# Data Stream Classification

Muhammad Abdullah S. Mulkana – Bilkent University, Ankara

21801075

## 1. Introduction

In this project, our objective is to explore the concept of classification of data streams. We generate data streams with varying noise and drift features. Then we implement K Nearest Neighbors (KNN), Hoeffding Tree (HT), and Naïve Bayes (NB) with varying batch sizes to analyze online learners, and we implement Majority Voting (MV) and Weighted Majority Voting (WMV) in order to observe the performance of ensemble methods. Finally, we also take a look at another way to increase the performance of our models.

## 2. Methods

In order to generate data streams, we use the HyperplaneGenerator function from the scikit-multiflow library [1]. This allows us to create datasets constraining data streams of varying features, samples, number of drift features, and noise levels [1]. For our purpose, we create four different datasets with the (number of drift features, noise level) parameters set as (2,10), (2,30), (5,10), and (5,30). Each dataset has 20000 samples and is stored as separate .csv files. Then we use different classifiers in order to observe how they perform on this data. The classifiers we use are K Nearest Neighbors (KNN) [2], Hoeffding Tree (HT) [3], and Naïve Bayes (NB) [4]. Other than these supervised classifiers, we also have two ensemble classifiers: Majority Voting (MV) and the Weighted Majority Voting (WMV) rules. These ensemble classifiers use the combination of the three supervised classifiers mentioned above in order to increase performance. For the MV classifier, we count the frequency of labels by the three classifiers. The label with the most frequency is the predicted label. For WMV, we calculate the prediction probabilities [5]. Then we use a set of weights, determined by trial and error, and multiply the prediction probabilities by their weights [5]. Then if the weighted sum for label 0 is higher than that for label 1, our predicted label is 0 and vice versa [5].

To train these models, we use two different approaches. The first is online learning, where the data is read one sample at a time in order to simulate a data stream. For online learning, we use the interleaved-test-then-train method, where we first test the sample on the classifier and then partially fit the classifier one by one as we read the samples [6]. The second is batch learning, where a certain amount of data samples are first collected, then the classifiers are fitted.

## 3. Results

In this section, we present the results of our classifiers. Table 1 shows the classification accuracies and the runtime of the three online classifiers. For each classifier, we implement the interleaved-test-then-train method using batch sizes of 1, 100, and 100. The runtime is the time taken to run all four datasets.

*Table 1: Accuracies of online classifiers.*

| **Dataset** | **KNN** | | | **HT** | | | **NB** | | |
|---|---|---|---|---|---|---|---|---|---|
| | **1** | **100** | **1000** | **1** | **100** | **1000** | **1** | **100** | **1000** |
| **Hyperplane Dataset 10_2** | 84.6 | 84.4 | 83.4 | 84.2 | 84.1 | 82.4 | 88.4 | 88.3 | 86.7 |
| **Hyperplane Dataset 30_2** | 66.3 | 66.4 | 66.2 | 65.0 | 65.1 | 65.1 | 65.2 | 65.2 | 65.2 |
| **Hyperplane Dataset 10_5** | 85.2 | 85.3 | 84.7 | 81.1 | 81.1 | 80.8 | 80.9 | 80.9 | 80.8 |
| **Hyperplane Dataset 30_5** | 65.9 | 66.1 | 66.0 | 63.7 | 63.7 | 63.6 | 64.4 | 64.4 | 64.4 |
| **Runtime (s)** | 198.6 | 23.7 | 21.5 | 26.2 | 18.7 | 17.3 | 12.5 | 8.1 | 7.9 |

While training the models using the interleaved-test-then-train method, we also record the accuracies as we progress. Figure 1, Figure 2, and Figure 3 show these intermediate accuracies and how they converge for each online classifier.

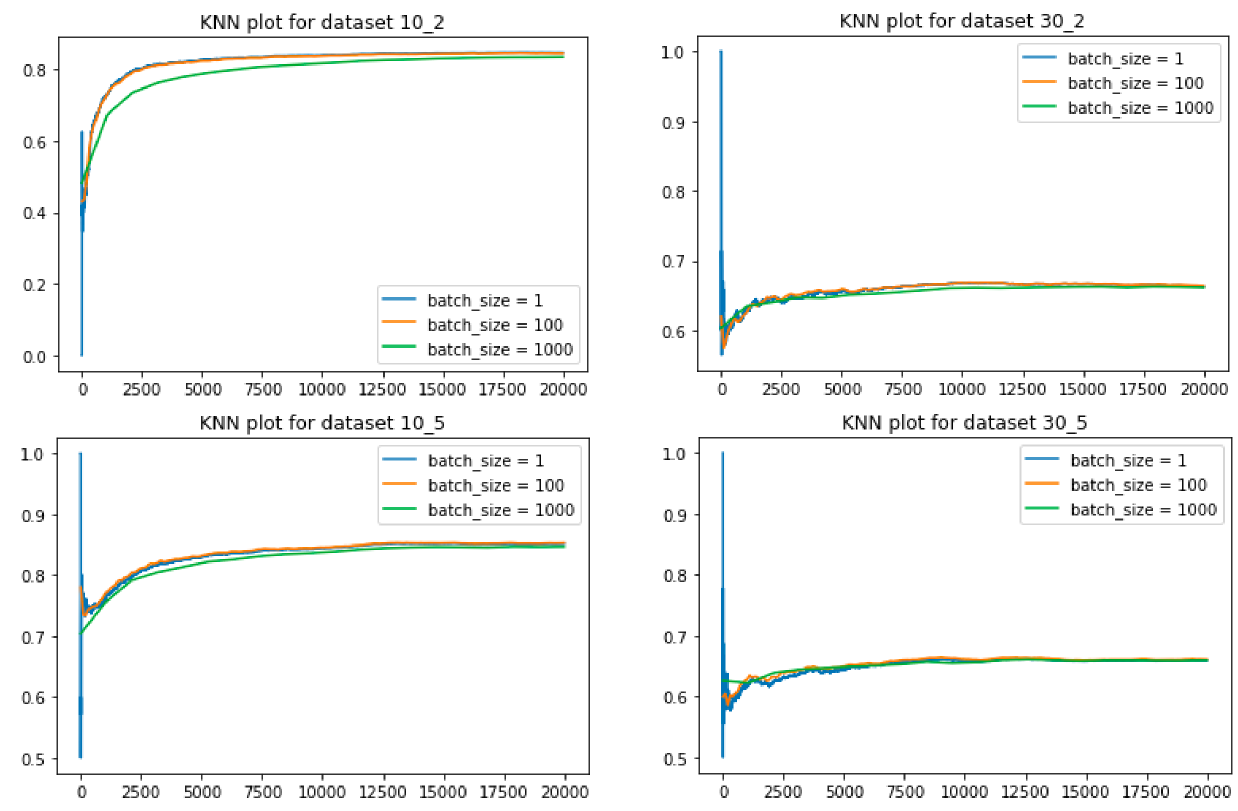

*Figure 1: Accuracies of KNN.*

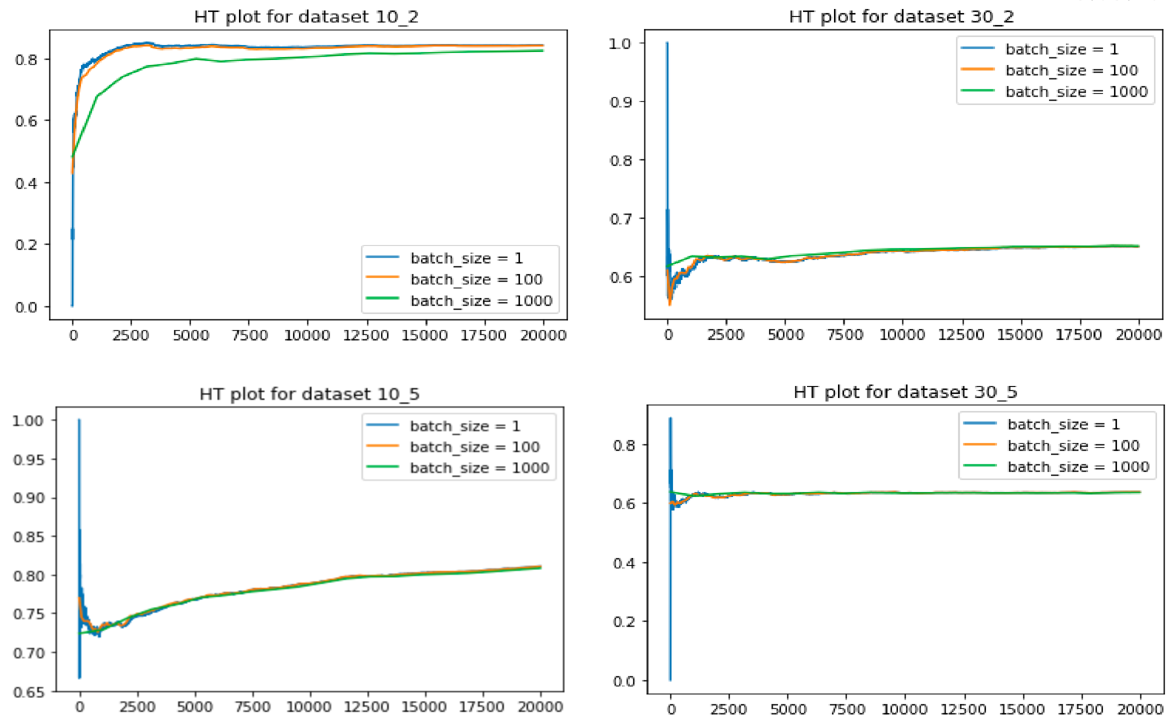

*Figure 3: Accuracies of HT.*

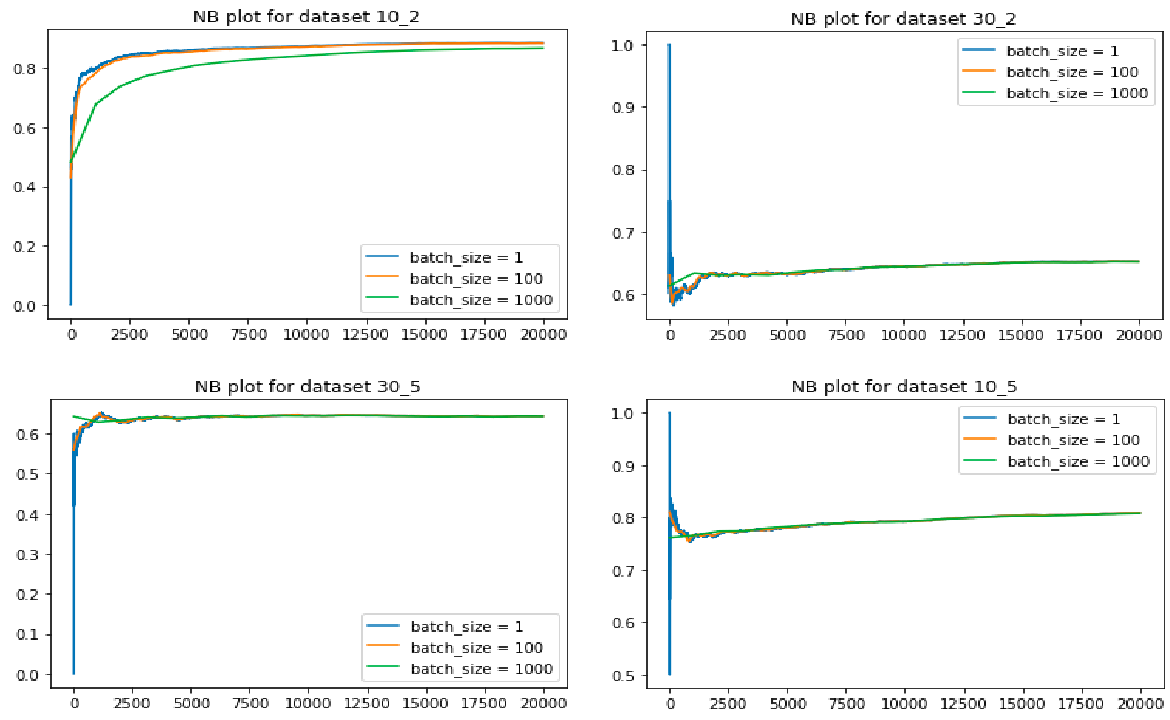

*Figure 2: Accuracies of NB.*

For all three classifiers, we can see that for the first dataset, the batch size affects the convergence of the accuracy to the final accuracy where larger batch sizes take more time than smaller ones. Furthermore, as expected, having a batch size of 1 creates a fluctuation at the start of the process, which we do not observe for a batch size of 100, where the accuracy plots are more smooth. Comparing the convergence of accuracies, we can see that the KNN takes approximately more samples to converge whereas, for example, the dataset 30_5 converges

very quickly for NB. These accuracy plots reflect how the models converge to their final accuracy and how the incoming data effects this behavior.
Table 2 shows the classification accuracy and runtime for the online ensemble classifiers. For both, we employ the interleaved-test-then-train method using batch sizes of 1, 100, and 1000.

*Table 2: Accuracies of online ensemble classifiers.*

| **Dataset** | **MV** | | | **WMV** | | |
|---|---|---|---|---|---|---|
| | **1** | **100** | **1000** | **1** | **100** | **1000** |
| **Hyperplane Dataset 10_2** | 87.4 | 87.4 | 85.8 | 88.4 | 88.3 | 86.7 |
| **Hyperplane Dataset 30_2** | 68.7 | 68.6 | 67.9 | 69.4 | 69.3 | 68.5 |
| **Hyperplane Dataset 10_5** | 87.4 | 87.3 | 85.8 | 88.2 | 88.1 | 86.6 |
| **Hyperplane Dataset 30_5** | 68.4 | 68.3 | 67.5 | 69.1 | 69.0 | 68.2 |
| **Runtime (s)** | 230.6 | 52.8 | 49.1 | 259.9 | 49.1 | 45.0 |

Table 3 shows the average accuracies of all classification models. From the table we see that the ensemble models perform better than the individual classification models. Furthermore, weighted majority voting classification model gives the highest mean accuracy because it combines the classification power of the individual classifiers by giving them certain weights. The weights were found on a trial and error basis by iterating through different combinations. From the trials, we notice that higher weight for the NB model is preferred whereas high weights for the HT classifier decreases accuracy. Initially weights were iterated from 1 to 5 where observed this trend. Then a more targeted approach was used where the weights for NB was increased ten-fold and the accuracy was calculated. Finally, the weights 10,2,100 for the KNN, HT, and NB respectively were found to be the best performing. The difference of the number of drift features does not affect the accuracy significantly, however increasing the noise from 10% to 30%, we see a huge difference.

*Table 3: Mean classification accuracy of all models.*

| **Classifier** | **Mean Accuracy** |
|---|---|
| **KNN** | 75.4 |
| **HT** | 73.3 |
| **NB** | 74.6 |
| **MV** | 77.5 |
| **WMV** | 78.3 |

Moreover, we can see that in terms of accuracy, the batch size does not have a significant effect. From Figure 4 we can see that for a batch size of 1, i.e. interleaved test-then-train on a single sample, we usually have the best accuracy. Increasing the batch size to 100 improves in the cases of some models (KNN) but usually degrades the performance, however not significantly. When the batch size is increased to 1000, we see that the accuracy usually decreases. In this case, the model is trained only when a certain amount (batch size) of data samples are collected. Therefore, the updates are averaged and thus the model has less updates. This can also be seen from the runtime values in Table 1 and Table 2 where the runtimes for the higher batch sizes are significantly less than models with a batch size of 1.

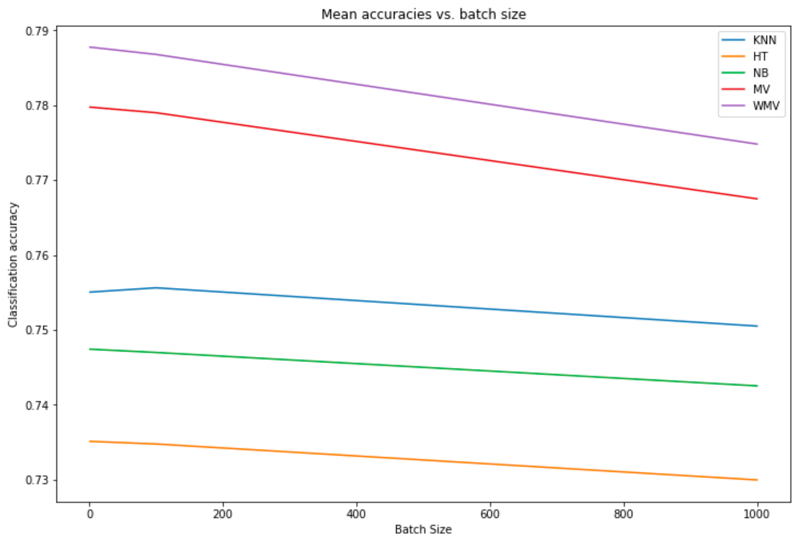

*Figure 4: Plot of the mean accuracies versus the batch size.*

Furthermore, we also see the discrepancy between the runtimes of the different classification models. The ensemble classifiers have the highest runtimes because they include all three individual models. However, for the individual models, we see that KNN takes the most time with NB taking the least time on average. This is because the NB model is simpler than the KNN model as the KNN model needs to calculate Euclidean distances between the test point and a certain number of recent training points. The number of training points to be used can be set using a parameter which is currently set to 2000. Although decreasing this value can reduce the runtime, the performance would be degraded. For KNN, we need to optimize the 'k' parameter. We set the value to $\sqrt{N}$, where N is the number of samples, and make sure that the value is an odd number as this is proved to give the best model [7].

Lastly, in order to increase the performance of the models, we implemented a 'Dynamic Weighted Majority Classifier' which uses multiple models of a certain classifier (Naïve Bayes by default) and:

' *…uses four mechanisms to cope with concept drift: It trains online learners of the ensemble, it weights those learners based on their performance, it removes them, also based on their performance, and it adds new experts based on the global performance of the ensemble [8].*'

Implementing this, we get 86.2%, 65.4%, 86.4%, and 64.6% classification accuracy for the four datasets respectively. The accuracy for the first dataset decreases but the model becomes more robust to the number of drift features and also the noise. In order to increase the performance further, one could increase the number of classifiers to use or the type of classifier to use.

## 4. Conclusion

In this project, we implement classification models on data streams. For this we generated sample data with varying noise and number of drift features. We used single classifiers such as KNN, HT, and NB, as well as ensemble classifiers such as MV, and WMV. We used a interleaved-test-then-train method to train and evaluate our models using both online and batch learning. From our results, we conclude that the WMV gives the best result. However the computational time for this model is high due to the fact that it uses all three single classifiers. Amongst the single classifier, KNN performs the best with NB second. Due to the simple nature of NB, it is the fastest model among all. We also observe that the increase in number of drift features usually decreases the accuracy (except for KNN) but the decrease is not significant. Furthermore, we conclude that the increase in noise significantly decreases our model performance, decreasing the accuracies from 80s to 60s. Next, from our results we also concluded that the increase in batch size negatively affects our performance after a certain threshold. We observed how the training procedure for data streams differs from whole datasets and how the stream effects the accuracy over samples processed. Lastly, we also presented a way to increase the accuracy and robustness of the models by using dynamic weighted majority classifiers which use multiple models and are designed specifically to cater to the issue of drifting features.

## References

[1] "skmultiflow.data.HyperplaneGenerator -- scikit-multiflow 0.5.3 Documentation," scikit-multiflow, [Online]. Available: https://scikit-multiflow.readthedocs.io/en/stable/api/generated/skmultiflow.data.HyperplaneGenerator.html#r52a7afcd91ae-1. [Accessed 3 May 2022].

[2] "skmultiflow.lazy.KNNClassifier -- scikit-multiflow 0.5.3 Documentation," scikit-multiflow, [Online]. Available: https://scikit-multiflow.readthedocs.io/en/stable/api/generated/skmultiflow.lazy.KNNClassifier.html. [Accessed 3 May 2022].

[3] "skmultiflow.trees.HoeffdingTreeClassifier -- scikit-multiflow 0.5.3 Documentation," scikit-multiflow, [Online]. Available: https://scikit-multiflow.readthedocs.io/en/stable/api/generated/skmultiflow.trees.HoeffdingTreeClassifier.html. [Accessed 3 May 2022].

[4] "skmultiflow.bayes.NaiveBayes -- scikit-multiflow Documentation," scikit-multiflow, [Online]. Available: https://scikit-multiflow.readthedocs.io/en/latest/api/generated/skmultiflow.bayes.NaiveBayes.html. [Accessed 3 May 2022].

[5] J. Brownlee, "How to Develop a Weighted Average Ensemble With Python," machinelearningmastery, 5 May 2021. [Online]. Available: https://machinelearningmastery.com/weighted-average-ensemble-with-python/. [Accessed 3 MAY 2022].

[6] A. Bifet, R. Gavaldà, G. Holmes and B. Pfahringer, "Chapter 6: Classification," in *Machine Learning for Data Streams: with Practical Examples in MOA*, The MIT Press, 2018.

[7] S. Thirumuruganathan, "A Detailed Introduction to K-Nearest Neighbor (KNN) Algorithm," 17 May 2010. [Online]. Available: https://saravananthirumuruganathan.wordpress.com/2010/05/17/a-detailed-introduction-to-k-nearest-neighbor-knn-algorithm/. [Accessed 3 May 2022].

[8] J. Z. Kolter and M. A. Maloof, "Dynamic Weighted Majority: An Ensemble Method for Drifting Concepts," *Journal of Machine Learning Research,* vol. 8, December 2007.

Note: References for code given in the .py file.