Collect tweets from Twitter API and use batch and online ML methods

Done by:
Salma EZZINA
Chaima ELMESSAI

I. Sentiment analysis

Steps:

1. Scrapping tweets that are related to the word ‘UKRAINE’ using the twitter api

Steps:

1. Scrapping tweets that are related to the word ‘UKRAINE’ using the twitter api
2. Labellisation: using SentimentIntensityAnalyzer from nltk.sentiment.vader import

Steps:

1. Scrapping tweets that are related to the word ‘UKRAINE’ using the twitter api
2. Labellisation: using SentimentIntensityAnalyzer from nltk.sentiment.vader import
3. Text preprocessing and feature extraction:

   2 approaches: Tfidf and Word embeddings

Steps:

1. Scrapping tweets that are related to the word ‘UKRAINE’ using the twitter api
2. Labellisation: using SentimentIntensityAnalyzer from nltk.sentiment.vader import
3. Text preprocessing and feature extraction:

   2 approaches: Tfidf and Word embeddings

4. Saving the dataset into a csv file for batch comparaison

Steps:

1. Scrapping tweets that are related to the word ‘UKRAINE’ using the twitter api
2. Labellisation: using SentimentIntensityAnalyzer from nltk.sentiment.vader import
3. Text preprocessing and feature extraction:

   2 approaches: Tfidf and Word embeddings

4. Saving the dataset into a csv file for batch comparaison
5. Training models: kNN, one Vs one logistic regression, Hoeffding tree for online task

Steps:

1. Scrapping tweets that are related to the word ‘UKRAINE’ using the twitter api
2. Labellisation: using SentimentIntensityAnalyzer from nltk.sentiment.vader import
3. Text preprocessing and feature extraction:

    2 approaches: Tfidf and Word embeddings

4. Saving the dataset into a csv file for batch comparaison
5. Training models: kNN, one Vs one logistic regression, Hoeffding tree for online task
6. Training models: kNN and one Vs one logistic regression for batch data (2 cases: using all the data as train & splitting data)

Online classification

Each 10 samples:

kNN results

```
Samples processed: 9
BalancedAccuracy: 20.83%
WeightedF1: 41.30%
CohenKappa: -3.45%
```

one_vs_one logistic regression results

```
Samples processed: 9
BalancedAccuracy: 16.67%
WeightedF1: 40.00%
CohenKappa: 1.64%
```

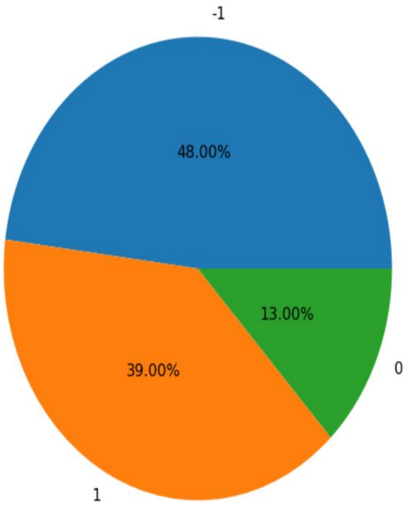
Hoeff tree results

```
Samples processed: 9
BalancedAccuracy: 20.83%
WeightedF1: 43.50%
CohenKappa: 1.64%
```

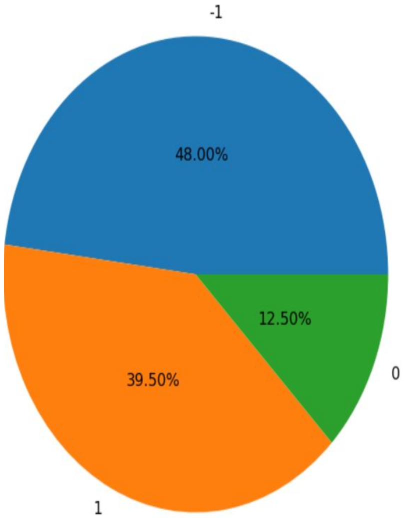
one_vs_one logistic regression results With embeddings

```
1/1 [==============================] - 0s 107ms/step
Samples processed: 9
BalancedAccuracy: 22.22%
WeightedF1: 29.32%
CohenKappa: -27.27%
--------------------------------------------------------------
```

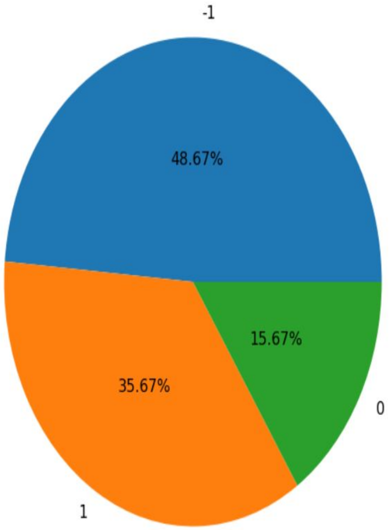
Study data distribution, each 100 samples



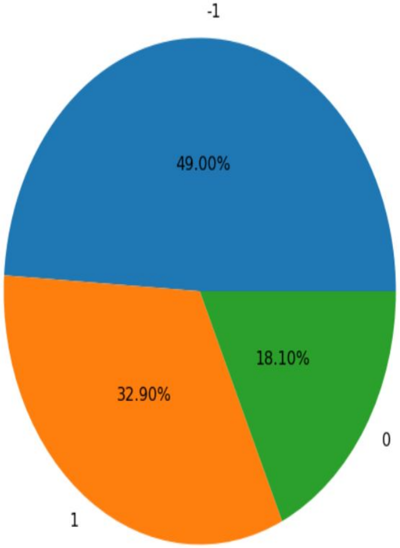
100 samples



200 samples

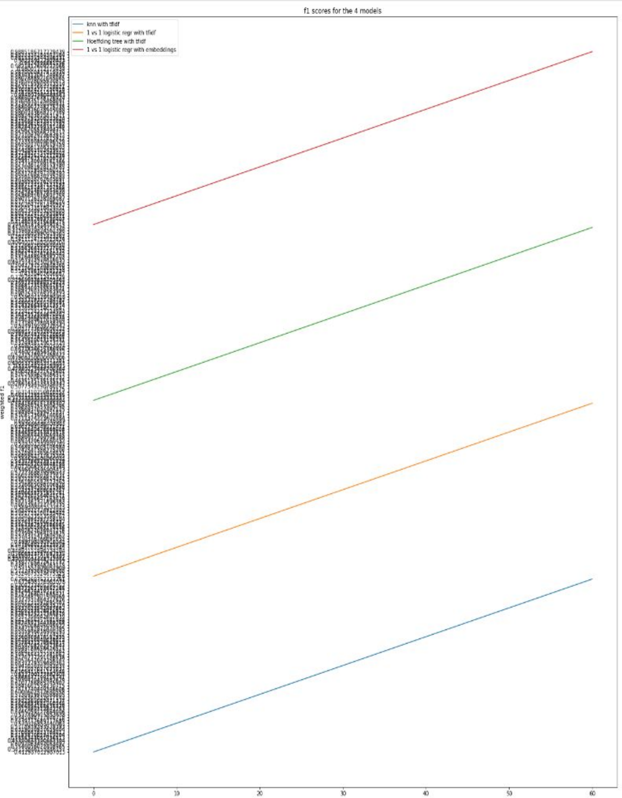


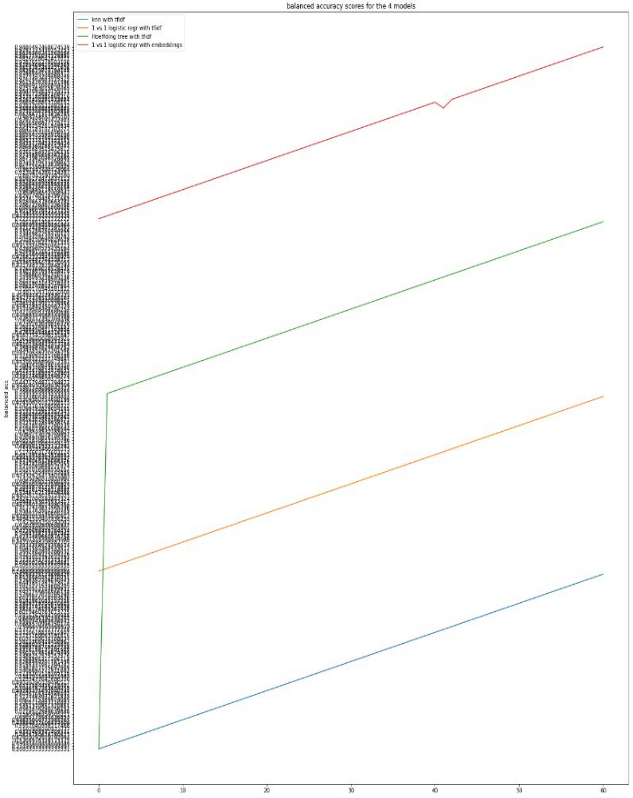
300 samples

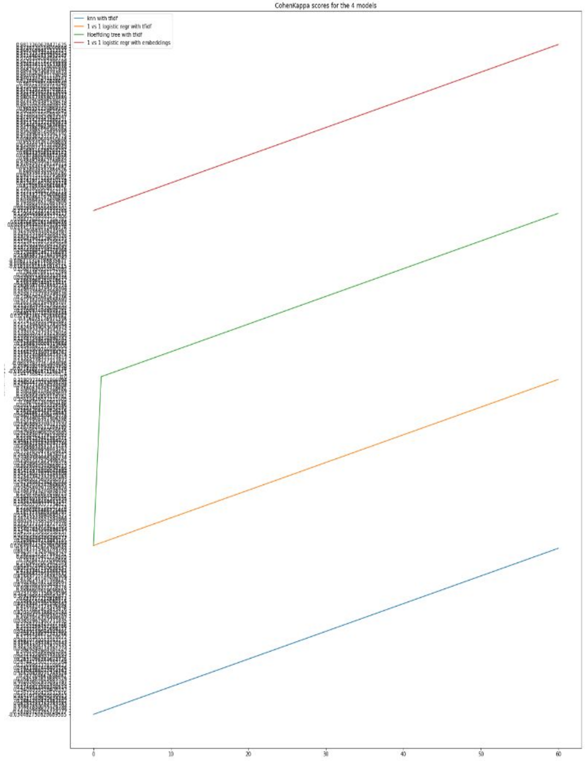


1000 samples

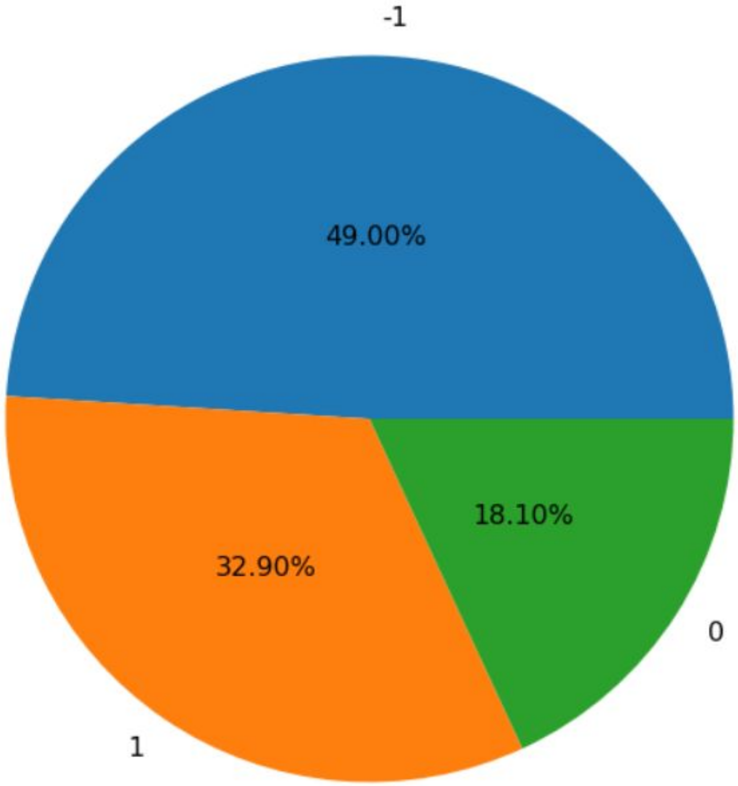
Plots of the scores:







Batch learning



Data distribution on 1000 samples

i. kNN with Tfidf

kNN

All the dataset trained: Results

weighted f1 score 0.8048374568654817

	precision	recall	f1-score	support
-1	0.79	0.91	0.85	490
0	0.73	0.70	0.71	181
1	0.89	0.71	0.79	329
accuracy			0.81	1000
macro avg	0.80	0.77	0.78	1000
weighted avg	0.81	0.81	0.80	1000

Train data results

weighted f1 score 0.7868823231775172

	precision	recall	f1-score	support
-1	0.77	0.89	0.83	334
0	0.72	0.66	0.69	120
1	0.87	0.70	0.78	216
accuracy			0.79	670
macro avg	0.79	0.75	0.77	670
weighted avg	0.79	0.79	0.79	670

Test data results

weighted f1 score 0.70696864746043

	precision	recall	f1-score	support
-1	0.69	0.88	0.77	156
0	0.59	0.44	0.50	61
1	0.84	0.64	0.72	113
accuracy			0.72	330
macro avg	0.71	0.65	0.67	330
weighted avg	0.72	0.72	0.71	330

ii. 1 Vs 1 Logistic Regression with Tfidf

oneVsone_logreg

All the dataset trained: Results

weighted f1 score 0.8959406650529741

	precision	recall	f1-score	support
-1	0.83	1.00	0.91	490
0	1.00	0.60	0.75	181
1	1.00	0.92	0.96	329
accuracy			0.90	1000
macro avg	0.94	0.84	0.87	1000
weighted avg	0.92	0.90	0.90	1000

Train data results

weighted f1 score 0.8345681254414153

	precision	recall	f1-score	support
-1	0.77	1.00	0.87	334
0	1.00	0.40	0.57	120
1	1.00	0.87	0.93	216
accuracy			0.85	670
macro avg	0.92	0.76	0.79	670
weighted avg	0.88	0.85	0.83	670

Test data results

weighted f1 score 0.654865936771163

	precision	recall	f1-score	support
-1	0.61	0.99	0.75	156
0	1.00	0.16	0.28	61
1	0.98	0.57	0.72	113
accuracy			0.69	330
macro avg	0.86	0.57	0.59	330
weighted avg	0.81	0.69	0.65	330

iii. 1 Vs 1 Logistic Regression with embeddings

oneVsone_logreg

All the dataset trained: Results

weighted f1 score 0.9980000911460346

	precision	recall	f1-score	support
-1	1.00	1.00	1.00	490
0	0.99	1.00	1.00	181
1	1.00	1.00	1.00	329
accuracy			1.00	1000
macro avg	1.00	1.00	1.00	1000
weighted avg	1.00	1.00	1.00	1000

Train data results

weighted f1 score 0.9970173213312356

	precision	recall	f1-score	support
-1	1.00	0.99	1.00	334
0	1.00	1.00	1.00	120
1	0.99	1.00	1.00	216
accuracy			1.00	670
macro avg	1.00	1.00	1.00	670
weighted avg	1.00	1.00	1.00	670

Test data results

weighted f1 score 0.6751498161518419

	precision	recall	f1-score	support
-1	0.67	0.88	0.76	156
0	0.62	0.43	0.50	61
1	0.78	0.56	0.65	113
accuracy			0.69	330
macro avg	0.69	0.62	0.64	330
weighted avg	0.70	0.69	0.68	330

II. Community Group Detection

Steps:

This task is a clustering task.

- We begin with a data preparation step in which we extract TF-IDF features
- Then we will test :
  - online learning algorithms: Kmeans, STREAMKmeans and DBSTREAM.
  - batch learning algorithms : Kmeans, DBSCAN, AgglomerativeClustering.
- Finally we will evaluate these algorithms using silhouette score

Batch clustering

- The issue with the clustering algorithms is to determine the number of clusters.
- The number of clusters found using the DBSCAN algorithm is equal to 103.
- We used n_clusters=103 for K_means and AgglomerativeClustering.
- We found the following performances :

  - For DBSCAN : Silhouette Coefficient = 0.433

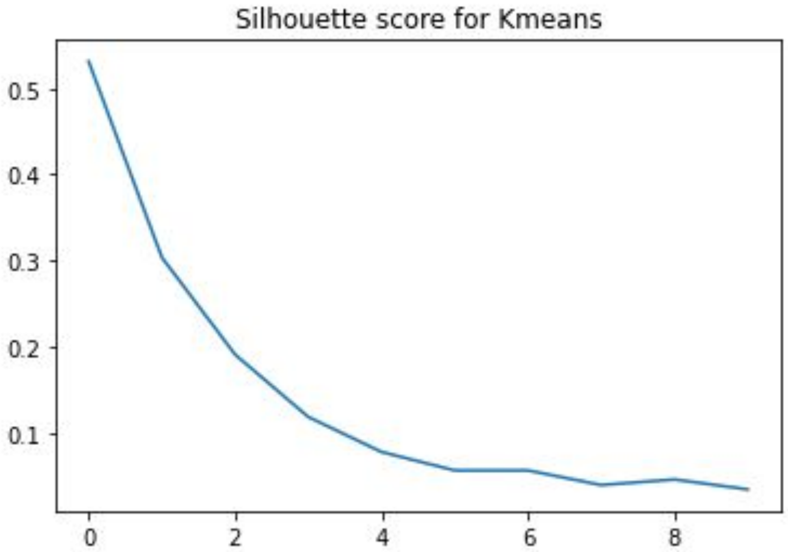
  - For Kmeans : Silhouette Coefficient = 0.414

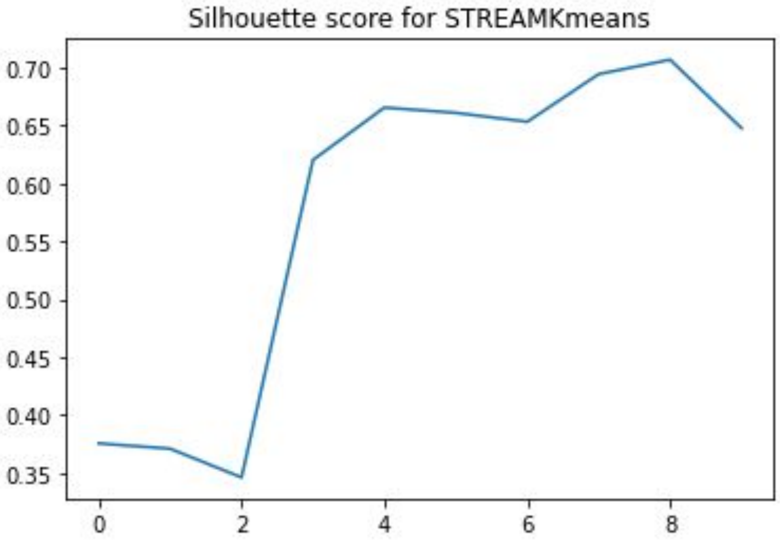
  - For Agglomerative Clustering : Silhouette Coefficient = 0.467

Online clustering

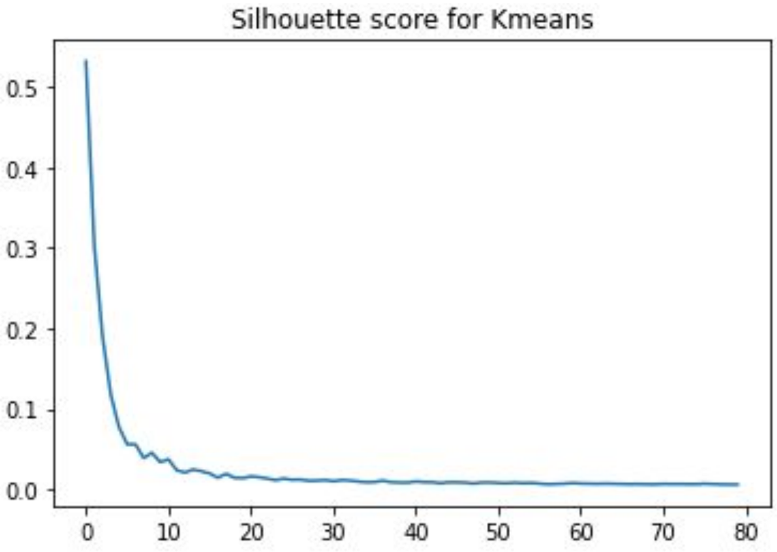
- We displayed for each ten samples the coefficient silhouette and we plotted the evolution of this coefficient in function of the number of samples.
- When we set the number of clusters to 103 like in the batch learning algorithms we get a coefficient silhouette equal to 1 for the STREAMKmeans.
- We fixed the number of clusters to n_clusters=3

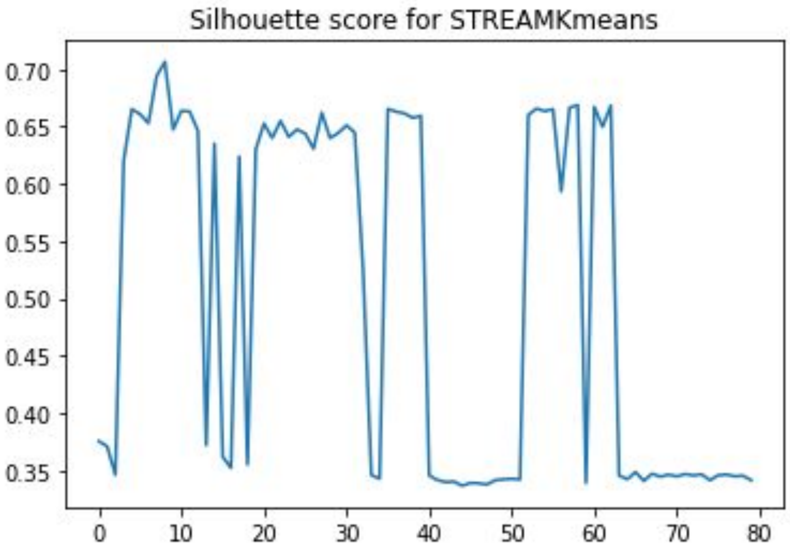
- For the first 10 samples





- After 80 samples





Conclusions

Batch is always faster, even with the same algorithms

You can’t know if your model is overfitting

Online models can detect change distributions in data and are better in case of frequent updates

Thank you!