Hellinger Distance Trees for Imbalanced Streams

Authors: R. J. Lyon, J. M. Brooke, J. D. Knowles, B. W. Stappers

DATE: 22/07/2020

ESTABLISHED BY: HOUCEM BEN MAKHLOUF

MENTOR: MSC AMIR ABOLFAZLI

PROFESSOR: PROF. DR. EIRINI NTOUTSI

Outline:

1- Problem definition

2- GH-VFDT

3- Datasets

4- Implementation

5- Results

6- Conclusion

Problem definition

- Datasets in the real world are imbalanced to some degree
- Algorithms till 2014 were not well suited for data streams which leads to poor generalisation capabilities

- Our concern in this project is to find a solution for data streams coming from radio telescope‘s data processing pipeline, this data is composed of one instance which belongs to the positive minority class every 10,000 instances which belongs to the majority class

GH-VFDT

Gaussian Hellinger Very Fast Decision Tree as a solution for imbalanced streamed data

Main idea:

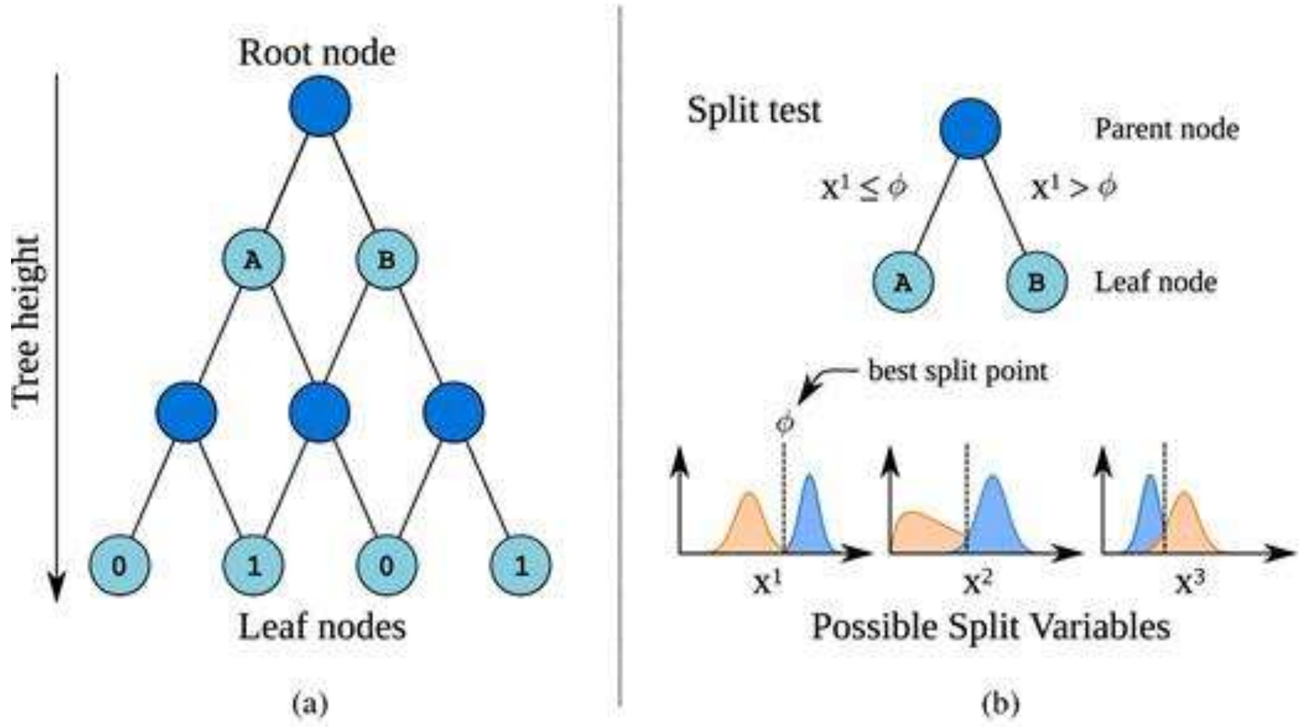
maximize classification performance on candidate data streams, which are heavily imbalanced in favour of the non-pulsar class

Advantages:

- Better performance in terms of accuracy in the favor of Pulsar classes
- skew insensitive split criterion using Hellinger distance
- Memory requirements are optimized comparing to those required for HD-VFDT O(lf2c)
- Suitable for very high throughput data streams

Very Fast Decision Tree:

- The algorithm uses tree learning, whereby the data are partitioned using feature split point tests that aim to maximize the separation of pulsar and non-pulsar candidates. This involves first choosing the variable that acts as the best class separator, and then finding a numerical threshold 'test point' for that variable that maximises class separability.



- A decision tree is learned by recrusively replacing leaves by test nodes, starting at the root. The feature to test at a node is chosen by comparing all the available features and choosing the best one according to some heuristic measure.

- It is sufficient to use a small sample of the available examples when choosing the split attribute at any given node

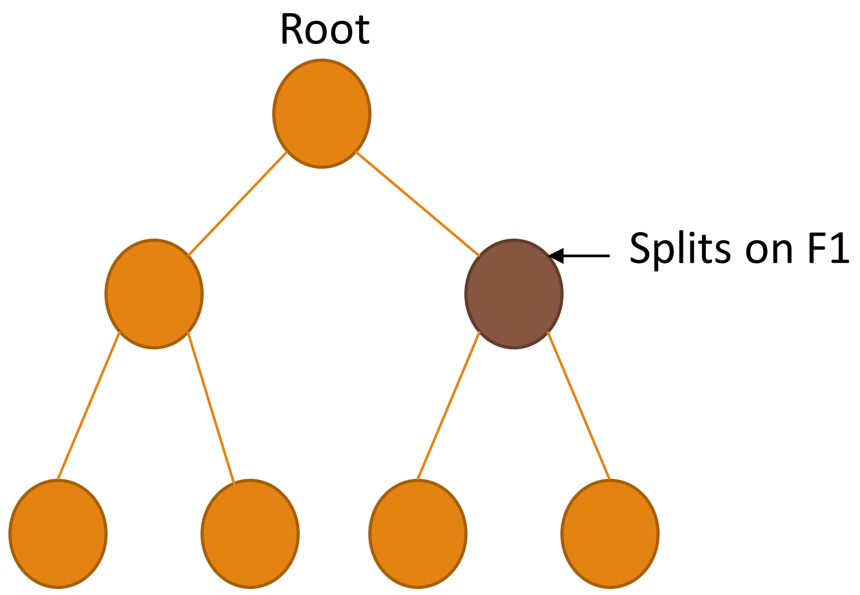
- VFDT reduces ist memory requirements by temporally desactivating learning in the least promosing nodes

The key feature of the GH-VFDT:

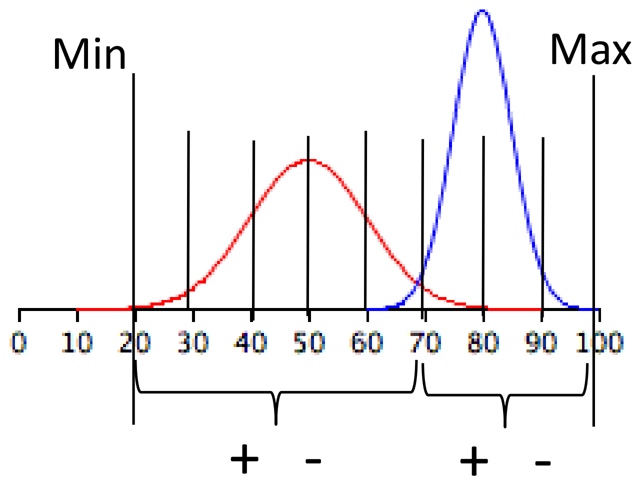
- The use of Hellinger distance criterion for the evaluation of the split points during learning
- Only the mean and the standard deviation of the distributions of majority and minority classes of the corresponding features need to be known
- This measure <u>preven</u>t the classifier from becoming biased to non-pulsar class

How? =>

- by modelling each feature distribution as a Guassian, the distance between majority and minority distributions P and Q can be measured
- Then getting the one that maximises the Hellinger distance between both distribution



The input has the form (Xi , Ci) with Xi=(F1,F2,F3) and Ci=label{0,1}



$$\text{mean}_{new} = \frac{Value - mean\_previous}{number\ of\ instances} + \text{mean}_{\text{previous}}$$

$$variance = \frac{variance\_sum}{number\ of\ instances - 1}$$

$$variance_{sum} = variance_{sumprevious} + \text{number of instances} \times \left(value - mean_{previous}\right) \times (value - mean_{new})$$

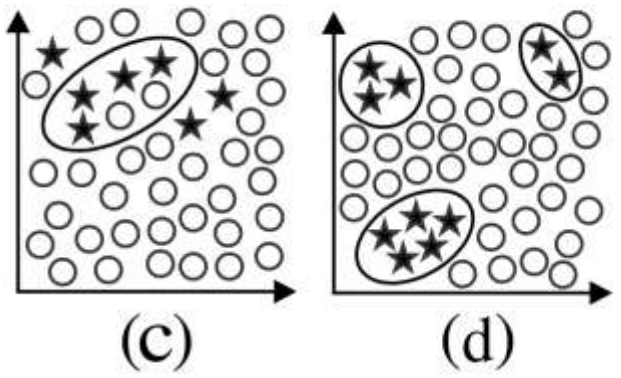
Datasets

Main problems faced in imbalanced data:

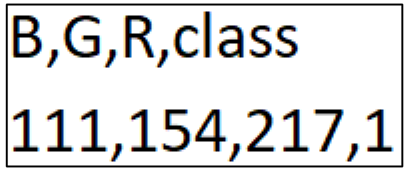
small sample size: rare minority class to learn from

class inseparability: distributions of minority class and majority class are superposed (fig. c)

small disjuncts: distribution of minority class is composed of disjuncts small distribution (fig. d)

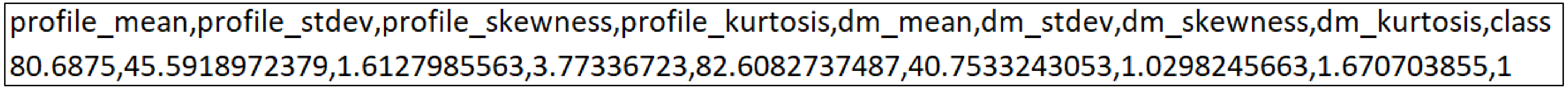


Dataset	instances	Attributes /type	Balance	Positive examples	Negative examples
Pulsar 1	90,192	9/Continuous	+1 : -75	1,196	88,996
Pulsar 2	17,898	9/Continuous	+1 : -10	1,639	16,259
Skin	245,057	3/Discrete	+1 : -4	50,859	194,198



```
B,G,R,class
111,154,217,1
```

The skin dataset is collected by randomly sampling B,G,R values from face images of various age groups (young, middle, and old)



```
profile_mean,profile_stdev,profile_skewness,profile_kurtosis,dm_mean,dm_stdev,dm_skewness,dm_kurtosis,class
80.6875,45.5918972379,1.6127985563,3.77336723,82.6082737487,40.7533243053,1.0298245663,1.670703855,1
```

Pulsars are a rare type of Neutron star that produce radio emission detectable here on Earth

Implementation

- Implementation of GHVFDT using the source code of Scikit-Multiflow
- Processing the datasets
- Implementation of training and testing schema
- Running experiments

Implementation of GHVFDT

Following points were implemented:

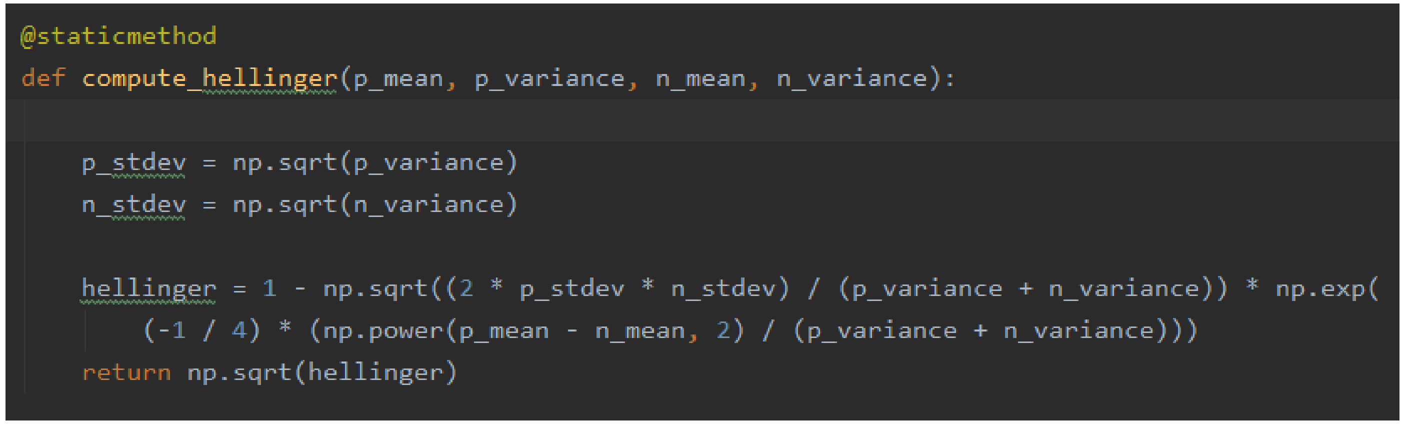
- new tree: ghvfdt_tree.py
- new split criterion: gaussian_hellinger_criterion.py
- new attribute observer: gh_numeric_attribute_class_observer_gaussian.py
- 3 new types of nodes: gh_active_learning_node.py, gh_learning_node_nb.py and gh_learning_node_nb_adaptive.py

Some details of GHVFDT

- **split criterion:**

one of the methods the criterion has is compute_Hellinger(), which calculates the Hellinger distance as in Eq.2 in the paper

$$d_H(P,N) \quad = \quad \sqrt{1 - \sqrt{\frac{2\sigma_1\sigma_2}{\sigma_1^2+\sigma_2^2}} e^{-\frac{1}{4}\frac{(\mu_1-\mu_2)^2}{\sigma_1^2+\sigma_2^2}}} . \qquad (2)$$



```
@staticmethod
def compute_hellinger(p_mean, p_variance, n_mean, n_variance):

    p_stdev = np.sqrt(p_variance)
    n_stdev = np.sqrt(n_variance)

    hellinger = 1 - np.sqrt((2 * p_stdev * n_stdev) / (p_variance + n_variance)) * np.exp(
        (-1 / 4) * (np.power(p_mean - n_mean, 2) / (p_variance + n_variance)))
    return np.sqrt(hellinger)
```

Some details of GHVFDT

- **attribute observer:**

One of the main methods the observer has is get_best_evaluated_split_suggestions(), which returns the best split suggestion of the corresponding feature

It is here where the mean and variance values are gotten from the gaussian estimator and where the Hellinger distance is computed

Ps: say we have 3 features in the data, then each leaf has
3 observers (for each feature) and each observer has
2 gaussian estimator (for each class)

Processing the datasets

I dealt with the Skin, pulsar1 and pulsar2 datasets obtained from the UCI machine learning repository.

What I implemented:

- a function that create a training set (.csv) with 200 random positive and 1000 random negative samples as in the paper
- a function that create 16 test sets (.csv) with different imbalance rates (+1:-10, +1:-100, +1:-1000, +1:-10000) and different labelling rates (10%, 50%, 75%, 100%) as in the paper

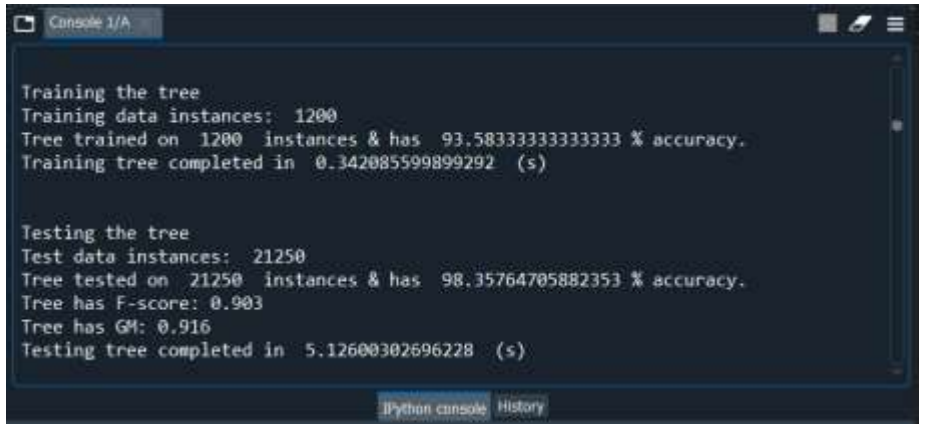
Implementation of training and testing schema

Two functions were implemented:

- train_tree() : for pre-training
- Test_tree()

In these functions, the streams were created from the .csv files. Then, an incremental test-then-train approach was adopted as mentioned in the paper.

At the end the F-score and GM were calculated



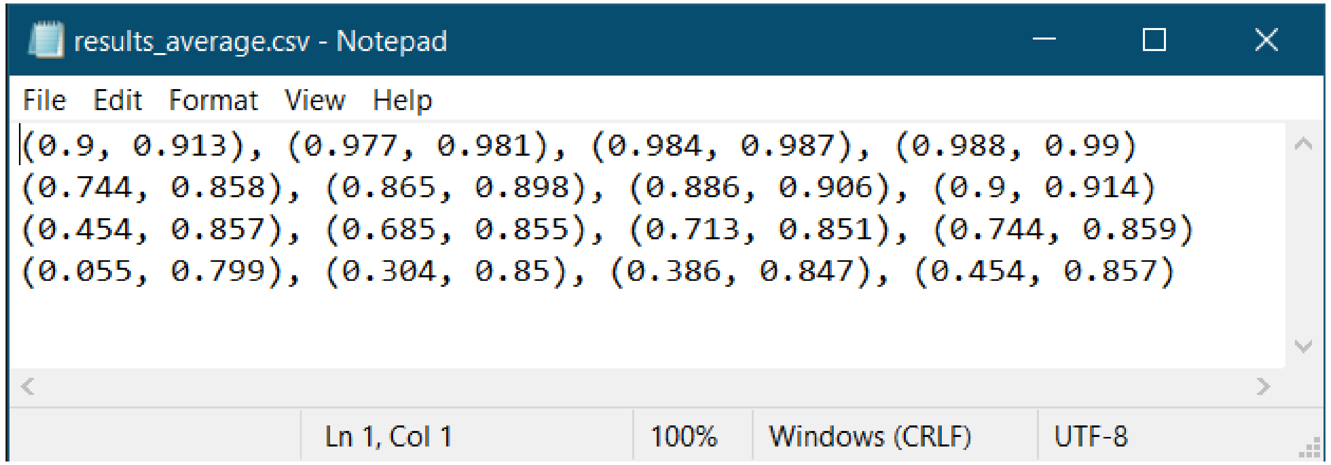
```
Training the tree
Training data instances:  1200
Tree trained on  1200  instances & has  93.58333333333333 % accuracy.
Training tree completed in  0.342085599899292  (s)


Testing the tree
Test data instances:  21250
Tree tested on  21250  instances & has  98.35764705882353 % accuracy.
Tree has F-score: 0.903
Tree has GM: 0.916
Testing tree completed in  5.12600302696228  (s)
```

Running experiments

Trying to reproduce the tables in the paper, experiments were done with the new GHVFDT and the old HDVFDT on the 3 datasets

One experiment consists in pre-training the tree on the train set then testing it on the different test sets.

For each tree, this experiment is repeated 10 times with different random arrangement of the data to get average values of the F-score and GM.



```
(0.9, 0.913), (0.977, 0.981), (0.984, 0.987), (0.988, 0.99)
(0.744, 0.858), (0.865, 0.898), (0.886, 0.906), (0.9, 0.914)
(0.454, 0.857), (0.685, 0.855), (0.713, 0.851), (0.744, 0.859)
(0.055, 0.799), (0.304, 0.85), (0.386, 0.847), (0.454, 0.857)
```

Results: F-score/G-Mean

Dataset	Balance	+1 : -10			
	Labelling (%)	10	50	75	100
pulsar	HD-VFDT	0.885 / 0.929	0.913 / 0.939	0.938 / 0.958	0.942 / 0.964
	GH-VFDT	0.856 / 0.93	0.895 / 0.925	0.913 / 0.931	0.918 / 0.931
Pulsar2	HD-VFDT	0.802 / 0.872	0.878 / 0.929	0.898 / 0.932	0.919 / 0.94
	GH-VFDT	0.791 / 0.871	0.87 / 0.923	0.896 / 0.926	0.913 / 0.929
Skin	HD-VFDT	0.958, 0.978	0.991, 0.995	0.994, 0.997	0.995, 0.998
	GH-VFDT	0.9 / 0.913	0.977 / 0.981	0.984 / 0.987	0.988 / 0.99

Dataset	Balance	+1 : -100			
	Labelling (%)	10	50	75	100
pulsar	HD-VFDT	0.858 / 0.928	0.902 / 0.94	0.924 / 0.957	0.932 / 0.962
	GH-VFDT	0.769 / 0.933	0.89 / 0.927	0.91 / 0.932	0.916 / 0.932
Pulsar2	HD-VFDT	0.494 / 0.848	0.556 / 0.906	0.648 / 0.916	0.698 / 0.924
	GH-VFDT	0.44 / 0.871	0.519 / 0.914	0.608 / 0.92	0.68 / 0.927
Skin	HD-VFDT	0.792, 0.91	0.916, 0.964	0.945, 0.974	0.958, 0.98
	GH-VFDT	0.744 / 0.858	0.865 / 0.898	0.886 / 0.906	0.9 / 0.914

Dataset	Balance	+1 : -1000			
	Labelling (%)	10	50	75	100
pulsar	HD-VFDT	0.477 / 0.852	0.738 / 0.907	0.819 / 0.92	0.829 / 0.922
	GH-VFDT	0.275 / 0.918	0.651 / 0.915	0.721 / 0.919	0.767 / 0.919
Pulsar2	HD-VFDT	0.043 / 0.595	0.113 / 0.848	0.182 / 0.861	0.201 / 0.85
	GH-VFDT	0.056 / 0.793	0.097 / 0.901	0.138 / 0.888	0.178 / 0.882
Skin	HD-VFDT	0.386, 0.762	0.645, 0.879	0.724, 0.894	0.792, 0.912
	GH-VFDT	0.454 / 0.857	0.685 / 0.855	0.713 / 0.851	0.744 / 0.859

Dataset	Balance	+1 : -10000			
	Labelling (%)	10	50	75	100
pulsar	HD-VFDT	0.1 / 0.999	0.264 / 0.788	0.368 0.846	0.362 / 0.841
	GH-VFDT	0.042 / 0.798	0.174 / 0.904	0.247 / 0.938	0.301 / 0.919
Pulsar2	HD-VFDT	0.043 / 0.595	0.011 / 0.596	0.016 / 0.796	0.014 / 0.698
	GH-VFDT	0.056 / 0.793	0.015 / 0.794	0.018 / 0.995	0.017 / 0.897
Skin	HD-VFDT	0.0 / 0.0	0.282 / 0.66	0.336 / 0.73	0.386 / 0.763
	GH-VFDT	0.055 / 0.799	0.304 / 0.85	0.386 / 0.847	0.454 / 0.857

Example

Dataset	Balance	+1 : -10			
	Labelling (%)	10	50	75	100
pulsar	HD-VFDT	0.885 / 0.929	0.913 / 0.939	0.938 / 0.958	0.942 / 0.964
	GH-VFDT	0.856 / 0.93	0.895 / 0.925	0.913 / 0.931	0.918 / 0.931
Pulsar2	HD-VFDT	0.802 / 0.872	0.878 / 0.929	0.898 / 0.932	0.919 / 0.94
	GH-VFDT	0.791 / 0.871	0.87 / 0.923	0.896 / 0.926	0.913 / 0.929
Skin	HD-VFDT	0.958 / 0.978	0.991 / 0.995	0.994 / 0.997	0.995 / 0.998
	GH-VFDT	0.9 / 0.913	0.977 / 0.981	0.984 / 0.987	0.988 / 0.99

Dataset	Balance	+1 : -1000			
	Labelling (%)	10	50	75	100
pulsar	HD-VFDT	0.477 / 0.852	0.738 / 0.907	0.819 / 0.92	0.829 / 0.922
	GH-VFDT	0.275 / 0.918	0.651 / 0.915	0.721 / 0.919	0.767 / 0.919
Pulsar2	HD-VFDT	0.043 / 0.595	0.113 / 0.848	0.182 / 0.861	0.201 / 0.85
	GH-VFDT	0.056 / 0.793	0.097 / 0.901	0.138 / 0.888	0.178 / 0.882
Skin	HD-VFDT	0.386 / 0.762	0.645 / 0.879	0.724 / 0.894	0.792 / 0.912
	GH-VFDT	0.454 / 0.857	0.685 / 0.855	0.713 / 0.851	0.744 / 0.859

Dataset	Balance	+1 : -10			
	Labelling (%)	10	50	75	100
pulsar	HD-VFDT	.860/.922	.855/.923	.855/.933	.850/.920
	GH-VFDT	.858/.918	.852/.915	.851/.909	.850/.917
Skin	HD-VFDT	.737/.893	.746/.898	.726/.877	.727/.888
	GH-VFDT	.815/.911	.824/.904	.835/.913	.816/.914

Dataset	Balance	+1 : -100			
	Labelling (%)	10	50	75	100
pulsar	HD-VFDT	.457/.835	.472/.929	.529/.919	.493/.931
	GH-VFDT	.518/.903	.483/.920	.536/.916	.552/.916
Skin	HD-VFDT	.277/.887	.292/.885	.319/.903	.263/.892
	GH-VFDT	.399/.898	.478/.915	.441/.904	.430/.900

Conclusion

In this work:

- Reading the paper and analysing different parts of the Concepts
- getting to know scikit-multiflow Framework
- Implementation of the HD-VFDT and GH-VFDT algorithms
- Using Pulsar and Skin datasets to test and train the model
- Compare results

➢GH-VFDT improves minority class recall rates on imbalanced data and it is a good memory optimiser

Thank you