CTT = y: N {UN:25, LN:0, N:107, LN672:0}
25 | | | | | | CTT = n: N {UN:31, LN:0, N:98, LN672:0}
26 | | | | | | Beds >= 1.5
27 | | | | | | Heating = n: N {UN:26, LN:0, N:111, LN672:0}
28 | | | | | | Heating = y: N {UN:31, LN:0, N:94, LN672:0}
29 Beds >= 2.5
30 | CTC = n
31 | | View = y
32 | | | CTS = y
33 | | | | Baths < 1.5
34 | | | | | Parkings < 0.5
35 | | | | | CTT = y: N {UN:19, LN:0, N:73, LN672:0}
36 | | | | | CTT = n: N {UN:26, LN:0, N:95, LN672:0}
37 | | | | | Parkings >= 0.5
38 | | | | | | Area < 788.5: N {UN:46, LN:0, N:142, LN672:0}
39 | | | | | | Area >= 788.5: UN {UN:4, LN:0, N:2, LN672:0}
40 | | | | | Baths >= 1.5
41 | | | | | | Area < 455.5: N {UN:8, LN:0, N:44, LN672:0}
42 | | | | | | Area >= 455.5
43 | | | | | | | Area < 736.5
44 | | | | | | | Parkings < 0.5: N {UN:33, LN:0, N:113, LN672:0}
45 | | | | | | | Parkings >= 0.5: N {UN:31, LN:0, N:125, LN672:0}
46 | | | | | | | Area >= 736.5: N {UN:8, LN:0, N:57, LN672:0}
47 | | | | | CTC = n
48 | | | | | CTT = y: N {UN:36, LN:0, N:134, LN672:0}
49 | | | | | CTT = n: N {UN:38, LN:0, N:143, LN672:0}
50 | | | | View = n

```

16. Accuracies of the test batches are stores in the CSV file as follows:



17. Open the csv file to get the accuracies:



Batch	Accuracy	Time (ms)
1	89.21%	97
2	90.10%	3489
3	89.19%	7988
4	90.29%	6730
5	90.92%	2957

Thank you for citing ADF as follows:

```
@article{RAHMAN2022108345,
title = {Adaptive Decision Forest: An incremental machine learning
framework},
journal = {Pattern Recognition},
volume = {122},
pages = {108345},
year = {2022},
issn = {0031-3203},
doi = {https://doi.org/10.1016/j.patcog.2021.108345},
url =
{https://www.sciencedirect.com/science/article/pii/S0031320321005252},
author = {Md Geaur Rahman and Md Zahidul Islam},
keywords = {Incremental learning, Decision forest algorithm, Concept drift,
Big data, Online learning},
abstract = {In this study, we present an incremental machine learning
framework called Adaptive Decision Forest (ADF), which produces a decision
forest to classify new records. Based on our two novel theorems, we
introduce a new splitting strategy called iSAT, which allows ADF to
classify new records even if they are associated with previously unseen
classes. ADF is capable of identifying and handling concept drift; it,
however, does not forget previously gained knowledge. Moreover, ADF is
capable of handling big data if the data can be divided into batches. We
evaluate ADF on nine publicly available natural datasets and one synthetic
dataset, and compare the performance of ADF against the performance of
eight state-of-the-art techniques. We also examine the effectiveness of ADF
in some challenging situations. Our experimental results, including
statistical sign test and Nemenyi test analyses, indicate a clear
superiority of the proposed framework over the state-of-the-art
techniques.}
}
```