Discussão das Diferenças entre Implementações do Algoritmo MINAS

Luís Henrique Puhl de Souza
Orientador: Prof. Dr. Hermes Senger

5 de agosto de 2020

Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Departamento de Computação
Programa de Pós-Graduação em Ciência da Computação

Introdução

Introdução

Este documento tem por objetivo apresentar o algoritmo Minas e suas implementações guiando discussões sobre os detalhes e decisões nas implementações.

- Um sistema para detecção de intrusão em Redes IoT implementando em névoa;
- A hipótese do trabalho é que o algoritmo MINAS pode ser distribuído em nós de nuvem e névoa reduzindo a latência e com pouco comprometimento na qualidade de detecção.
- Fundamentos
  - Métodos Detecção de Novidade;
  - Ambientes de computação Distribuída;
  - Plataformas de processamento distribuído de fluxos.

Algoritmo MINAS

- Modelo de aprendizado *Offline-Online*;
- Transformação dos dados analisados para o espaço $\mathbb{R}^d$;
- Modelo de classificação com *Clusters*;
- Função de classificação baseada em distância euclideana;
- Algoritmo de agrupamento para identificação de novos padrões;
- Classificação de novos padrões entre recorrência, extensão e novidade;

Proposta

Proposta da Pesquisa

- Implementar a distribuição do algoritmo MINAS em nuvem e névoa conforme arquitetura IDSA-IoT;
- Paralelizar o método de classificação do algoritmo MINAS.

Metodologia

- Plataforma de processamento distribuído;
- Estratégias de implementação da arquitetura IDSA-IoT;
- Experimentação com a distribuição do algoritmo MINAS em ambientes;
- Métricas de qualidade de classificação para validação da implementação;
- Métricas de escalabilidade.

Proposta

O sistema M-FOG é dividido em 5 módulos subdivididos em 2 grupos.

Módulos principais implementam o algoritmo MINAS

- módulo treinamento (*Training Module*);
- módulo classificador (*Classification Module*);
- módulo detector de novidades (*Novelty Detection Module*).

Módulos auxiliares, utilizados para avaliação

- módulo auxiliar *source* (fonte);
- módulo auxiliar *sink* (sorvedouro, consumidor final).

Proposta

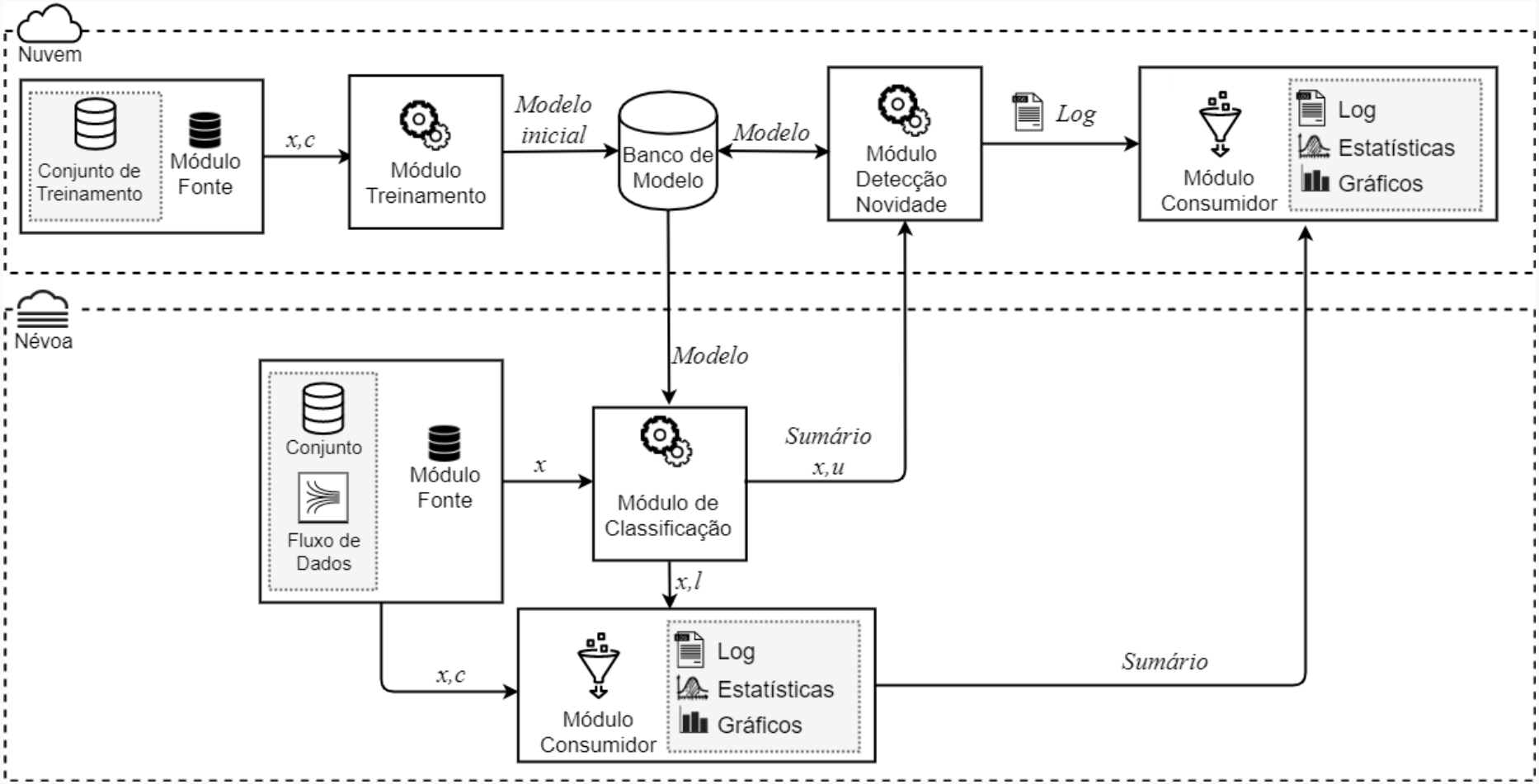


Figura 1: Arquitetura e fluxos de dados do sistema M-FOG.

Método de Avaliação

Métricas e Ambientes

- Métricas de qualidade de classificação:
  - Avaliação do fluxo de saída do classificador;
  - Uso de uma matriz de confusão ou erro;
  - Taxa de desconhecidos;
  - Macro F-score;

$$\mathrm{E}_n = \begin{pmatrix} e_{1,1} & e_{1,2} & \cdots & e_{1,J} \\ e_{2,1} & e_{2,2} & \cdots & e_{2,J} \\ \vdots & \vdots & \ddots & \vdots \\ e_{M,1} & e_{M,2} & \cdots & e_{M,J} \end{pmatrix}$$

$$UnkR_n = \frac{1}{M} \sum_{i=1}^{M} \frac{\#Unk_i}{\#ExC_i}$$

$$\mathrm{Fscore1}_n = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Método de Avaliação

Métricas de referência

without considering the *unknown* set. In Eq. 3, Unk_i is the number of examples from the class C_i classified as *unknonw*, and ExC_i is the total number of examples from the class C_i.

$$CER = \frac{1}{2}\sum_{i=1}^{M}\frac{\#ExC_i}{\#Ex}(FPR_i + FNR_i) \tag{2}$$

$$UnkR = \frac{1}{M}\sum_{i=1}^{M}\frac{\#Unk_i}{\#ExC_i} \tag{3}$$

In order to verify the classifier behavior over time, the evaluation methodology proposed in (Faria et al. 2013b) builds a 2D-graphic, where the axis X represents

Método de Avaliação

Métricas e Ambientes

- Métricas de escalabilidade:
  - Número e tipo de processadores;
  - Uso de memória;
  - Tempo de processamento;
  - Taxa de eventos;
  - Latência entre a produção e classificação.
- Ambientes de teste:
  - Computador Pessoal (para desenvolvimento);
  - Nuvem UFSCar;
  - Nevoa composta de SBC (*Sigle Board Computer*) ARM 4 núcleos;

Resultados Preliminares

Resultados Preliminares

Primeira Implementação com Python e Apache Kafka

- *Python* é acessível e fornece bibliotecas diversas;
- *Apache Kafka* é um sistema de mensagens distribuído;
  - Interface de programação com cliente produtor e consumidor;
  - Mensagens organizadas em tópicos que são distribuídos em partições;
- A hipótese de que a carga seria distribuída entre os consumidores, uma vez que o consumidor pode selecionar uma partição para leitura;
- Em experimento com um produtor, 8 partições e 8 consumidores, observou-se que um consumidor processava a maior parte das mensagens, poucos consumidores recebiam algumas mensagens e a maioria dos consumidores não recebia mensagem alguma.

Resultados Preliminares

Segunda Implementação com Apache Flink

- Implementação escrita em Scala ou Java;
- Processamento de fluxos *Stateful*;
- Falta de bibliotecas que distribuam algoritmos base como *K-means*;
- Ambiente de execução (*Flink Cluster*) consome mais memória do que disponível no *hardware*;
- Tempo de execução não foi melhor que a implementação original mesmo sem o trecho de detecção de novidades;

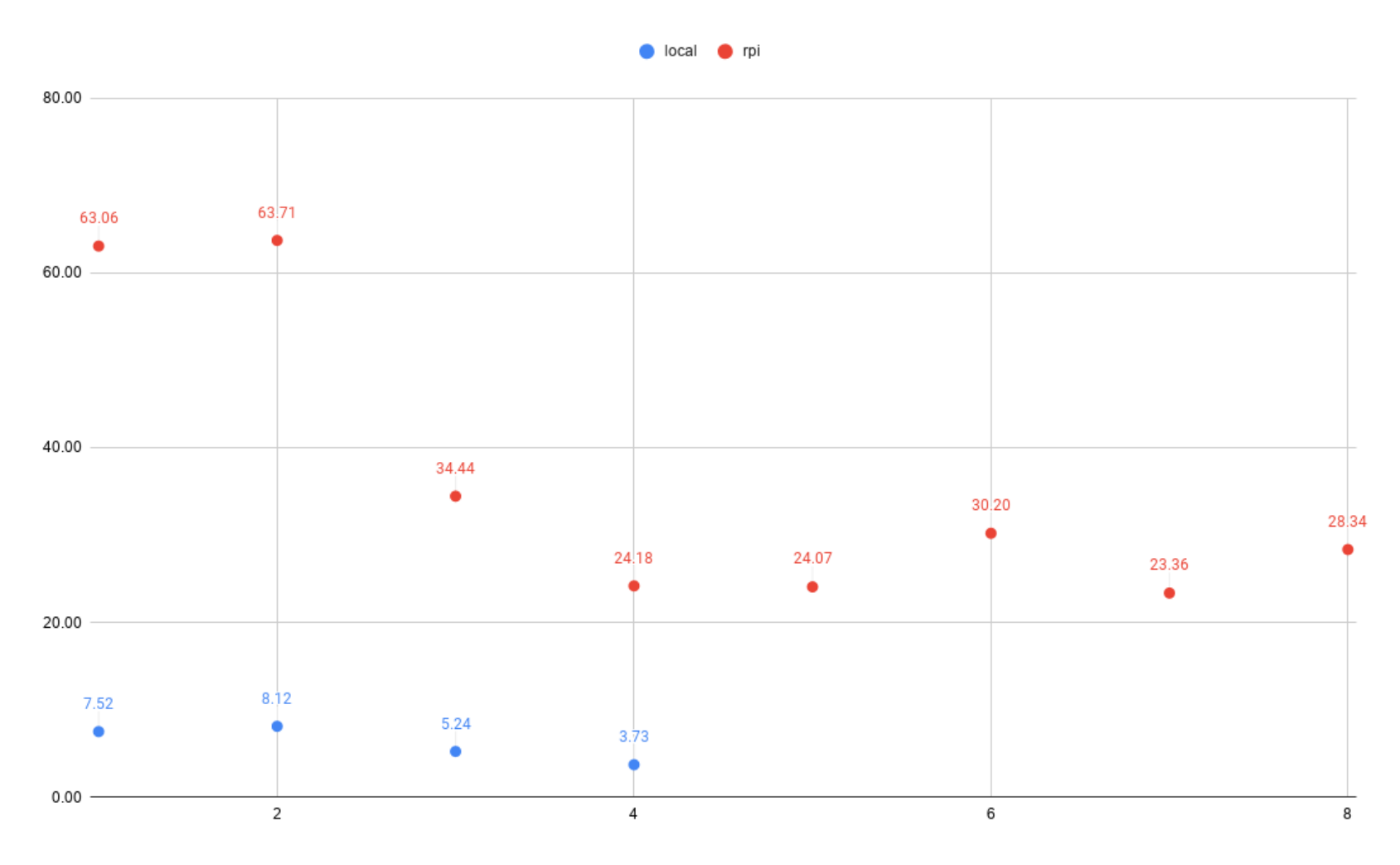
Resultados Preliminares

Terceira Implementação com OpenMPI

- Implementação escrita em C;
- Versão serial e versão paralela e distribuída com *MPI*;
- Reimplementação de algoritmos base como *K-means*;
- Sistema *M-FOG* em desenvolvimento, atualmente na fase de validação através das métricas de qualidade de classificação.
  - Diferença entre os modelos iniciais gerados pelo algoritmo *K-means*;
  - Diferença na matriz de confusão resultante da avaliação dos fluxos de saída;

Resultados Preliminares

Terceira Implementação com OpenMPI



Resultados Preliminares

Desafios Atuais

- Diferença entre `double` e `float`;
- Formato do fluxo de saída;
- Tratamento de exemplos com etiqueta *desconhecido* utilizados para atualização do modelo;
- Diferença entre incluir ou não a borda do cluster;
- Definição de raio;

Notas de Implementação

Algoritmo MINAS

Outras abordagens e implementações

- FuzzyND por Da Silva 2018;
- Minas-LC e Minas-BR por Costa 2019;
- Implementação em Java por Douglas (douglas.m.cavalcanti@gmail.com) em Jul 2 09:37:42 2019
- Implementação em Python por Vitor Sexto Bernardes (vitorsb@gmail.com) em May 11 23:51:09 2020

Algoritmo MINAS

Parâmetros

Table 3 Different configurations of MINAS

Parameters	Setting 1	Setting 2	Setting 3	Setting 4
Number of examples to execute a ND procedure ($\#ExND$)	2000	$\#ExClu * K$	2000	50
Minimum number of examples in the cluster ($\#ExClu$)	$\#ExMem/K$	3	$\#ExMem/K$	30
Window size to forget outdated data	2*$\#ExND$	1000–10,000	2*$\#ExND$	2*$\#ExND$
Threshold	$TV1$ or $TV2$ or $TV3$	$TV1$	$TV1$	$TV1$
Clustering algorithm	Clustream	CluStream + KMeans	Clustream	Clustream
	k = 100	k = 100	k = 100	k = 50
Update micro-cluster	No	No	Yes	No

$\#ExMem$ is the current number of examples in the short-term memory

Implementação referência

Parâmetros

```
class br.ufu.noveltydetection.minas.MinasOg with
        filenameOffline = datasets/training.csv
        filenameOnline = datasets/test.csv

        outputDirectory = out/minas-og//2020-07-20T12-18-21.758/
        algClusteringOff = kmeans
        algClusteringOnl = kmeans

        threshold = 2.0
        flagEvaluationType = 1
        thresholdForgettingPast = 10000
        numMicro = 100
        flagMicroClusters = true

        minExCluster = 20
        validationCriterion = dec

        skipNd = false
```

Nova Implementação

Parâmetros

```
params->kParam = 100;
params->dimension = 22;
params->noveltyThreshold = 2;
params->minExCluster = 20;
params->maxUnkSize = params->kParam * params->minExCluster;
params->thresholdForgettingPast = 10000;
```

Algoritmo MINAS

Fase offline

Algorithm 1 MINAS: Algorithm for the initial training phase

Require: k: number of micro-clusters, alg: clustering algorithm, S: Training Set

```
Model ← ∅
for all (class C_i in S) do
    ModelTmp ← Clustering(S_{Class=C_i}, k, alg)
    for all (micro-cluster micro in ModelTmp) do
        micro.label ← C_i;
    end for
    Model ← Model ∪ ModelTmp;
end for
return Model
```

Algoritmo MINAS

Final da fase offline, inicio da fase online

[. . .] N number of examples, LS linear sum of the examples, SS squared sum of the elements and T timestamp of the arrival of the last example classified in this micro-cluster.

[. . .] After the execution of the clustering algorithm, each micro-cluster is represented by four components (N, LS, SS and T).

[. . .] MINAS uses these measures to classify new examples. For such, it computes the distance between a new example and the closest centroid. If this distance is less than the micro-cluster radius, the example is classified

[. . .] MINAS calculates the radius of a micro-cluster as the standard deviation of the distance between the examples and the centroid, multiplied by a factor f.

Implementação referência

Fase Offline, definição de raio

```
//execution of the KMeans
//variable to store the results of the Kmeans
double soma[]=new double[1];
Clustering centers;
ArrayList<Integer> clusterSize = new ArrayList<~>();
int elemCluster [] = new int [elemList.size()];
ArrayList<Double> radius = new ArrayList<~>();
ArrayList<Double> meanDistance = new ArrayList<~>();

KMeansMOAModified cm = new KMeansMOAModified();
centers = cm.kMeans2(initialCenters,elemList, soma,clusterSize, elemCluster,radius,meanDistance);

for (int g=0; g< elemCluster.length;g++){
    clusters[g] = elemCluster[g];
}
for (int w = 0; w < k; w++) {
    Cluster clus = new Cluster(meanDistance.get(w),centers.get(w).getCenter(),clusterSize.get(w),radius.get(w),cl, cat: "normal",timestamp);
    clusterSet.add(clus);
}
return clusterSet;
```

Detalhe:

```
KMeansMOAModified cm = new KMeansMOAModified();
centers = cm.kMeans2(initialCenters,elemList, soma,clusterSize, elemCluster,radius,meanDistance);

for (int g=0;
    clusters
}
for (int w =
    Cluster
    clusterS
}
```

```
NoveltyDetection.KMeansMOAModified
public Clustering kMeans2(@NotNull Cluster[] centers,
                          List<? extends Cluster> data,
                          double[] soma,
                          ArrayList<Integer> clusterSize,
                          int[] elemCluster,
                          ArrayList<Double> maxDistance,
                          ArrayList<Double> meanDistance)
```

```
er(),clusterSize.get(w),radius.get(w)
```

Implementação referência

Fase Offline, definição de raio

```
public Clustering kMeans2(Cluster[] centers, List<? extends Cluster> data,double soma[], A
    int k = centers.length;

    int dimensions = centers[0].getCenter().length;

    ArrayList<ArrayList<Cluster>> clustering = new ArrayList<~>();
    for ( int i = 0; i < k; i++ ) {
    —— clustering.add( new ArrayList<Cluster>() );
    }

    int repetitions = 100;
    double sum=0;
    do {...}while ( repetitions >= 0 );

    //keeping a vector that contains in each position, the number of elements of the clust
    for (int i=0;i<clustering.size(); i++)
        clusterSize.add(clustering.get(i).size());

    getRadiusMeanDistance(clustering, new Clustering(centers), maxDistance, meanDistance);
    soma[0] = sum;
    return new Clustering( centers );
}
```

Implementação referência

```
public void getRadiusMeanDistance(ArrayList<ArrayList<Cluster>> clustering, Clustering centers
    int k = centers.size();
    double [] centro, elem;
    double maiorDist, soma, distancia, mean;

    for (int i=0; i<k; i++){ // para cada macro-cluster
        // clustering eh o resultado do kmeans
        centro = centers.get(i).getCenter(); //obter o centro do macro-cluster
        maiorDist = 0;
        mean = 0.0;
        for (int j=0;j<clustering.get(i).size(); j++){ //para cada elemento no macro-cluster
            elem = clustering.get(i).get(j).getCenter();
            soma=0;
            for (int l=0; l< elem.length; l++){ //distancia do elem ao centro
                soma = soma + Math.pow(centro[l] - elem[l], 2);
            }
            distancia = Math.sqrt(soma);
            mean = mean+distancia;
            if (distancia > maiorDist) // procurando pelo elem cuja distancia e a maior
                maiorDist = distancia;
        }
        maxDistance.add(maiorDist);
        meanDistance.add(mean/clustering.get(i).size());
    }
}
```

Algoritmo MINAS

Fase online

Algorithm 2 MINAS: Algorithm for the online phase

Require: $Model$: decision model created in the initial training phase, DS: data stream, T: threshold, $NumExamples$: minimal number of examples to execute a ND procedure, $windowsize$: size of a data window, alg: clustering algorithm.

$ShortMem \leftarrow \emptyset$
$SleepMem \leftarrow \emptyset$
for all (example ex in DS) **do**
  $(Dist, micro) \leftarrow closer\text{-}micro(ex, Model)$
  if ($Dist \leq radius(micro)$ **then**
    $ex.class \leftarrow micro.label$
    update-micro($micro, ex$)
  else
    $ex.class \leftarrow unknown$
    $ShortMem \leftarrow ShortMem \cup ex$
    if ($|ShortMem| \geq NumExamples$) **then**
      $Model \leftarrow novelty\text{-}detection(Model, ShortMem, SleepMem, T, alg)$
    end if
  end if
  $CurrentTime \leftarrow ex.time$
  if ($CurrentTime$ mod $windowSize == 0$) **then**
    $Model \leftarrow move\text{-}sleepMem(Model, SleepMem, CurrentTime, windowSize)$
    $ShortMem \leftarrow remove\text{-}oldExamples(ShortMem, windowsize)$
  end if
end for

Implementação referência

Fase online

```
public String[] identifyExample(double [] dataEx) {
    double dist = 0.0;
    double minDist = Double.MAX_VALUE;
    int pos = 0;
    String[] vectorResults = new String[2];

    for(int j = 0; j < model.size(); j++){
        dist = KMeansMOAModified.distance(model.get(j).getCenter(), dataEx);
        if (dist <= minDist){
            minDist = dist;
            pos = j;
        }
    }
    try{
        if (minDist < model.get(pos).getRadius()){
            vectorResults[0] = model.get(pos).getLblClasse();
            vectorResults[1] = model.get(pos).getCategory();
            model.get(pos).setTime(timestamp);
            return(vectorResults);
        }
    }
```

Nova Implementação

Fase online

```
void classify(int dimension, Model *model, Point *ex, Match *match) {
    // assumes dataset is normalized in [0, 1]
    match->distance = (double) dimension;
    match->pointId = ex->id;
    match->label = '-';
    for (int i = 0; i < model->size; i++) {
        double distance = MNS_distance(ex->value, model->vals[i].center, dimension);
        if (distance <= match->distance) {
            match->clusterId = model->vals[i].id;
            match->clusterLabel = model->vals[i].label;
            match->clusterCatergoy = model->vals[i].category;
            match->clusterRadius = model->vals[i].radius;
            match->secondDistance = match->distance;
            match->distance = distance;
        } else if (distance <= match->secondDistance) {
            match->secondDistance = distance;
        }
    }       You, 9 days ago • More revision on ND and evaluation
    // If the border isn't included, a novelty cluster would miss the farthest
    // example that was used to create the cluster in the first place.
    // if (match->distance < match->clusterRadius) {
    if (match->distance <= match->clusterRadius) {
        match->label = match->clusterLabel;
    }
```

Algoritmo MINAS

Fase online, Detecção de novidades

Algorithm 3 MINAS: Algorithm for detection of NPs or extensions

Require: $Model$: current decision model, $ShortMem$: short-term memory, $SleepMem$: sleep memory, T: threshold, alg: clustering algorithm

```
ModelTmp ← Clustering(ShortMem, k, alg)
for all (micro-grupo micro in ModelTemp) do
  if ValidationCriterion(micro) then
    (Dist, microM) ← closest-micro(micro,Model)
    if Dist ≤ T then
      micro.label ← microM.label
    else
      (Dist, microS) ← closest-micro(micro,SleepMem)
      if Dist ≤ T then
        micro.label ← microS.label
      else
        micro.label ← new label
      end if
    end if
    Model ← Model ∪ micro
  end if
end for
return Model
```

Implementação referência

Output Stream

```
57099    Ex: 57099    Classe Real: A  Classe MINAS: Unk
57100    Ex: 57100    Classe Real: A  Classe MINAS: C N
57101    Ex: 57101    Classe Real: A  Classe MINAS: Unk
57102    Ex: 57102    Classe Real: A  Classe MINAS: Unk
57103    Ex: 57103    Classe Real: A  Classe MINAS: Unk
57104    Ex: 57104    Classe Real: A  Classe MINAS: C N
57105    Ex: 57105    Classe Real: A  Classe MINAS: Unk
57106    Ex: 57106    Classe Real: A  Classe MINAS: Unk
57107    Ex: 57107    Classe Real: A  Classe MINAS: C N
57108    Ex: 57108    Classe Real: A  Classe MINAS: Unk
57109    Ex: 57109    Classe Real: A  Classe MINAS: Unk
57110    ThinkingNov: Novidade 1 - 30 examples
57111    Thinking Extension:C N - 23 examples
57112    ThinkingNov: Novidade 2 - 203 examples
57113    Thinking NoveltyExtension: N 2 - 30 examples
57114    Thinking NoveltyExtension: N 2 - 21 examples
57115    ThinkingNov: Novidade 3 - 26 examples
57116    Thinking Extension:C N - 324 examples
57117    Thinking Extension:C N - 864 examples
57118    Thinking NoveltyExtension: N 3 - 38 examples
57119    Thinking NoveltyExtension: N 3 - 30 examples
57120    ThinkingNov: Novidade 4 - 20 examples
57121    Ex: 57110    Classe Real: A  Classe MINAS: ExtCon N
57122    Ex: 57111    Classe Real: A  Classe MINAS: ExtCon N
```

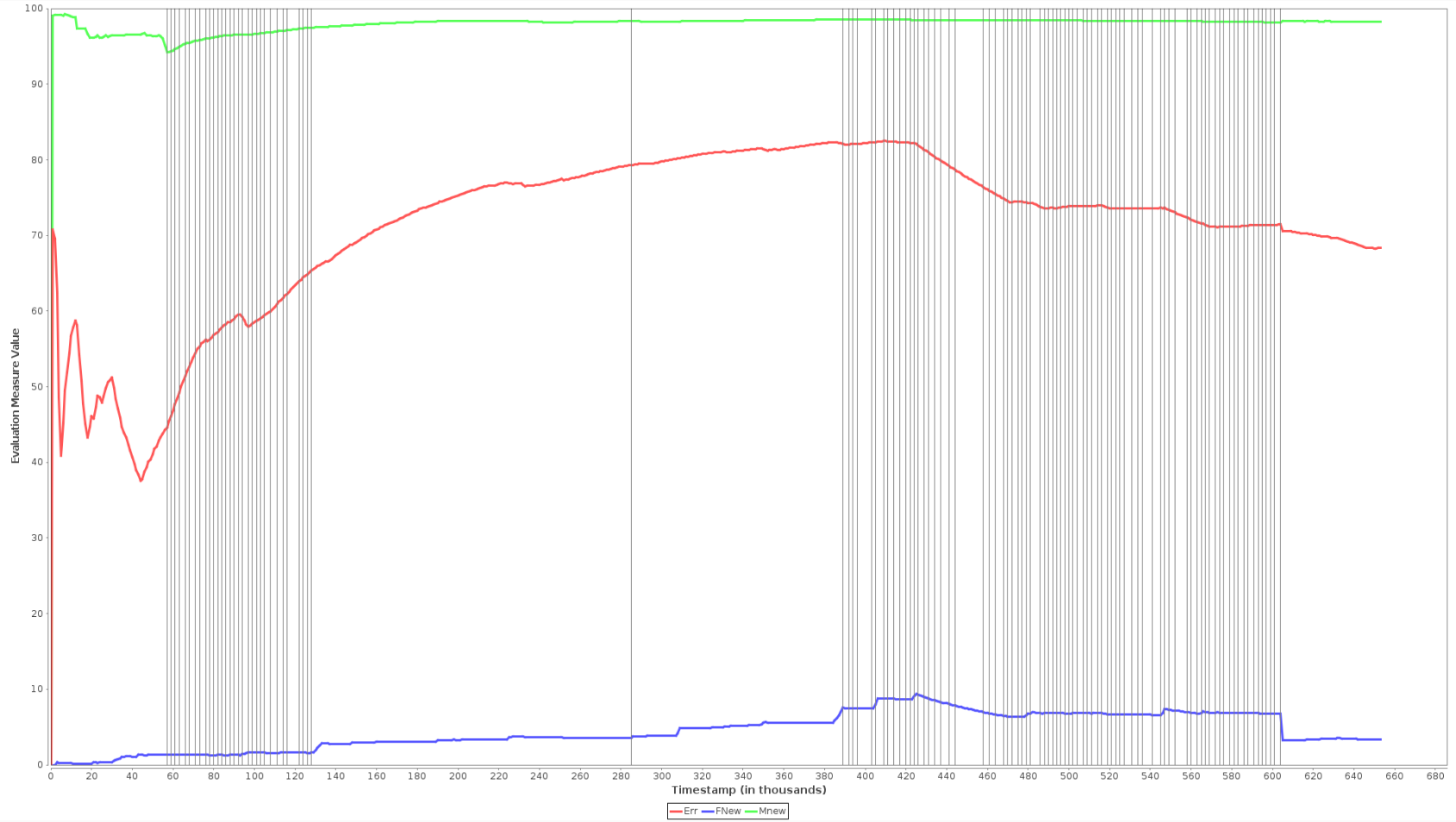
Nova Implementação

Output Stream

```
#pointId,clusterLabel,clusterCategory,clusterId,clusterRadius,label,distance,secondDistance
0,N,n,0,1.736760e-01,-,1.414214e+00,4.472136e+00
1,N,n,0,1.736760e-01,-,1.756069e+00,4.263076e+00
2,N,n,0,1.736760e-01,-,1.414214e+00,4.472136e+00
3,N,n,0,1.736760e-01,-,1.414214e+00,4.472136e+00
4,N,n,0,1.736760e-01,-,1.732051e+00,4.358899e+00
5,N,n,0,1.736760e-01,-,1.732051e+00,4.358899e+00
6,N,n,0,1.736760e-01,-,1.732051e+00,4.358899e+00
7,N,n,0,1.736760e-01,-,1.414673e+00,4.461084e+00
8,N,n,0,1.736760e-01,-,1.734020e+00,4.333510e+00
9,N,n,0,1.736760e-01,-,1.733688e+00,4.333678e+00
10,N,n,0,1.736760e-01,-,1.734346e+00,4.328142e+00
11,N,n,0,1.736760e-01,-,1.734179e+00,4.328383e+00
12,N,n,0,1.736760e-01,-,1.732590e+00,4.342143e+00
13,N,n,0,1.736760e-01,-,1.732051e+00,4.358899e+00
14,N,n,0,1.736760e-01,-,1.732051e+00,4.358899e+00
15,N,n,0,1.736760e-01,-,1.732051e+00,4.358899e+00
16,N,n,0,1.736760e-01,-,1.733914e+00,4.334378e+00
17,N,n,0,1.736760e-01,-,1.600781e+00,4.366062e+00
18,N,n,0,1.736760e-01,-,1.755992e+00,4.263700e+00
19,N,n,0,1.736760e-01,-,2.364014e+00,3.901834e+00
20,N,n,0,1.736760e-01,-,1.414214e+00,4.472136e+00
21,N,n,0,1.736760e-01,-,1.734036e+00,4.331682e+00
22,N,n,0,1.736760e-01,-,1.734261e+00,4.328838e+00
```

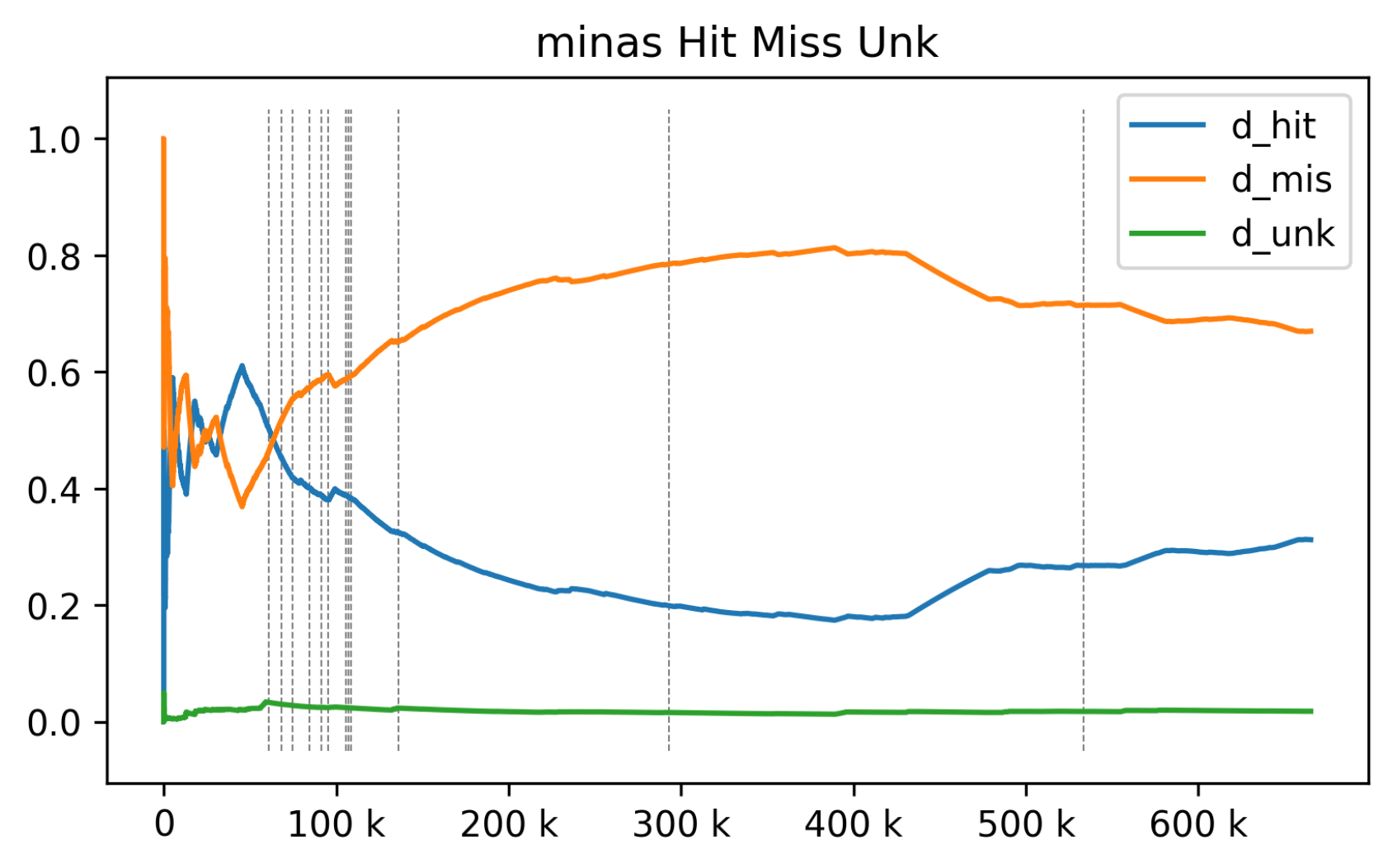
Implementação referência

Output Stream, Evaluation



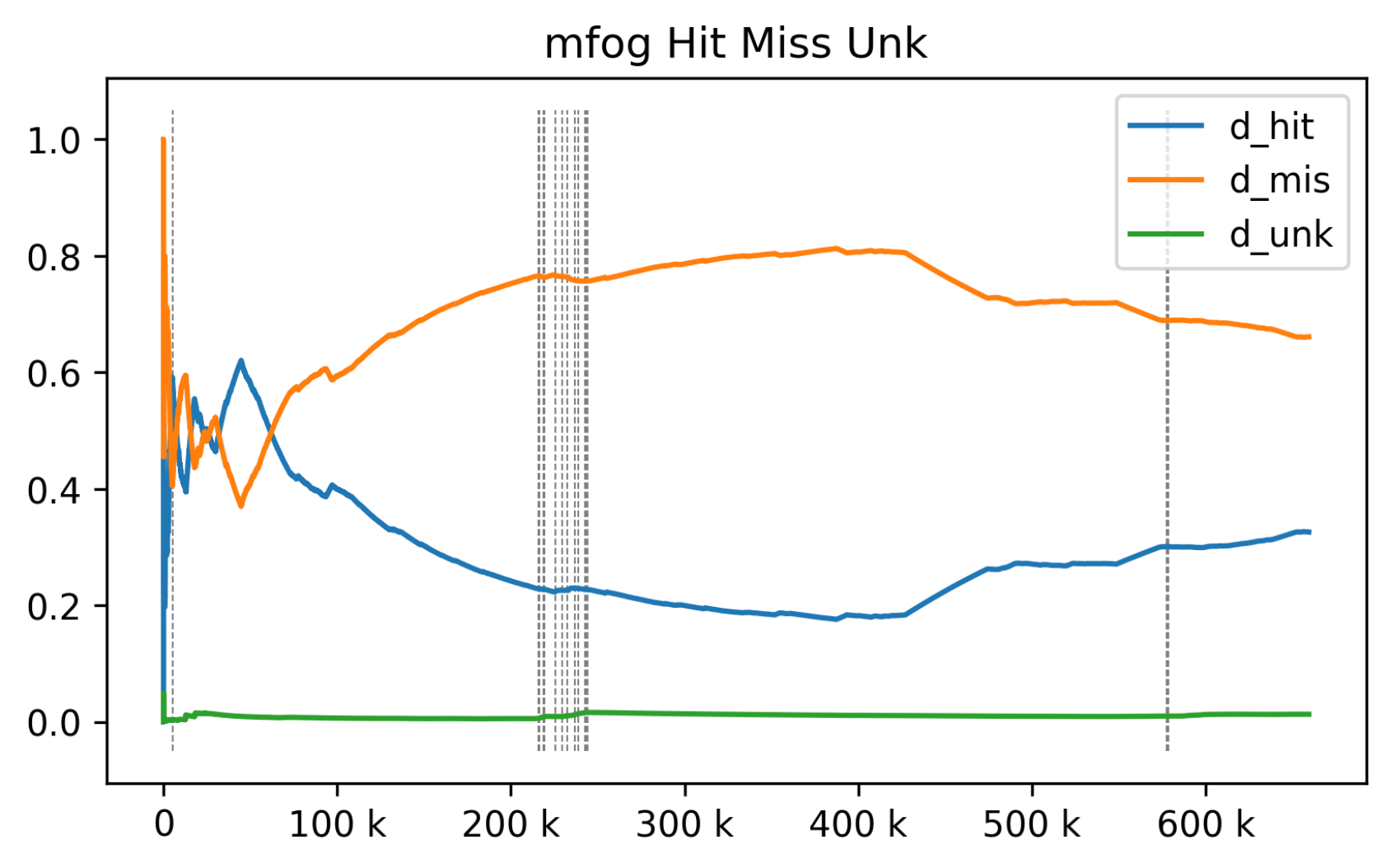
Implementação referência

Output Stream, Evaluation



Nova Implementação

Output Stream, Evaluation



Implementação referência

Matrix de confusão e avaliação

```
### Minas

Confusion Matrix
Classes (act)       A       N assigned    hits  misses
Labels (pred)
-                3774    8206        -       0       0
1                 123       0        A     123     123
10               3520    5130        N    5130    5130
11                 71     289        N     289     289
12                 26       0        A      26      26
2                 152      82        A     152     152
3                 368      44        A     368     368
4                   8       0        A       8       8
5                  82       1        A      82      82
6                 165       0        A     165     165
7                   8     396        N     396     396
8                1054     183        A    1054    1054
9                 161     154        A     161     161
N              441395  199715        N  199715  199715
Classes           ['A' 'N']
Initial labels    ['-', 'N']
Labels            ['-', '1', '10', '11', '12', '2', '3', '4', '5', '6', '7', '8', '9', 'N'] 14
Total examples    (653457, 25)
Total matches     (665107, 9)
Hits               207669 ( 31.223397%)
Misses             445458 ( 66.975389%)
Unknowns            11980 (  1.801214%)
Unk. reprocessed    11650 ( 97.245409%)
Total              665107 (100.000000%)
```

Nova Implementação

Matrix de confusão e avaliação

```
### Mfog

Confusion Matrix
Classes (act)       A       N assigned    hits  misses
Labels (pred)
-                8306     355        -       0       0
N              435763  205887        N  205887  205887
a                 236       0        A     236     236
b                 756      81        A     756     756
c                 574       0        A     574     574
d                 214       0        A     214     214
e                 243       0        A     243     243
f                 499       0        A     499     499
g                 431       0        A     431     431
h                 495       0        A     495     495
i                 237       0        A     237     237
j                 640       0        A     640     640
k                  57       0        A      57      57
l                 539       0        A     539     539
m                  55       0        A      55      55
n                  91       0        A      91      91
o                  69       0        A      69      69
p                3464       0        A    3464    3464
q                 169       0        A     169     169
r                 269       0        A     269     269
Classes           ['A' 'N']
Initial labels    ['-', 'N']
Labels            ['-', 'N', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r'] 20
Total examples    (653457, 25)
Total matches     (659430, 9)
Hits               214925 ( 32.592542%)
Misses             435844 ( 66.094051%)
Unknowns             8661 (  1.313407%)
Unk. reprocessed     5973 ( 68.964323%)
Total              659430 (100.000000%)
```

Notas de Implementação

Algoritmo MINAS vs Implementação referência

- Definição de raio: desvio padrão das distâncias versus distancia máxima;
- Atualização do micro-cluster limita-se à atualização do atributo `T`;
- Remoção de exemplos na implementação de referência é feita somente para o algoritmo *CluStream*;
- Inclusão de borda: algoritmo inclui ($<=$), referência não inclui ($<$);

Notas de Implementação

Algoritmo MINAS vs Nova Implementação

- Seguiu-se as mesmas divergências anteriores para comparação dos resultados com a implementação referência;
- Inclusão da borda;
- Comportamento do mecânismo de *sleep-model* não está definido, portanto não está ativo;
- Processo de clusterização é limitado ao algoritmo *K-Means*. Algoritmo *CluStream* não está implementado;