Classification Algorithm Evaluations on Hyperplane Data

Doruk Barkın Durak

21802255

Industrial Engineering

1. Dataset Generation Phase

For evaluation of algorithms, 4 datasets including 20000 instances with different noise and drifting features were created using Hyperplane Generator, that is supplied by scikit learn multiflow library. Datasets consists 10 levels of labels, last column is either a 0 or 1 value. Datasets named as their noise percentage values, followed by their drifting feature counts. Datasets are recorded to csv files because of not creating a different dataset at each run, resulting in increased performance efficiency.

2. Data Stream Classification with Used Classification Algorithms

3 algorithms of online single classifiers and 2 algorithms of online ensemble classifiers used in this face. Measurement criteria was the overall accuracy of each model based on 4 datasets provided. In MLP part 4 hidden layers of 100 neurons were used. Interleaved-test-then-train method was used for each dataset to find accuracies for batch size of 1. Results can be seen as

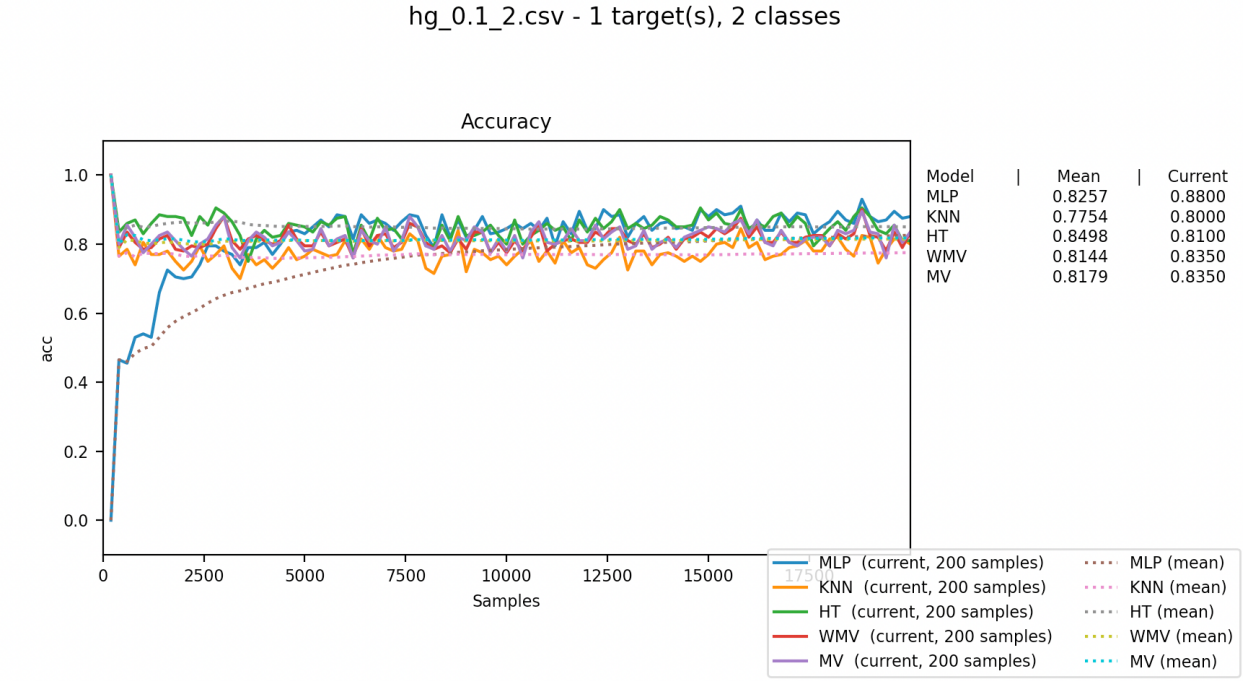

Figure 1: Temporal accuracies for used algorithms on dataset 0.1_2.csv

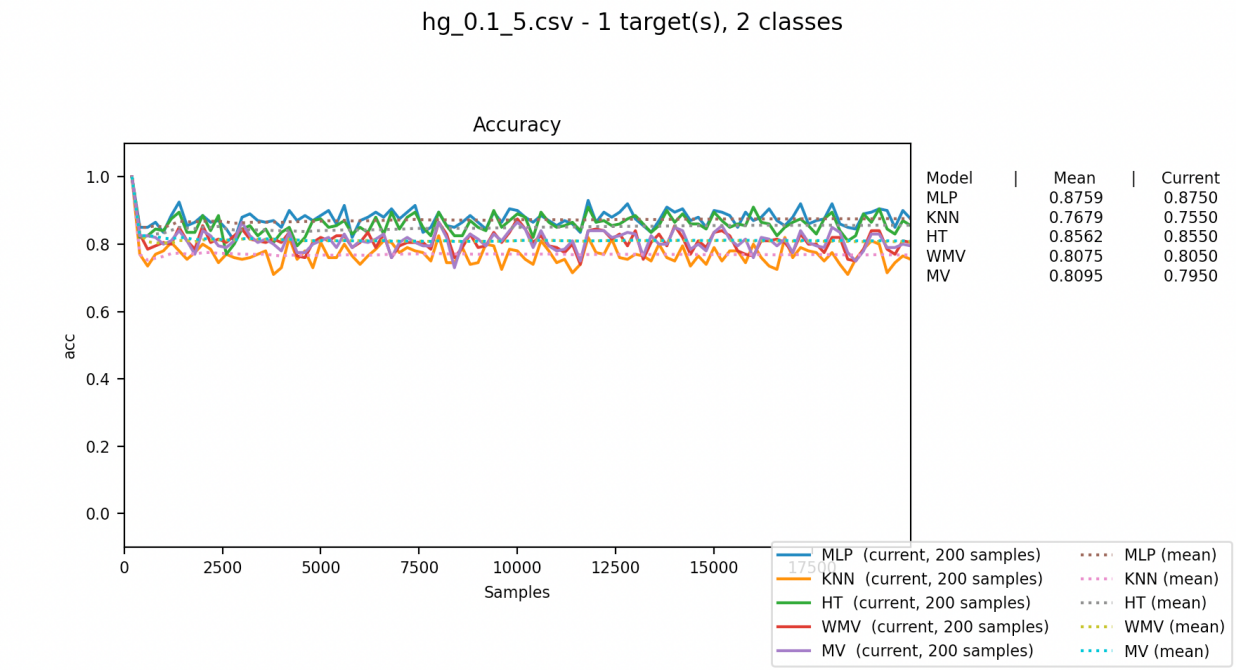

Figure 2: Temporal accuracies for used algorithms on dataset 0.1_5.csv

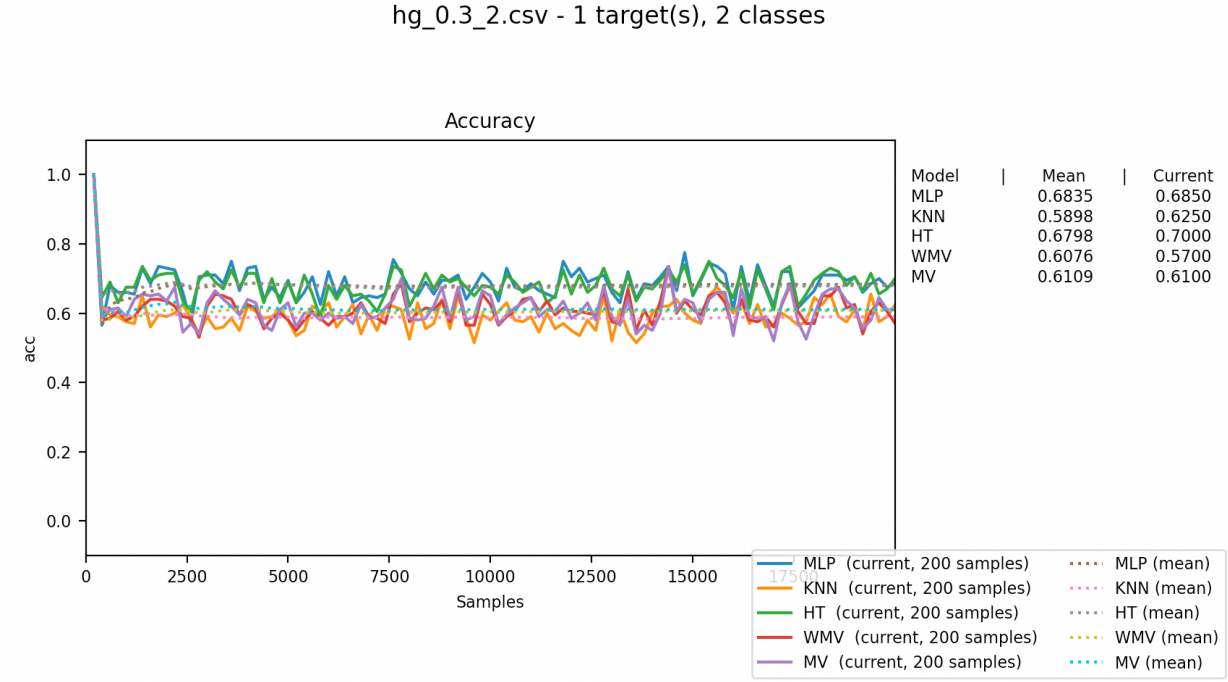

Figure 3: Temporal accuracies for used algorithms on dataset 0.3_2.csv

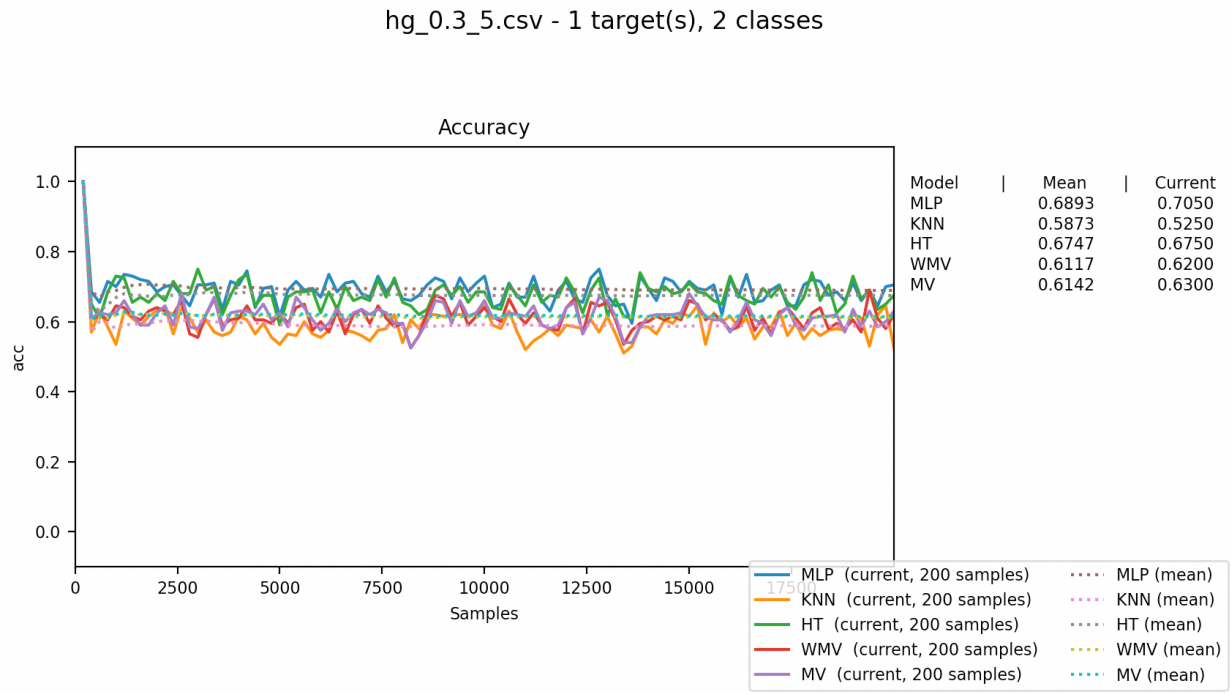

Figure 4: Temporal accuracies for used algorithms on dataset 0.3_5.csv

For first dataset, winning classification is Hoeffding Tree classifier, for other datasets, MLP method has the highest percentage in terms of temporal accuracy. An important thing that is

noticed is that the noise percentage effected the accuracy of the all classifiers more than changed drifting features. More noise percentage punished the data's accuracy for all classifiers about %25. Overall accuracies of all classifiers can be found below.

Classifier/Dataset	0.1_2	0.1_5	0.3_2	0.3_5
MLP	0,8257	0,8759	0,6835	0,6893
KNN	0,7754	0,7659	0,5898	0,5873
HT	0,8498	0,8562	0,6798	0,6747
WMV	0,8144	0,8075	0,6076	0,6117
MV	0,8179	0,8095	0,6109	0,6142

Table 1: Temporal Accuracy Rates for each classifier in each dataset.

The accuracies of classifications are close to each other for every dataset but MLP and HT online classifiers performed better than HT. In online ensemble methods there is little difference but Majority voting classifier performed better than weighted majority voting classifier in terms of accuracy. Accuracy is decreased significantly when noise percentage increased. The reason for that was drift in data and online classifiers are not sufficient enough to capture the differences in here. Difference between drifting features does not have significant effect on accuracy of data.

3. Effect of Batch Learning and Evaluations

Online single and ensemble classifiers were evaluated based on different batch sizes of 1, 100 and 1000 in order to see if batch size affect the accuracy of the data. For MLP:

Classifier	Dataset	Batch Size	Accuracy
MLP	0.1_2	1	0,8257
MLP	0.1_2	100	0,9004
MLP	0.1_2	1000	0,9011
MLP	0.1_5	1	0,8759
MLP	0.1_5	100	0,9032
MLP	0.1_5	1000	0,9051
MLP	0.3_2	1	0,6835
MLP	0.3_2	100	0,7029
MLP	0.3_2	1000	0,7024
MLP	0.3_5	1	0,6893
MLP	0.3_5	100	0,7071
MLP	0.3_5	1000	0,7084

Table 2: Batch size and accuracy comparison for MLP

It is seen that for MLP, increasing the batch size has a significant effect on first data set but for other datasets, increasing batch size still affects the accuracy but at a minor rate. Accuracy is still affected by changing the noise percentage and drifting features still does not have a significant effect on data. For batch sizes of 100 and 1000 there wasn't a significant change. Best results are obtained at 1000 batch size but for dataset with noise percentage %30 and 2 drifting features, optimal batch size is 100.

Classifier	Dataset	Batch Size	Accuracy
KNN	0.1_2	1	0,7754
KNN	0.1_2	100	0,7759
KNN	0.1_2	1000	0,7765
KNN	0.1_5	1	0,7659
KNN	0.1_5	100	0,7704
KNN	0.1_5	1000	0,7715
KNN	0.3_2	1	0,5898
KNN	0.3_2	100	0,5883
KNN	0.3_2	1000	0,5931
KNN	0.3_5	1	0,5873
KNN	0.3_5	100	0,5878
KNN	0.3_5	1000	0,5877

Table 3: Batch size and accuracy comparison for KNN

For KNN classifier, batch size does not have a significant effect on accuracy of the data. Accuracy is still affected because of changing noise percentage and drifting features still does not have significant effect on data. Best batch sizes are obtained at 1000 batch again but at dataset with %30 noise and 5 drifting features, optimal batch size is 100.

Classifier	Dataset	Batch Size	Accuracy
HT	0.1_2	1	0,8498
HT	0.1_2	100	0,8549
HT	0.1_2	1000	0,8615
HT	0.1_5	1	0,8562
HT	0.1_5	100	0,8661
HT	0.1_5	1000	0,866
HT	0.3_2	1	0,6798
HT	0.3_2	100	0,6921
HT	0.3_2	1000	0,6998
HT	0.3_5	1	0,6747
HT	0.3_5	100	0,6951
HT	0.3_5	1000	0,7066

Table 4: Batch size and accuracy comparison for HT

For HT classifier, accuracy of the data still does not have a significant difference for different batch sizes but increasing the batch size still increases the accuracy of data but at a minor rate. Optimal batch sizes are obtained at 1000 again.

Classifier	Dataset	Batch Size	Accuracy
WMV	0.1_2	1	0,8144
WMV	0.1_2	100	0,8159
WMV	0.1_2	1000	0,8166
WMV	0.1_5	1	0,8075
WMV	0.1_5	100	0,8104
WMV	0.1_5	1000	0,8059
WMV	0.3_2	1	0,6076
WMV	0.3_2	100	0,6085
WMV	0.3_2	1000	0,61
WMV	0.3_5	1	0,6117
WMV	0.3_5	100	0,6137
WMV	0.3_5	1000	0,6196

Table 5: Batch size and accuracy comparison for WMV

For weighted majority vote classifier accuracy of the data still does not have a significant difference for different batch sizes but increasing the batch size still increases the accuracy of data but at a minor rate. Optimal batch sizes are obtained at 1000 again but at dataset with %10 noise percentage and 5 drifting features, optimal batch size is at 100.

Classifier	Dataset	Batch Size	Accuracy
MW	0.1_2	1	0,8179
MW	0.1_2	100	0,8165
MW	0.1_2	1000	0,8173
MW	0.1_5	1	0,8095
MW	0.1_5	100	0,8105
MW	0.1_5	1000	0,8088
MW	0.3_2	1	0,6109
MW	0.3_2	100	0,6098
MW	0.3_2	1000	0,6069
MW	0.3_5	1	0,6142
MW	0.3_5	100	0,612
MW	0.3_5	1000	0,6194

Table 6: Batch size and accuracy comparison for MV

For majority voting classifier, for first dataset, the optimal batch size is at 1, for the second dataset the optimal batch size is at 1, for third dataset it is at 100 and for last dataset the optimal batch size is at 1000. This is the dataset with most variated accuracies based on changing the batch sizes, for the first assumption, noise percentage may be affecting the optimal batch size for this classifier.

4. Comparison of Online Classifiers vs. Ensemble Classifiers

Based on the provided data from Table 2 to Table 6, MLP and HT classifiers have more accuracy rates against the ensemble classifiers but interestingly, KNN method has lesser accuracy rates than ensemble methods. MLP in most cases is the winning classifier in most of the situations. This result may be about the content of the dataset that was generated and for other datasets results can be different for this comparison.

5. Comparison of Batch Classification for All Classifiers

Data was divided to training and testing data with percentages of 0.8 and 0.2 respectively. Model was trained according to training data and evaluated based on test data.

	MLP	KNN	HT	WMV	MV
0,1_2	0,90075	0,77075	0,8065	0,79125	0,7935
0,1_5	0,89825	0,78	0,8435	0,8205	0,8135
0.3_2	0,68775	0,576	0,65625	0,594	0,58525
0.3_5	0,689	0,58775	0,6665	0,61825	0,61575

Table 7: Accuracies for Online and Ensemble Classifiers for Batch Classification

In this method, winning solution is again MLP in all accuracy rates followed by HT and ensemble classifiers as KNN comes last. Because of the increased noise percentage rate, classifiers again performed poorly on last 2 datasets compared to first datasets with and average reduction of %25.

6. Ensemble Methods vs Batch Models

As comparison says, Batch models are better than Ensemble methods in terms of maximum accuracy. For minimum accuracy, ensemble methods win in this case because batch models also have the smallest accuracy. This conclusion is again may not be %100 correct because there is a possibility that the results may be change if the structure of the datasets were different.

7. Comparing all of the Methods for Accuracy

	MLP	MLP Batch	KNN	KNN Batch	HT	HT Batch	WMV	WMV Batch	MV	MV Batch
0,1_2	0,87573333	0,90075	0,77593333	0,77075	0,8554	0,8065	0,81563333	0,79125	0,81723333	0,7935
0,1_5	0,89473333	0,89825	0,76926667	0,78	0,86276667	0,8435	0,80793333	0,8205	0,8096	0,8135
0.3_2	0,69626667	0,68775	0,5904	0,576	0,69056667	0,65625	0,6087	0,594	0,6092	0,58525
0.3_5	0,7016	0,689	0,5876	0,58775	0,69213333	0,6665	0,615	0,61825	0,6152	0,61575

Table 8: Accuracy for all models evaluated

From this comparison, MLP batch method stands with the highest accuracy rates followed by Online MLP method. For KNN method for higher drifting features, batch method is more accurate. For HT method, Online method seems to be better than Batch method. For ensemble methods, at higher drifting features, accuracy of batches is better than online methods.

8. Accuracy Improvement

2 methods are proposed in this state, RandomForestClassifier and Dynamic Weighted Majority Classifier.

	RFC	MLP	DWMC
0.1_2	0.7945	0.7890	0.8656
0.1_5	0.8025	0.5529	0.8670
0.3_2	0.6253	0.6484	0.6735
0.3_5	0.6279	0.6562	0.6769

Table 9: Accuracy Improvement Comparison Table

As seen in above table, RFC performed better than MLP at lower rates of noise and also DWMC performed better also but at higher rates of noise, MLP performed better than RFC but not at DWMC so in order to improve the accuracy of the model we can implement Dynamic Weighted Majority Classifier for ensemble classifying.

References:

1) https://github.com/MAKman1/Bilkent-CS-Curriculum/tree/master/GE461/Project5
2) https://github.com/iremecem/GE461-Introduction-to-Data-Science/tree/main/Project%205
3) https://scikit-multiflow.github.io
4) https://scikit-learn.org/stable/
5) https://numpy.org
6) https://pandas.pydata.org
7) https://scikit-multiflow.readthedocs.io/en/stable/installation.html
8) https://scikit-multiflow.readthedocs.io/en/stable/api/generated/skmultiflow.data.HyperplaneGenerator.html
9) https://numpy.org/doc/1.16/reference/generated/numpy.random.RandomState.html