**Data Stream Assignment Two**

Name: Jeffrey Jose  
ID: 1313512  
COMPX523-20A (HAM) Data Stream Mining Assignment Two  
16th June 2020

# **Experiment 1: MyEnsembleClassifier variations and sanity check**

**First table: Accuracy**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Electricity | RTG\_2abrupt | SEA\_abrupt. | SEA\_gradual | Covertype |
| Hoeffding Tree | 77.89% | 53.70% | 85.64% | 85.79% | 78.96% |
| MyEnsemble(S = 5) | 85.32% | 68.54% | 87.39% | 87.17% | 90.77% |
| MyEnsemble(S = 10) | 85.30% | 70.53% | 87.42% | 87.58% | 91.87% |
| MyEnsemble(S = 20) | 84.72% | 71.51% | 87.35% | 87.65% | 92.01% |
| MyEnsemble(S = 30) | 84.26% | 71.62% | 87.37% | 87.70% | 92.09% |

*Note: The length of the window was set to 1000 for all the ensemble run throughs above.*

**First table: Total Time (sec)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Electricity | RTG\_2abrupt | SEA\_abrupt. | SEA\_gradual | Covertype |
| Hoeffding Tree | 8.59 | 113.12 | 11.15 | 15.15 | 151.52 |
| MyEnsemble(S = 5) | 73.04 | 651.21 | 105.70 | 142.84 | 559.84 |
| MyEnsemble(S = 10) | 131.22 | 1355.74 | 196.82 | 274.89 | 1022.70 |
| MyEnsemble(S = 20) | 244.54 | 2342.27 | 348.77 | 478.27 | 1999.96 |
| MyEnsemble(S = 30) | 369.07 | 3408.07 | 535.81 | 753.11 | 2934.66 |

**Second table: Accuracy**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Electricity | RTG\_2abrupt | SEA\_abrupt. | SEA\_gradual | Covertype |
| MyEnsemble(seed = 1) | 85.58% | 74.23% | 87.25% | 87.12% | 88.89% |
| MyEnsemble(seed = 2) | 85.18% | 74.02% | 87.30% | 87.07% | 88.67% |
| MyEnsemble(seed = 3) | 85.29% | 75.51% | 87.35% | 86.89% | 88.37% |
| MyEnsemble(seed = 4) | 85.28% | 74.84% | 87.23% | 86.92% | 88.60% |
| MyEnsemble(seed = 5) | 85.40% | 74.99% | 87.25% | 87.05% | 88.41% |

*Note: The number of learners: 20 and the length of window: 1000 for all ensembles.*

**Second table: Total Time (sec)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Electricity | RTG\_2abrupt | SEA\_abrupt. | SEA\_gradual | Covertype |
| MyEnsemble(seed = 1) | 154.35 | 1465.07 | 374.72 | 363.40 | 1397.96 |
| MyEnsemble(seed = 2) | 148.79 | 1535.42 | 398.99 | 346.28 | 1386.20 |
| MyEnsemble(seed = 3) | 153.08 | 1427.61 | 358.64 | 343.05 | 1418.13 |
| MyEnsemble(seed = 4) | 151.70 | 1442.32 | 346.35 | 335.15 | 1400.74 |
| MyEnsemble(seed = 5) | 148.07 | 1445.54 | 349.57 | 344.42 | 1392.28 |

**Third table: Accuracy**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Electricity | RTG\_2abrupt | SEA\_abrupt. | SEA\_gradual | Covertype |
| MyEnsemble(l = 500) | 85.53% | 75.10% | 87.40% | 87.69% | 88.69% |
| MyEnsemble(l = 1000) | 84.89% | 74.92% | 87.50% | 87.81% | 89.13% |
| MyEnsemble(l = 2000) | 84.05% | 74.81% | 87.36% | 87.74% | 88.80% |
| MyEnsemble(l = 5000) | 83.59% | 76.43% | 87.44% | 87.80% | 88.51% |
| MyEnsemble(l = 10000) | 83.26% | 75.62% | 87.38% | 87.63% | 88.49% |

*The number of learners for all ensembles: 20*

**Third table: Total Time (sec)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Electricity | RTG\_2abrupt | SEA\_abrupt. | SEA\_gradual | Covertype |
| MyEnsemble(l = 500) | 239.53 | 1419.26 | 336.72 | 472.47 | 1396.53 |
| MyEnsemble(l = 1000) | 236.77 | 1388.18 | 346.29 | 478.46 | 1400.52 |
| MyEnsemble(l = 2000) | 238.68 | 1478.87 | 332.28 | 462.59 | 1379.07 |
| MyEnsemble(l = 5000) | 251.53 | 1489.38 | 330.52 | 357.44 | 1382.81 |
| MyEnsemble(l = 10000) | 247.62 | 1556.00 | 337.62 | 468.32 | 1642.07 |

1. ***How does TheEnsemble compares against a single hoeffding tree? Also, does TheEnsemble improve results if more base models are available? Present a table with results varying from S = {5, 10, 20, 30}, where S is the total number of learners.***

An ensemble learning is used to help improve machine learning results by combining several models which allows the production of better predictive performance compared to that of a single model. The basic advantage of an ensemble is the improvement in predictive accuracy and the primary disadvantage is that it is difficult to understand an ensemble of classifiers. Therefore, the basic idea is to learn a set of classifiers and allow them to vote. The main challenge in developing ensemble models is to obtain base models which make different kinds of errors but not to obtain highly accurate base models.

There are three main reasons to use ensembles and the challenges they overcome:

* **Statistical Problem –** The generalization capabilities of each classifier might be different when they are applied to a test set which in turn means that their accuracy will vary. Therefore, it is a much safer option to use the mean of individual predictions from these classifiers instead of using only one of them, since the chance of selecting the classifier with the worst generalization capabilities is eliminated.
* **Computational Problem –** The computational problem arises when the learning algorithm cannot give assurance of finding the best hypothesis.
* **Representational Problem –** The representational problem arises when the hypothesis space does not contain any good approximation of the target classes.

Within our project, we have used Hoeffding Tree as the set of classifiers with randomized values for the split confidence, grace period and nb threshold. The number of learners is defined by the user but within my implementation, I have assigned 3 learners as the default number of learners. Therefore following, the information provided above, my hypothesis is that my ensemble which features a higher number of base models would provide a much more accurate result but in terms of computational time, it may take longer to compute. From my results outlined in table one, I can confirm that as the number of learners increase, the prediction accuracy of the ensemble also increases. We can see that in the rtg\_abrupt dataset which contains a drift at 30,000 and 60,000, that the accuracy when the base learner is set to 5 is 68.54% which gradually increases as the number of base learners increase to the point that when the base learner is set at the maximum of 30 for this experiment, the accuracy vastly improved to 71.62%. This pattern is reciprocated in the sea\_gradual dataset where the accuracy is 87.17% when the number of base learners is set to 5 as opposed to 87.70% as the number of base learners is set to 30. Although the accuracy is greatly improved, increasing the number of base learners resulted in a very large computational time as shown by the second table. Going back to the example using the dataset, rtg\_abrupt, we can see that the computational time is only 651.21 seconds with 5 base learners as opposed to 3408.07 seconds when there are 30 base learners defined. Through this experiment, I have discovered that setting a higher number of samples to be classified helps to highlight the effect of increasing the number of learners more clearly as some datasets may contain drifts which can take some period for the ensemble to cover from. We can see that for example, the sea\_abrupt dataset does not show a very clear pattern as to highlight the effect of varying the levels of the number of learners. But I believe that I set the max number of samples to be assessed to 75,000 and the drift within this data occurs approximately around 60,000 hence the ensemble may not have had enough time to recover. The other two datasets which consisted of a drift but showed a very clear pattern had a max number of samples to be assessed set to 100,000 hence there was enough time for the ensemble to recover after experiencing the drift. It will be interesting to look at the effect if I had set the max samples to 100,000 for the sea\_abrupt dataset. When comparing the accuracy of the ensemble to that of single Hoeffding Tree, I can confidently say that the ensemble outperformed the Hoeffding Tree on every dataset. This is because the ensemble uses weighted prediction accuracies from multiple learners in order to finalize the best model for the dataset and produce an overall highly accurate result whereas a single Hoeffding Tree classifies instances without an optimum for comparison and selection of best model. For example, coming back to sea\_gradual dataset, we can see that the accuracy of a single Hoeffding Tree is 85.79% which is lower than even the lowest number of base learners which resulted in an accuracy of 87.17%. However, a positive aspect of the single Hoeffding Tree is that it is very fast compared to the ensemble which I have implemented. We can see that across the five datasets, the Hoeffding Tree recorded the lowest times, and this is because the Hoeffding Tree is a very fast decision tree algorithm primarily for streaming data in which it waits for new instances to arrive rather than reusing instances. On average, the Hoeffding Tree took less than 30 seconds to compute the full number of samples for each given dataset whereas the ensembles went into hours in some cases. This is due to the ensembles creating numerous base learners which runs the computation in parallel and the result must be compared to find the optimal and most accurate result which is a very time-consuming process.

1. ***Is the ensemble able to recover from concept drifts? Present a plot depicting the accuracy over time to support your claim. There should be a plot for each dataset comparing whichever version of TheEnsemble that produced the best results on the first table against the single hoeffding tree.***

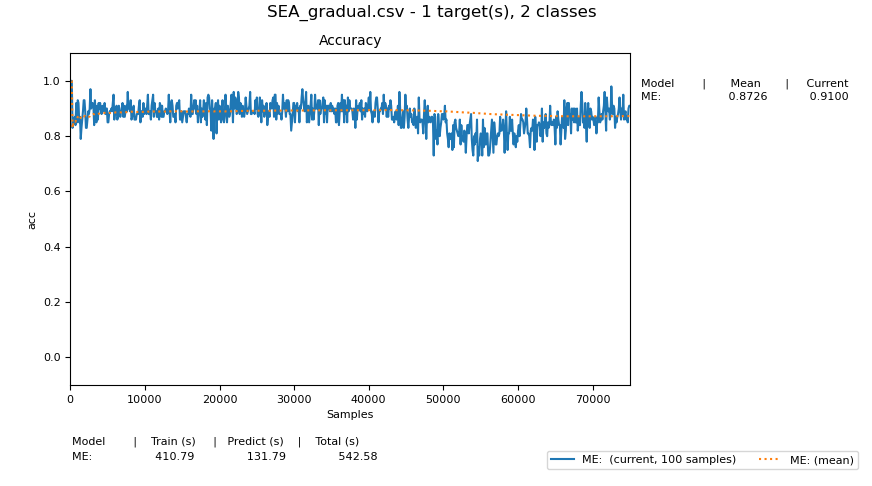
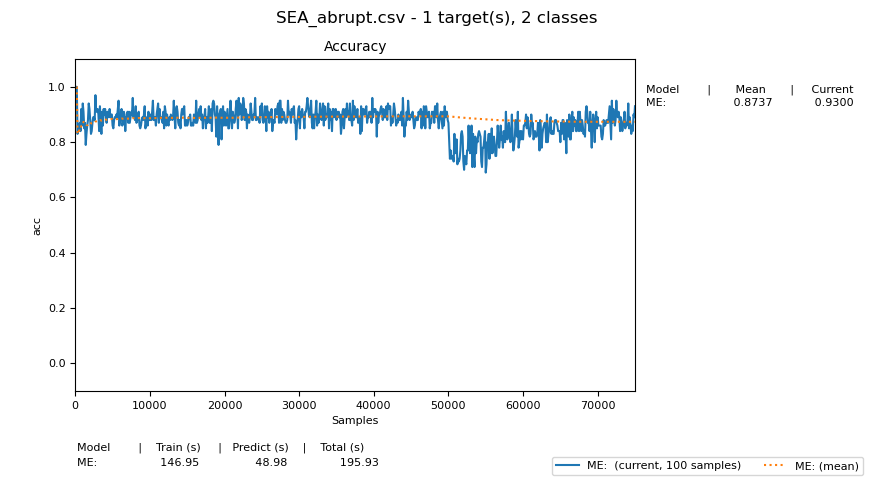
A concept drift in data mining refers to the progression/change in the underlying distribution of the data. Therefore, as a result of this, predictions by models may become less accurate as time passes by and the chance to improve the accuracy of the model may be missed (depending on whether it is able to recover). Due to this, it is very important that models adapt to changes quickly and accurately. A change to the data can occur in shape or form. It is often easier to consider changes which hold a temporal consistency such as the where the data collected within a specific period show the same relationship and this relationship changes smoothly over time. Below is some possible ways data could change resulting in poor predictive performance:

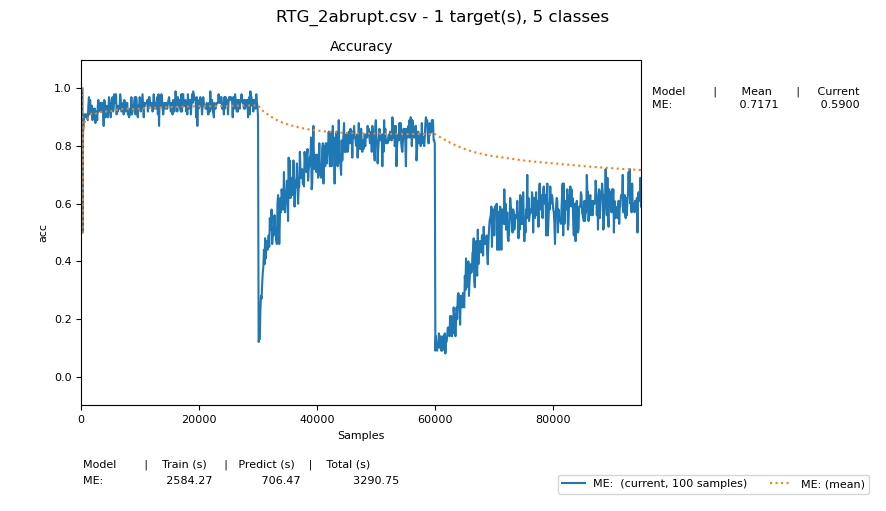
* A gradual change over time.
* A recurring or cyclical change.
* A sudden or abrupt change.

There are many ways to deal with concept drifts as we have tried to do within this project. Below are some that were talked about in *Handling Concept Drift: Importance, Challenges & Solutions*:

1. Do Nothing (Static Model)
2. Periodically Re-fit the static model.
3. Periodically update the learning model.
4. Weight the data as in weigh the importance of the input data.
5. Learn the change such as possibly boosting type ensembles.
6. Detect and choose a specific model for predictions.
7. Prepare the data in a way to remove the systematic changes to the data over time.

Now we will look into how my ensemble handles concept drift within the multiple datasets. I will be displaying the drift recovery for the datasets: sea\_gradual, sea\_abrupt and rtg\_abrupt as the other two datasets [electricity and covtype] did not contain any drifts.





As highlighted in the series plots above, we can see that around the 50,000th instance on both the sea datasets, that the prediction accuracy dramatically drops indicating a drift in which most class predictions by the ensemble is incorrect. However, it learns from this and we can see that it slowly starts to rise back up hence indicating the ability of the ensemble to recover from a concept drift. We can also see that in the rtg\_abrupt dataset, there are two sudden drops in accuracy occurring around 30,000 and 60,000 instances. This drop occurred so suddenly, and the ensemble took more time to recover as opposed to the sea datasets primarily due to the fact that the drop in prediction accuracy was so sudden and steep. Therefore, we can confidently conclude that the current ensemble is able to recover from concept drift if one ever occurs.

1. ***What is the impact of the l hyperparameter? Discuss the results varying the l hyperparameter.***

Although there is no clear structured pattern to emphasis the effect of varying the window length, I did notice that the accuracy in general decreases as the window length increases. The updates to the ensemble will occur every ‘l’ instances and therefor at the start of each window, a new base model is added as a candidate and the end of the window, the ensemble will either replace another learner currently in the ensemble by *c* or ignore *c* and discard it. A possible explanation as to why the prediction accuracy decreases slightly when the window length is large is due to the increased time to replace the weakest learner in the ensemble. The weakest learner contains the prediction accuracy which are not fully as representative of the ensemble’s prediction capabilities as opposed to the other learners. In the table previously, we can see for example on the dataset sea\_gradual that the accuracy given the window length is 500 is 87.69% but as we increase the size of the window length, we can see that the accuracy gets better and worse when finally at a window length of 10,000, we see that the prediction accuracy drops down to 87.63%. This is the almost identical trend across all the datasets in which there is no clear pattern but the highest value of window length has a lower prediction accuracy than that of the lowest window length, although in some cases, this difference may not be large but it is still a difference nonetheless.

# **Experiment 2: MyEnsembleClassifier vs. others**

**Taking Accuracy Into Consideration:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Electricity | RTG\_2abrupt | SEA\_abrupt. | SEA\_gradual | Covertype |
| MyEnsemble(S = 20, l = 500) | 85.99% | 75.22% | 87.42% | 87.15% | 88.46% |
| Leverage Bagging Classifier | 81.48% | 57.51% | 78.18% | 77.73% | 89.77% |
| Dynamic Weighted Classifier | 79.30% | 69.25% | 88.03% | 87.54% | 71.19% |
| Adaptive Random Forest | 82.31% | 66.14% | 87.61% | 87.28% | 77.66% |

**Taking Time Into Consideration (Secs):**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Electricity | RTG\_2abrupt | SEA\_abrupt. | SEA\_gradual | Covertype |
| MyEnsemble(S = 20, l = 500) | 262.59 | 1454.83 | 354.91 | 384.68 | 1260.54 |
| Leverage Bagging Classifier | 1941.64 | 10790.31 | 2261.16 | 2250.04 | 7405.76 |
| Dynamic Weighted Classifier | 113.55 | 1297.22 | 190.83 | 189.29 | 889.32 |
| Adaptive Random Forest | 284.24 | 736.20 | 452.68 | 444.86 | 570.17 |

1. ***How does TheEnsemble performs in terms of predictive performance against the others?***

I found that a relatively high number of learners and low number of window size is the best combination in order to achieve maximum predictive accuracy and hence I have used it for my experiment 2.

Leverage Bagging:

* Leverage bagging improves online bagging accuracy by adding more randomization to the input by using a larger lambda value for the Poisson distribution. With leverage bagging, the more instances used for each learner results in a more refined hypothesis and the random error correcting codes in turn creates more diversity at the output. Leverage bagging uses the same strategy as ADWIN Bagging for concept drifts.

Dynamic Weighted Majority:

* Weighted Majority weighs learner votes based on previous performance. In this method, every learner has a weight which is decreased every time it predicts the class incorrectly. Dynamic Weighted Majority uses weighted majority with the addition of updating heuristics to cope with evolving streams and the update method includes removing classifiers if their weight is below a specific user specified threshold. The ensemble size varies due to removals and additions of classifiers and new classifiers are added every time the ensemble prediction is incorrect. Dynamic Weighted Majority does not include an explicit drift detection method and it adapts to drifts by removing old classifiers.

Adaptive Random Forest (ARF):

* Adaptive Random Forest is the streaming version of the original random forest which uses a variation of the Hoeffding tree although the key differences are relating to the bootstrap aggregation and the base learner. Adaptive Random Forest is a parallel process and there are no dependencies between the trees. The most important aspects of the Adaptive Random Forest algorithm are that:
  + It adds diversity through resampling.
  + It adds diversity through randomly selecting subsets of features for node splits.

In ARF, there are one warning and one drift detector per base model and it uses the ADWIN algorithm as the primary detector. Background learners are started once a warning is detected, their subspace of features may not correspond to the subspace of features used by the current learner. Once a drift is detected, the background learner replaces the current learner.

In terms of predictive performance, the ensemble which I implemented varies between the different datasets so I will be going through each. In the electricity dataset, my ensemble is the performs the best with an overall predictive accuracy of 85.99% as opposed to that of the dynamic weighted classifier which had the poorest predictive performance with an accuracy percentage of 79.30%. As for both the sea\_abrupt and sea\_gradual, I see that dynamic weighted classifier had the highest accuracy with a value of 88.03% and 87.54% respectively. This performance was followed by that of the adaptive random forest with a predictive accuracy of 87.61% and 87.28%. The third best performer was my classifier with an accuracy of 87.42% and 87.15%. Although the performance was slightly worse than that of the adaptive random forest, the adaptive random forest did take a longer time to execute and so there is a trade-off between time and predictive accuracy in play. The leverage bagging classifier had a large computational time as well as poor predictive accuracy with a value of 78.18% and 77.73% respectively which entails that this may not be the best classifier for the two given dataset which features drifts around 60000. Overall, I believe that the ensemble did perform reasonably well compared to the other methods. Although it did not always produce the highest accuracy, its ability to be executed in a short space of time allows it to be a reasonable option to use when classifying large datasets and ones which may contain a drift. (Refer to table under experiment two for full comparison of accuracy and the second table for comparison of computational time).

1. ***In terms of time, is TheEnsemble more efficient than the other methods?***

When comparing the computational times between the different classifiers, I found the Leverage Bagging Classifier to be taking the longest amount of time. As shown by the table, on most occasions, the execution time of the Leverage Bagging Classifier is close to 2000 seconds and this large time frame is explained by leverage bagging’s property to handle multiclass problems using only a binary classifier and using a sampling without replacement method. As for the ensemble that I implemented I see that the computational time on most part is around 300 seconds, however this is much slower than that of dynamic weighted classifier which is only around 200 seconds on maximum. This is because algorithms for coping with concept drifts must converge quickly and accurately to new target concepts, whilst being efficient in time and space and dynamic weighted classifier is especially implemented for dealing with drifts in particular. I have also made the observation that the dataset which included drifts within them such as sea\_abrupt experienced a larger computational time as opposed to datasets which do not contain drifts such as electricity. Although it may not be as easy to notice within the testing phase, I did notice that datasets which included a high number of predictor attributes contributed to a larger computational time and I assume that this is due to the high number of information processing that is done by the classifier which in turn takes time. Although the adaptive random forest isn’t computationally intensive, I did notice that it completes its phase at a much slower rate than that of dynamic weighted classifier and the ensemble which I implemented. Overall in terms of time, the ensemble which I implemented is more efficient than the adaptive random forest and the leverage bagging classifier but not as efficient as the dynamic weighted classifier. The pattern of computational time efficiency is maintained within the execution of each individual dataset.

# **Reference**

Bifet, A. (n.d.). *Machine learning for data streams*. Computing and Mathematical Sciences: University of Waikato. Retrieved June 22, 2020, from <https://www.cms.waikato.ac.nz/~abifet/book>

Gomes, H. M., Montiel, J., & Bifet, A. (n.d.). *Ensembles 2* [PowerPoint slides]. <https://elearn.waikato.ac.nz/pluginfile.php/1990624/mod_resource/content/1/COMPX523-20-ensembles2.pdf>

Gomes, H. M., Montiel, J., & Bifet, A. (n.d.). *Ensembles 1* [PowerPoint slides]. <https://elearn.waikato.ac.nz/pluginfile.php/1986352/mod_resource/content/1/COMP523X-20-ensembles1.pdf>

Kolter, J. Z., & Maloof, M. A. (n.d.). *Dynamic Weighted Majority: A New Ensemble Method for Tracking Concept Drift∗*. Georgetown University, Washington, DC. <https://ai.stanford.edu/~kolter/lib/exe/fetch.php?media=pubs:kolter-icdm03.pdf>