Nursery

Danilo Souza Hugo Santos Iago Medeiros
Welton Araújo

[1]Universidade Federal do Pará

16 de Julho de 2013

O Problema

- Modelo de decisão hierárquica
- Desenvolvido para classificar requerimentos à escolas infantis
- Muito utilizado nos anos 80 na Slovênia
- Os requerimentos rejeitados precisavam de uma explicação objetiva
- A decisão final dependia de 3 principais fatores
  - Ocupação dos pais e berçário da criança
  - Estruturas familiar e financeira
  - Condições sociais e de saúde da família
- O modelo foi desenvolvido baseado no sistema DEX [M. Bohanec, V. Rajkovic: Expert system for decision making. Sistemica 1(1), pp. 145-157, 1990]

Atributos e seus estados

- *parents* (Ocupação dos pais)
  - usual, pretentious, great_pret
- *has_nurs* (Berçário da criança)
  - proper, less_proper, improper, critical, very_crit
- *form* (Estrutura familia)
  - complete, completed, incomplete, foster
- *children* (Número de crianças)
  - 1, 2, 3, more
- *housing* (Condições de moradia)
  - convenient, less_conv, critical
- *finance* (Condições financeiras)
  - convenient, inconv
- *social* (Condições sociais)
  - non-prob, slightly_prob, problematic
- *health* (Condições de saúde)
  - recommended, priority, not_recom

- Programa que possui uma coleção de algoritmos de aprendizagem de máquina para usar em tarefas de mineração de dados
- Desenvolvido pelo Machine Learning Group da Universidade de Waikato
- O Weka permite que se trabalhe sobre o dataset e traz ferramentas para: pré-processamento, classificação, regressão, clusterização, regras de associação e visualização
- Todos os nossos resultados foram feitos com o apoio do Weka

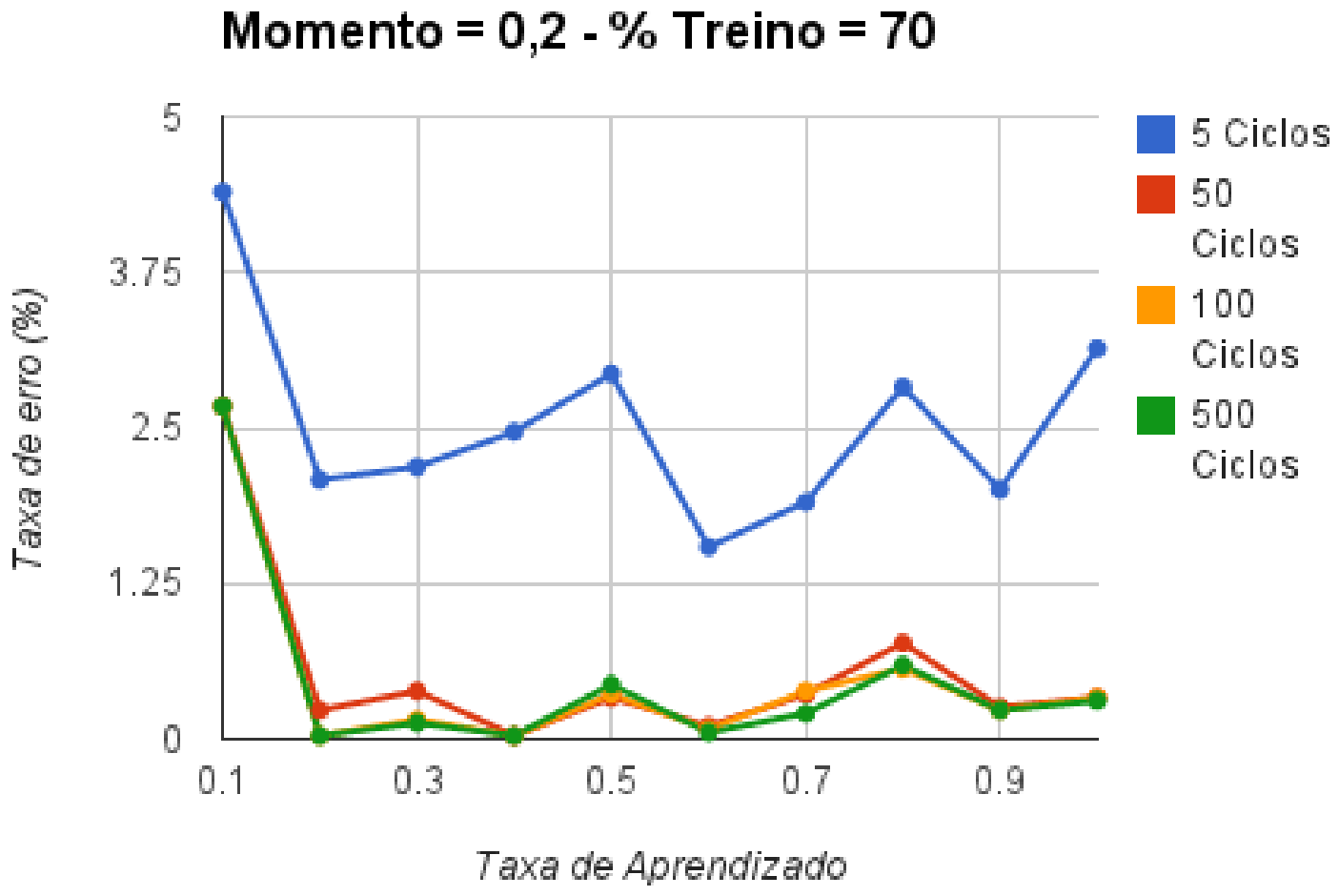
O *momentum*

- Utilizado para acelerar a convergência de rede
- Adiciona uma fração proporcional à alteração enterior no cálculo dos pesos sinapticos
- Aumenta a estabilidade do processo de aprendizagem

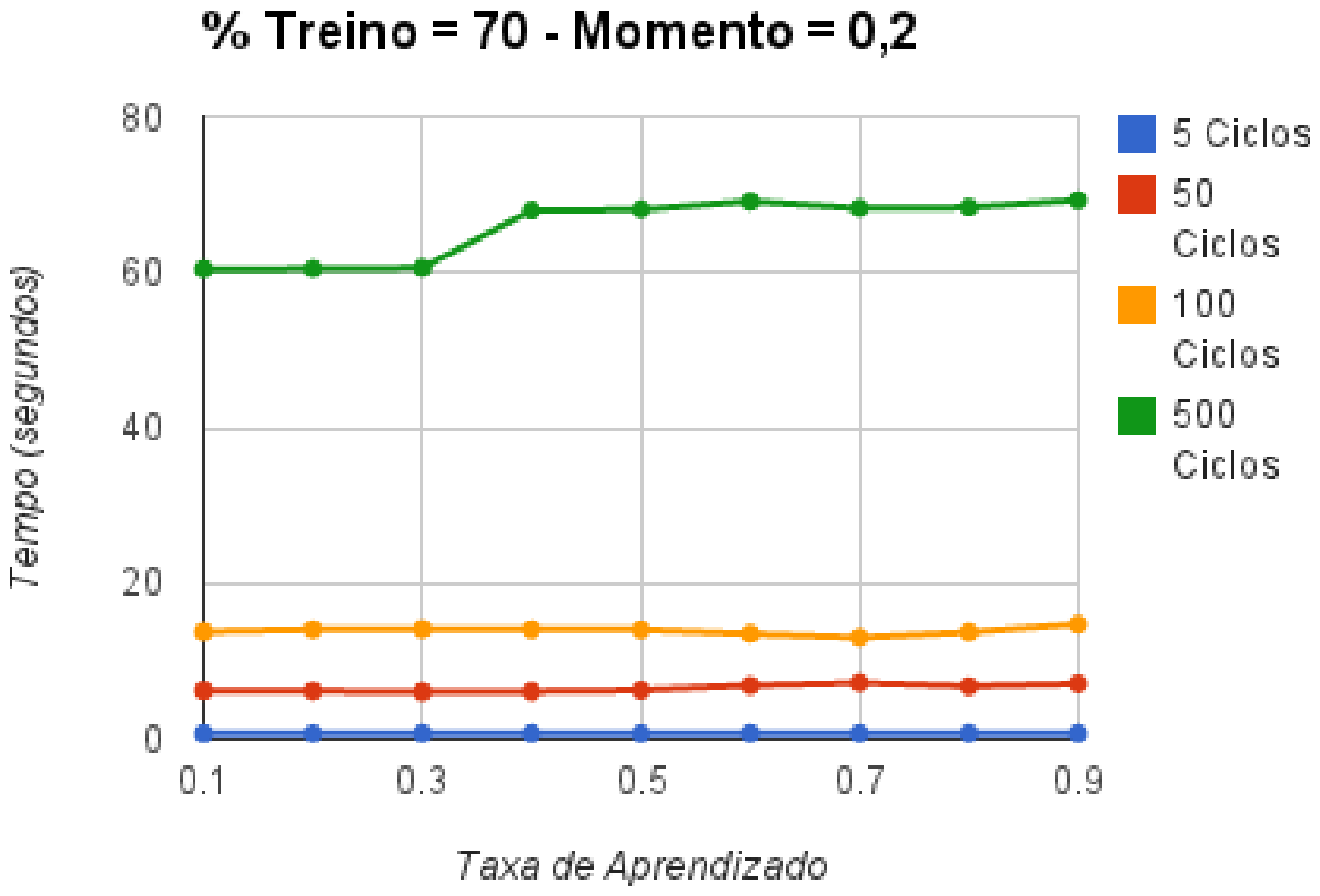
$$w_{ij}(n+1) = w_{ij}(n) + \alpha e_i(n) x_j(n) + \beta[w_{ij}(n) - w_{ij}(n-1)]$$

- Parâmetros fixos
  - Função de ativação: sigmóide
  - Rede de 1 Camada
- Parâmetros variados para simulação
  - Poncentagem da base de dados para treinamento da rede
  - Taxa de aprendizado
  - Coeficiente de inércia (*momentum*)
  - Número de ciclos (iterações) realizados
- Parâmetros analisados nas simulações
  - Taxa de erro de classificação
  - Tempo de construção do modelo

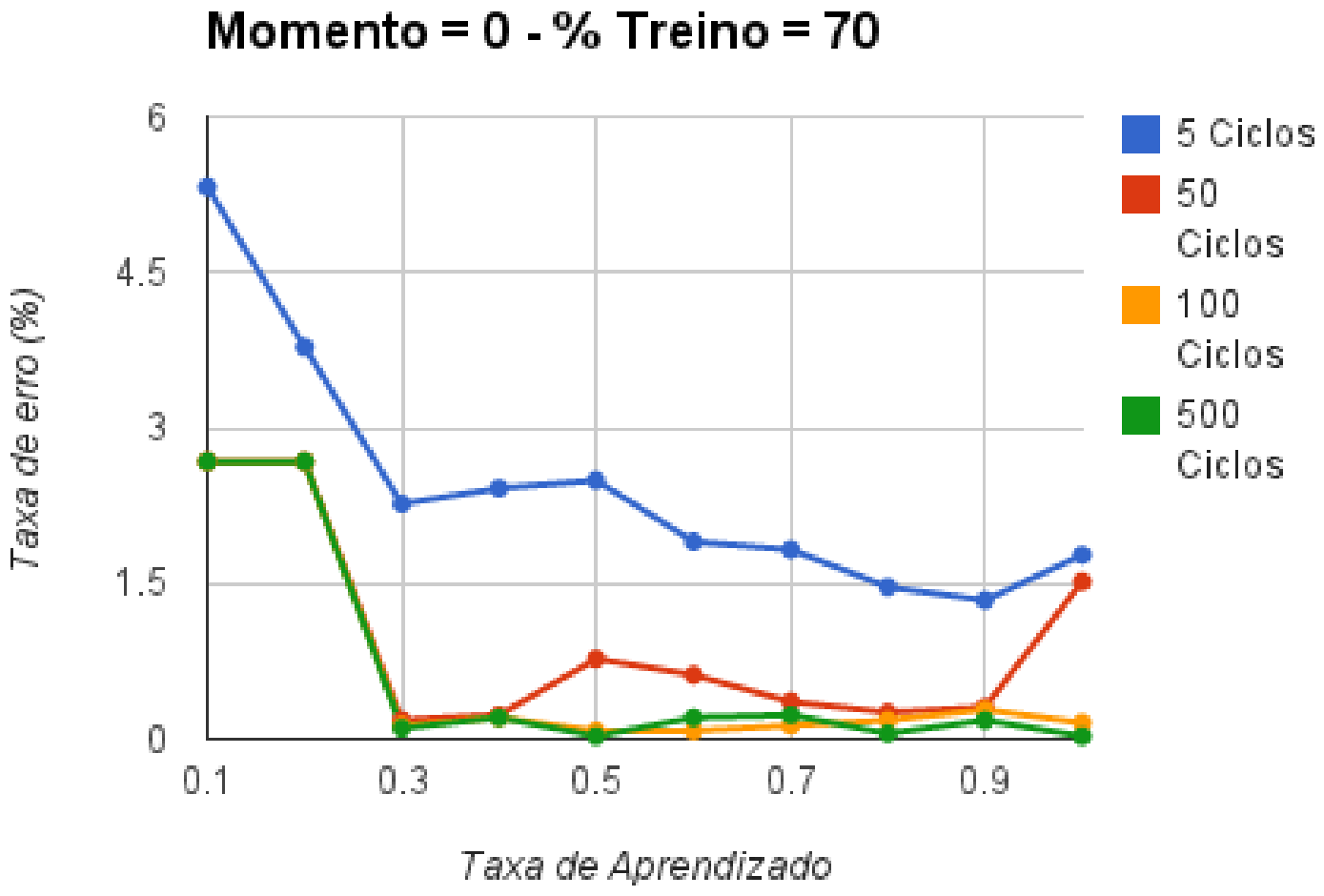
Momento constante



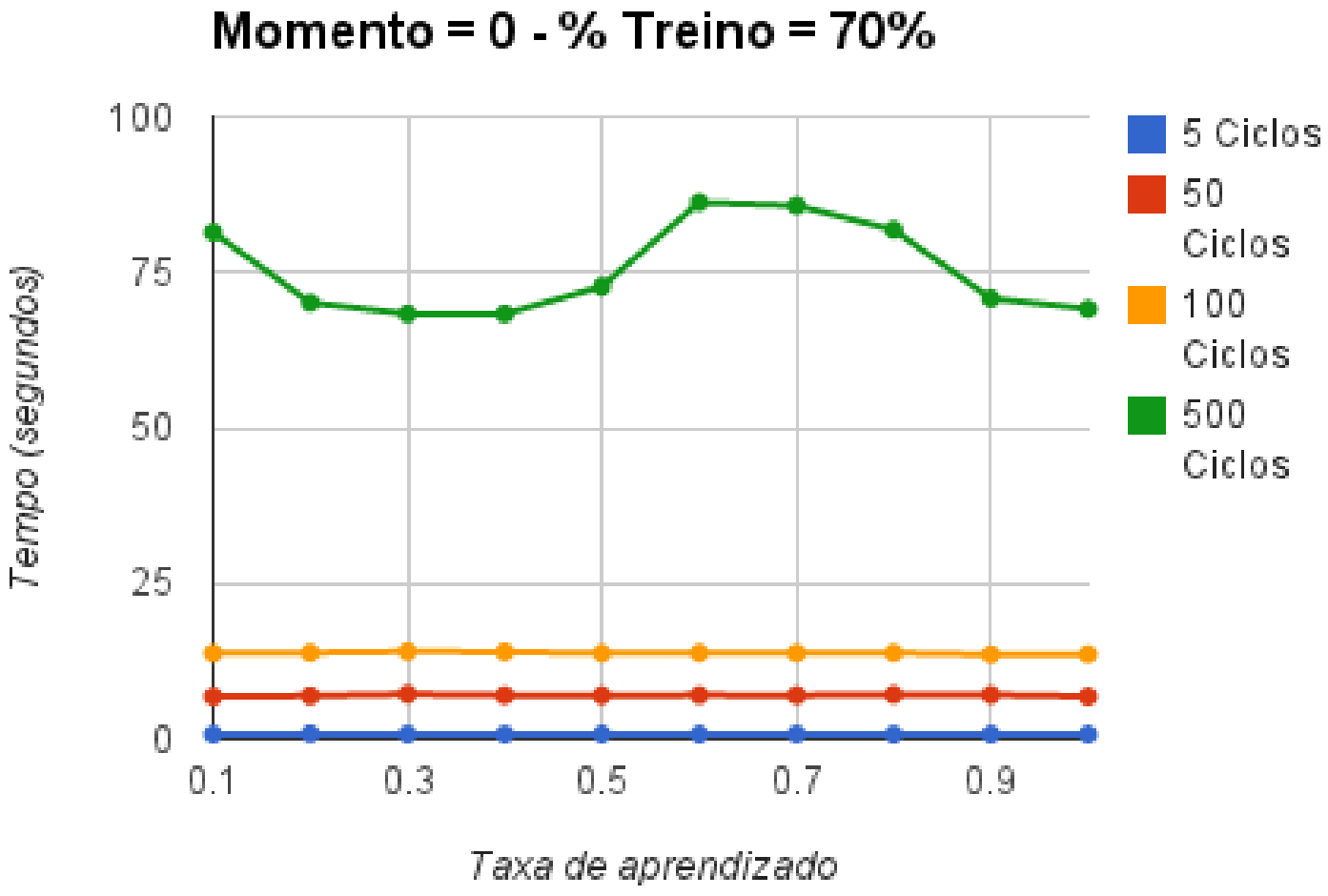
Momento constante



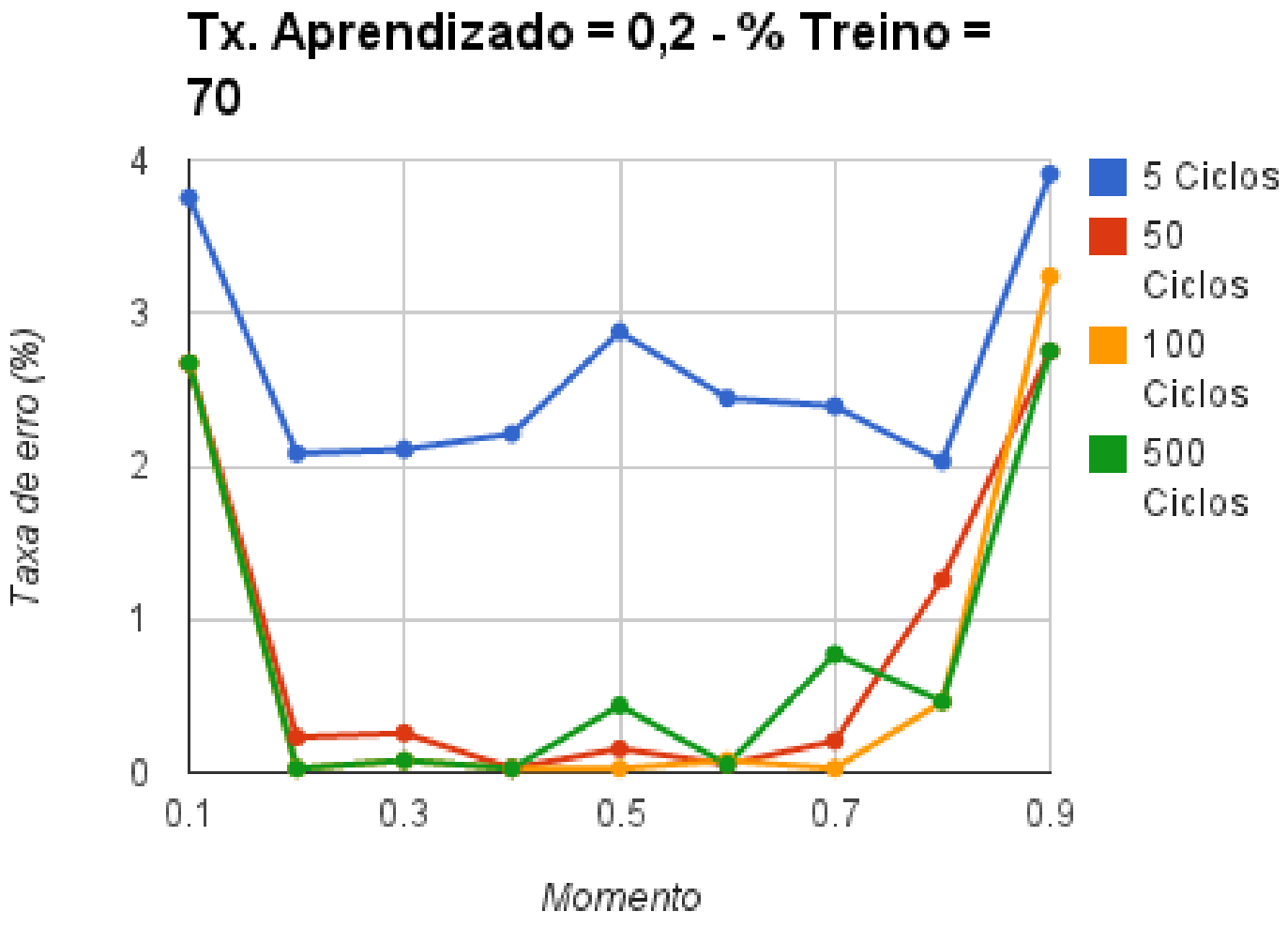
Sem momento


Momento = 0 - % Treino = 70
6
4.5
3
1.5
0
0.1
0.3
0.5
0.7
0.9
Taxa de erro (%)
Taxa de Aprendizado
5 Ciclos
50 Ciclos
100 Ciclos
500 Ciclos

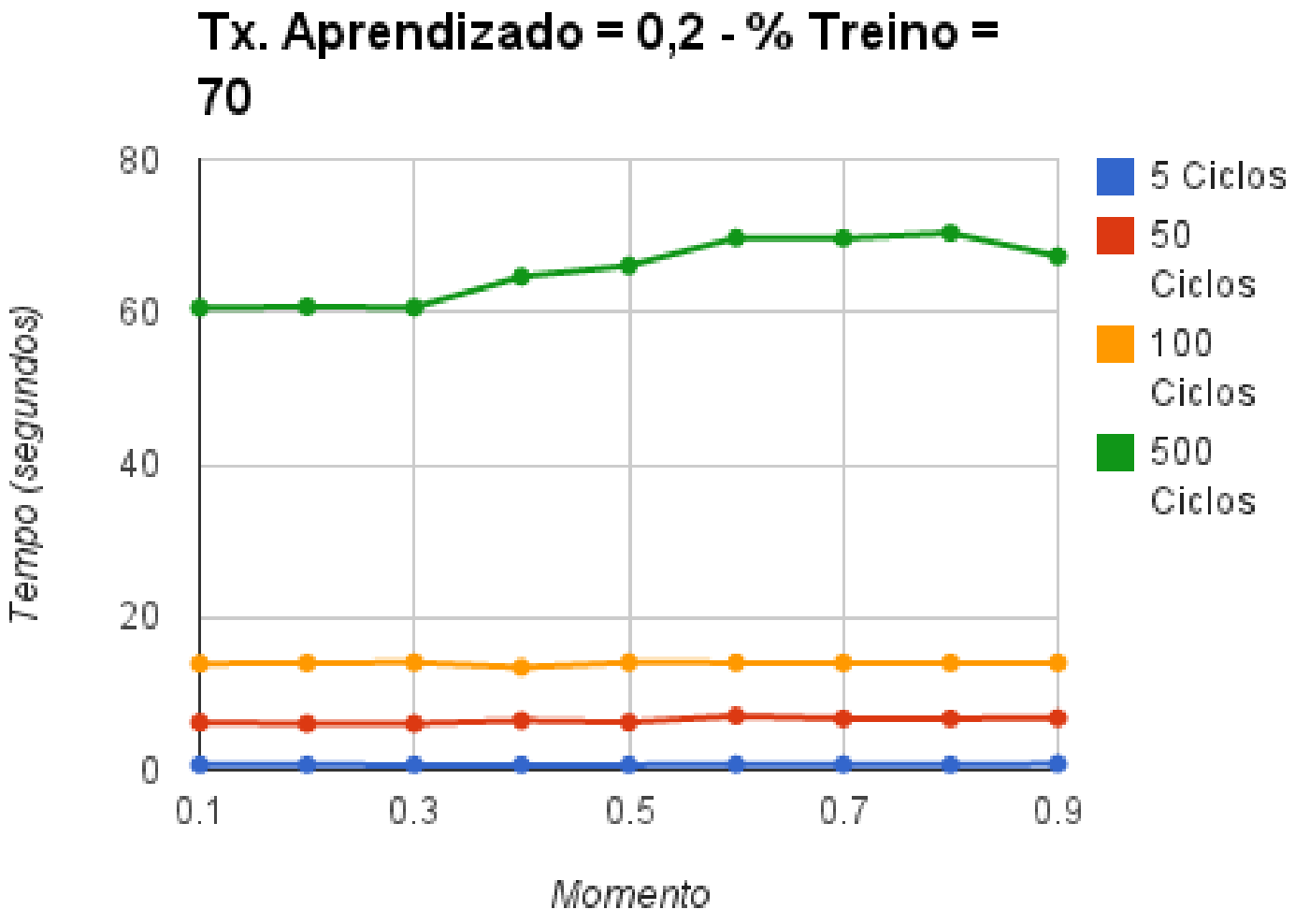
Sem momento


Momento = 0 - % Treino = 70%
100
75
50
25
0
Tempo (segundos)
0.1
0.3
0.5
0.7
0.9
Taxa de aprendizado
5 Ciclos
50 Ciclos
100 Ciclos
500 Ciclos

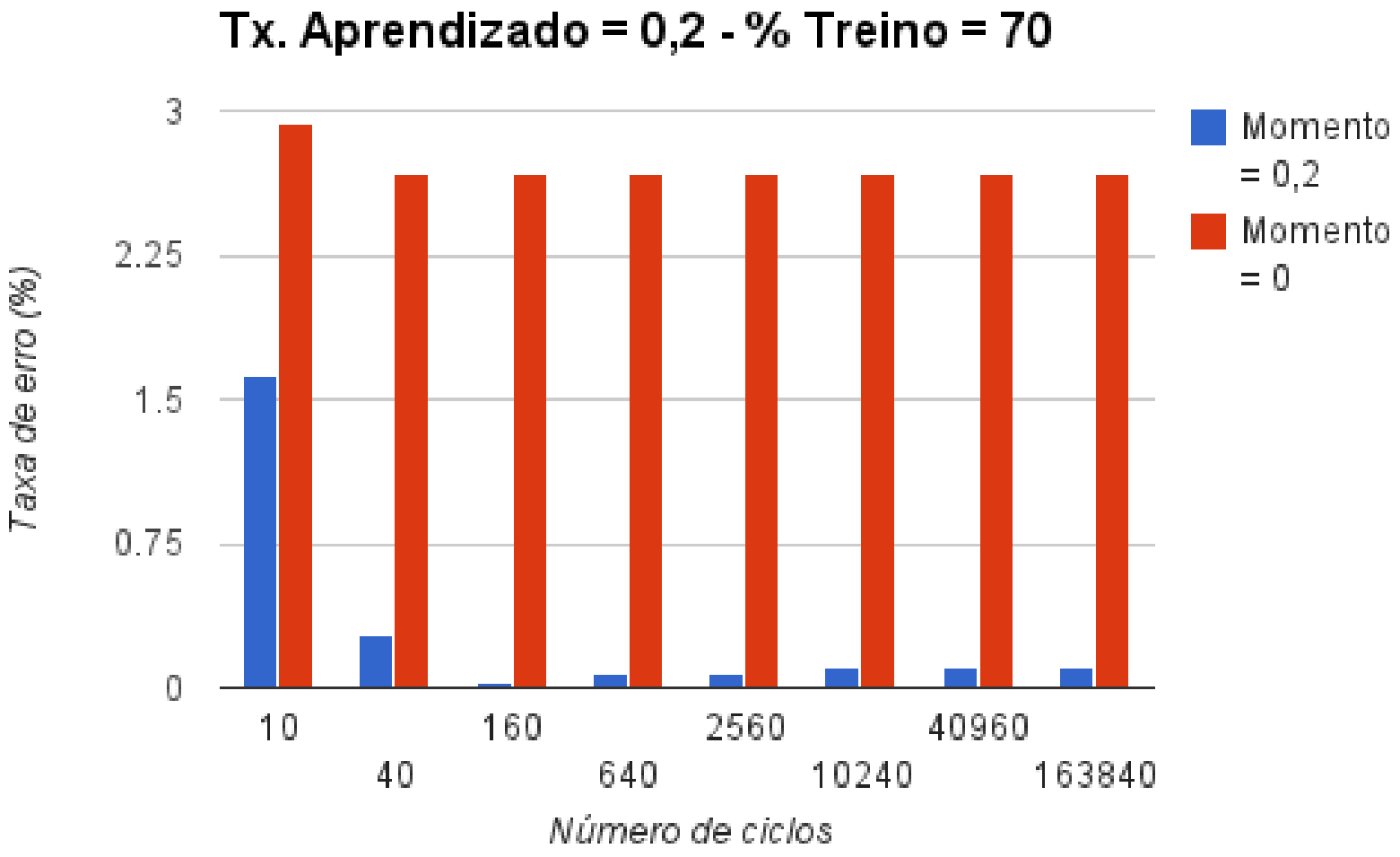
Taxa de aprendizado constante



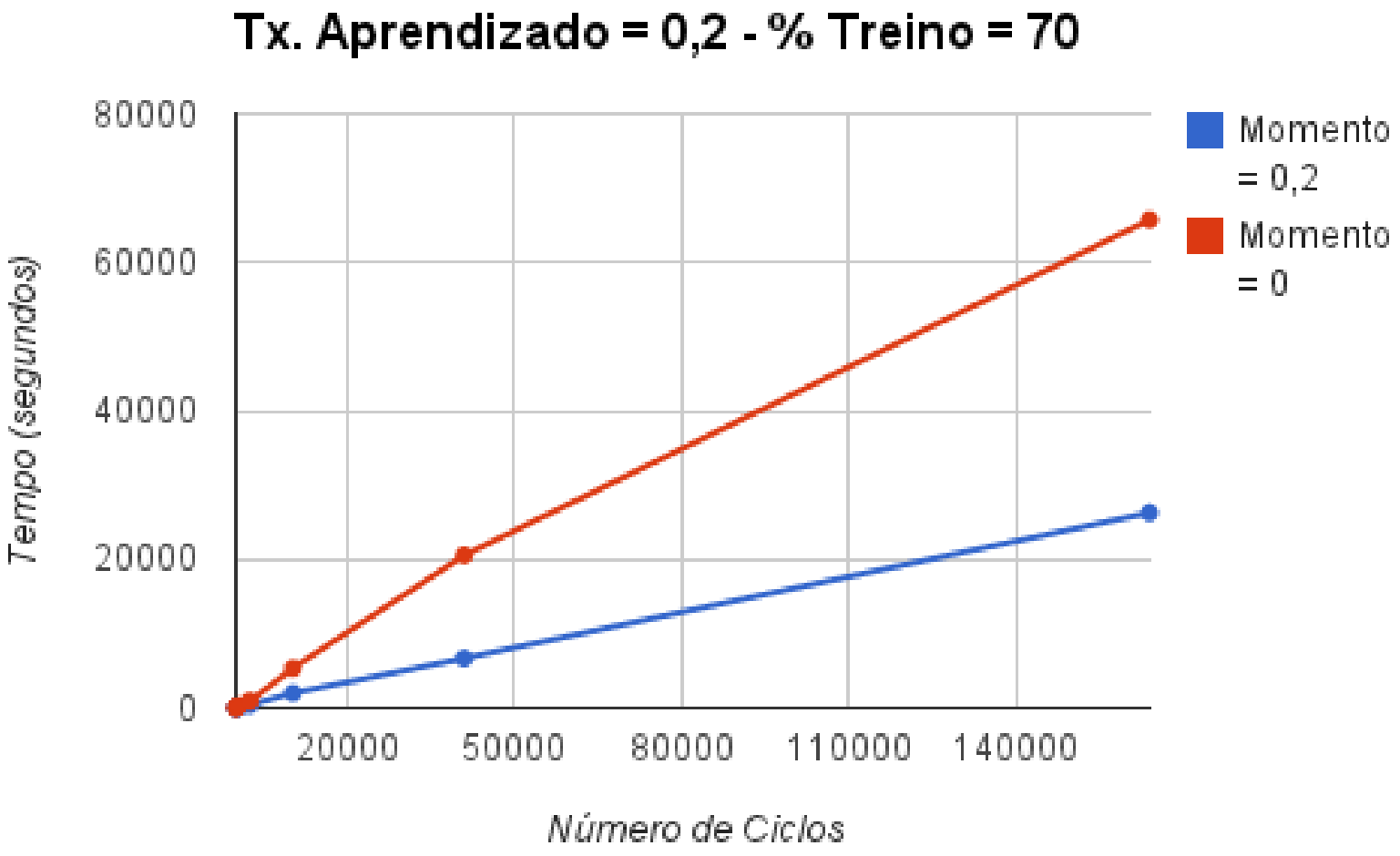
Resultados - Taxa de aprendizado constante



Resultados - Taxa de aprendizado constante e momento = 0 & momento = 0.2



Resultados - Taxa de aprendizado constante e momento = 0 & momento = 0.2



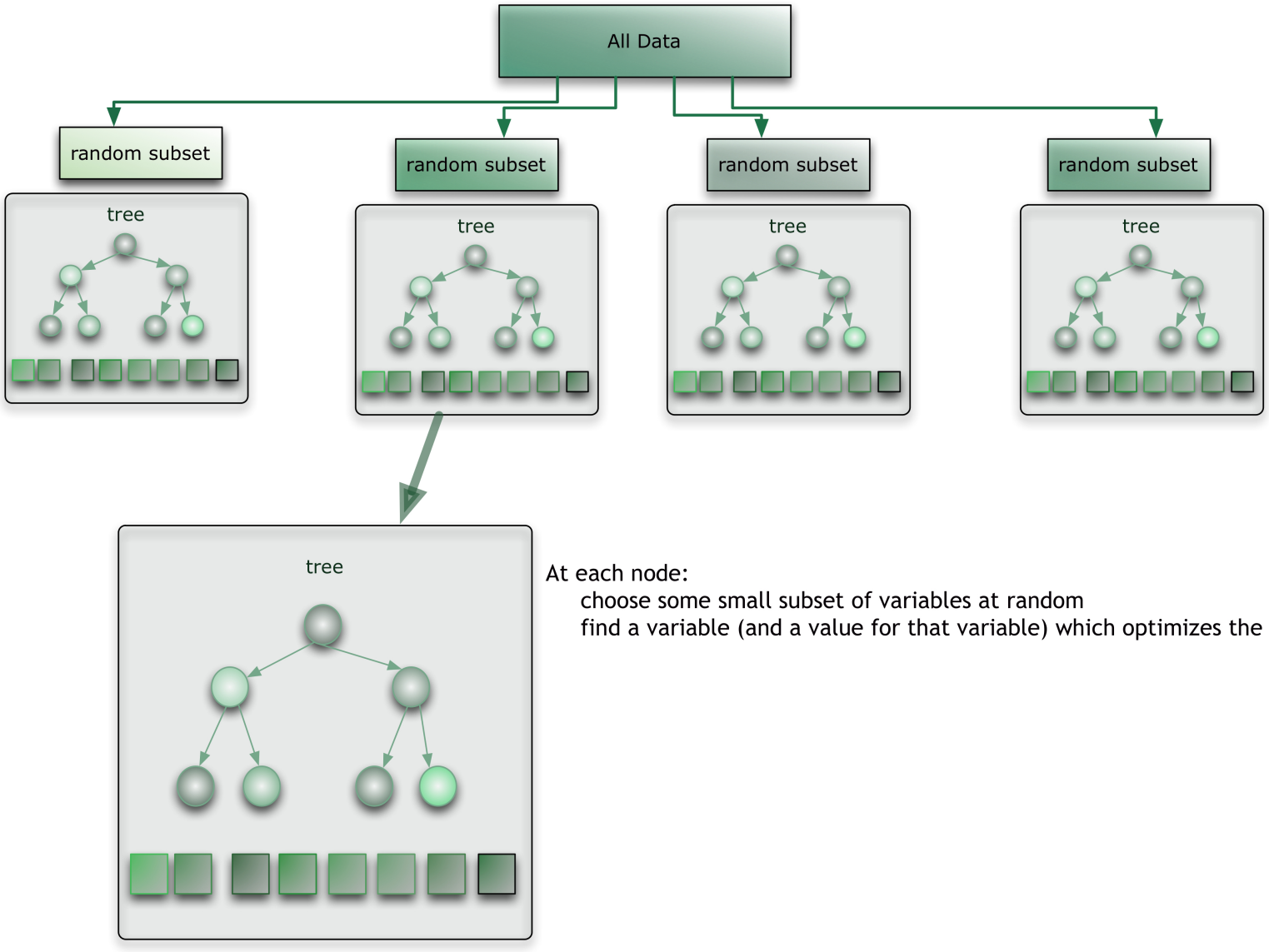
Floresta Randômica (Random Forest)

- Tudo começou em 1996, com o surgimento do meta-algoritmo de Bagging (Bootstrap Aggregating), por Leo Breiman
- Bootstrap ajuda a reduzir variância e overfitting
  - Escolhe aleatoriamente amostras, Di, de um conjunto de treino, D
  - De cada amostra, encontra o classifier.
  - O classifier escolhido será o que mais aparecer, ou seja, tiver maior votos.
  - Esse conjunto de treino D tem reposição (ou seja, uma amostra pode ser escolhida mais de uma vez)
  - Tamanho do conjunto de treino: n
  - Tamanho do conjunto de amostra: n'
  - Geralmente n' < n
  - Quando n' for praticamente n, nota-se uma chance de 63% de aparecer amostras não repetidas. Daqui que surgiu o nome de Bootstrap

Floresta Randômica (Random Forest)

- Em 2001, Breiman lançou o Random Forest
- É um método usado para classificação (e regressão)
- O método faz a construção de várias árvores de decisão, com a diferença de que não usa pruning (poda)
- Algoritmo:
  - Escolhe-se aleatoriamente a amostra (bootstrap) Di dentre D
  - Constrói a árvore Ti usando amostra Di
  - Em cada árvore Ti, escolhe aleatoriamente M variáveis e encontra a melhor divisão (split)
- No fim, pode-se
  - pegar o voto majoritário (classificação)
  - calcular a média dos resultados (regressão)

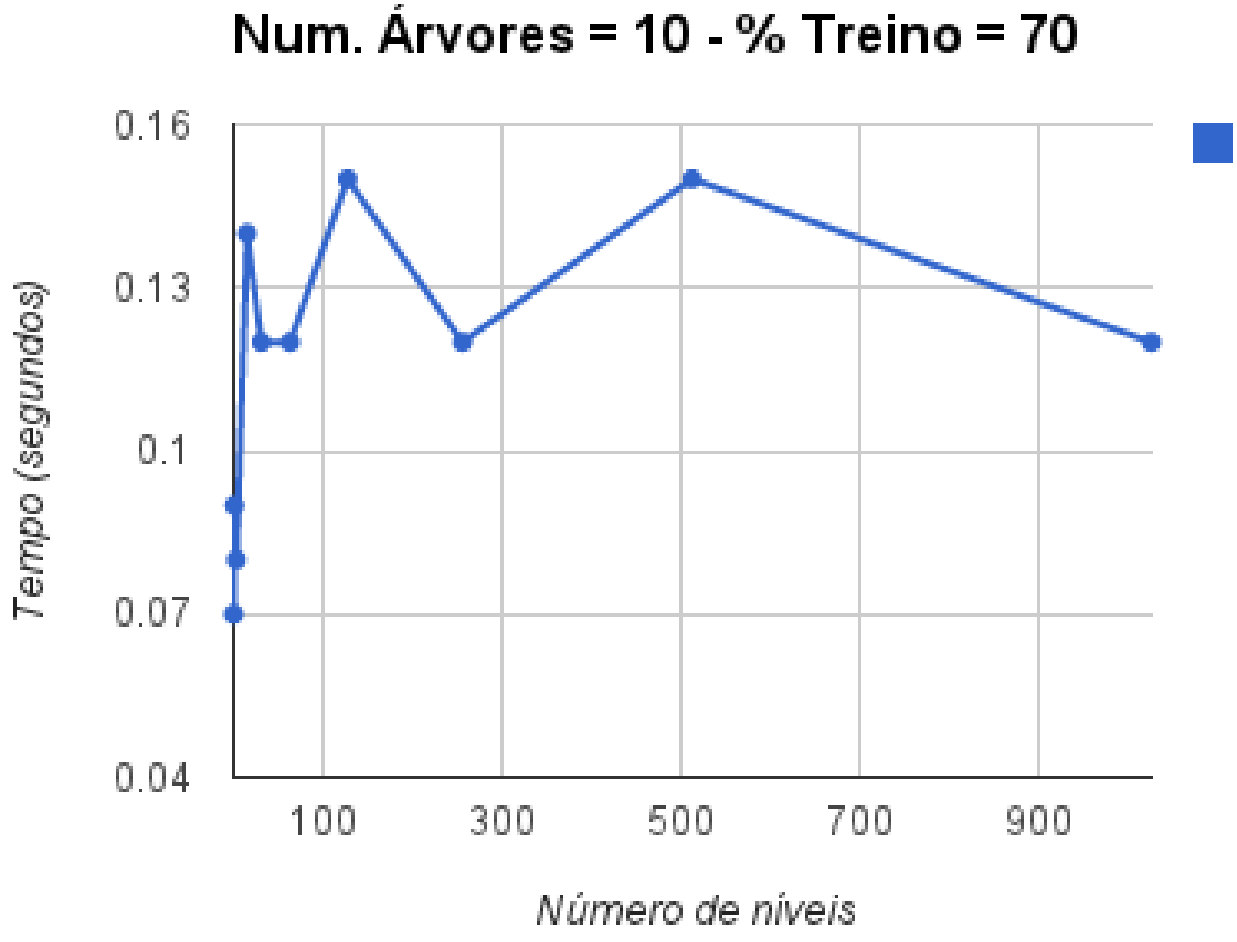
Exemplo de Random Forest



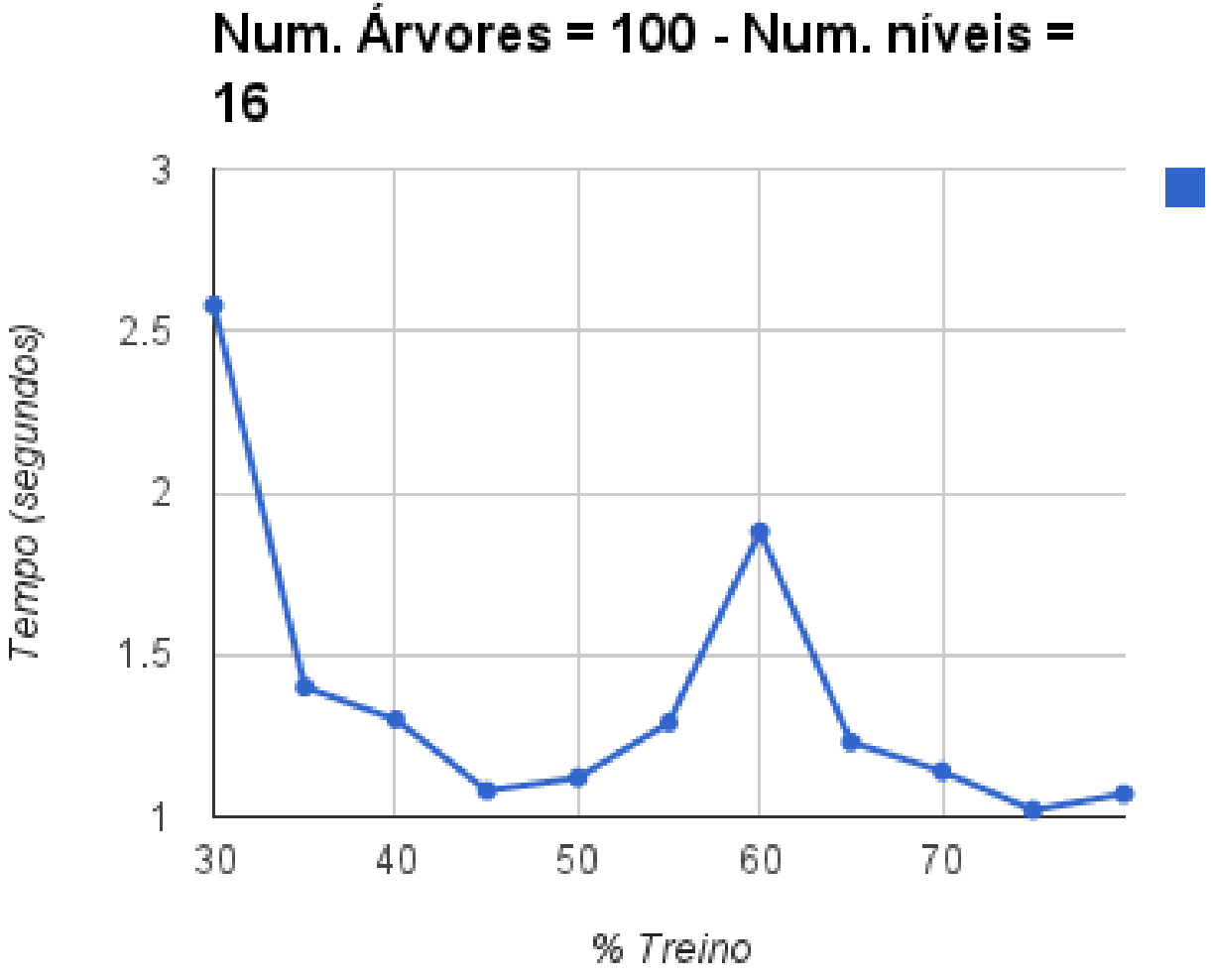
At each node:
choose some small subset of variables at random
find a variable (and a value for that variable) which optimizes the split

- Parâmetros variados para simulação
  - Número de Árvores
  - Número de Níveis (profundidade)
  - Número de Atributos Utilizados
  - Poncentagem da base de dados para treinamento da rede
- Parâmetros analisados nas simulações
  - Taxa de erro de classificação
  - Tempo de construção do modelo

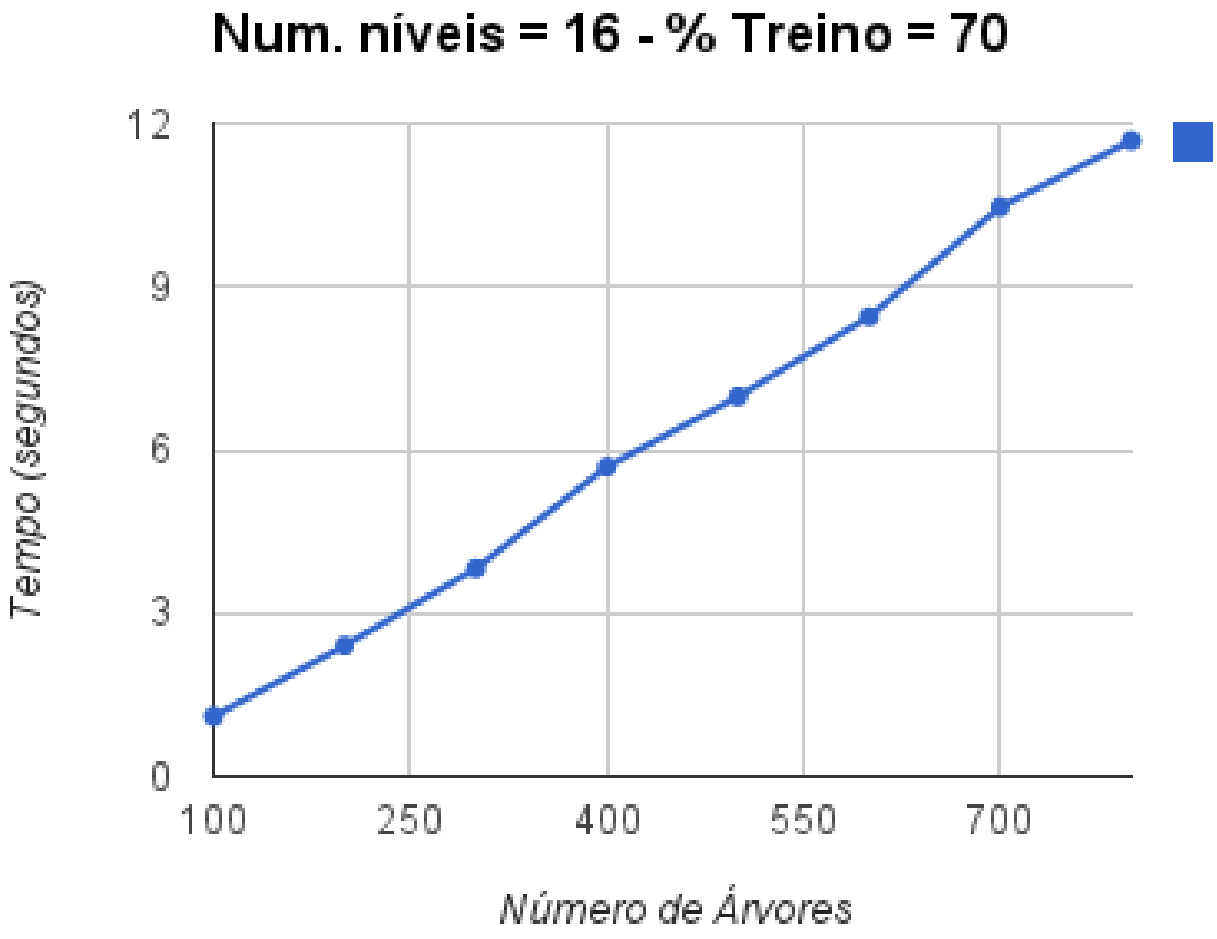
Tempo x Profundidade



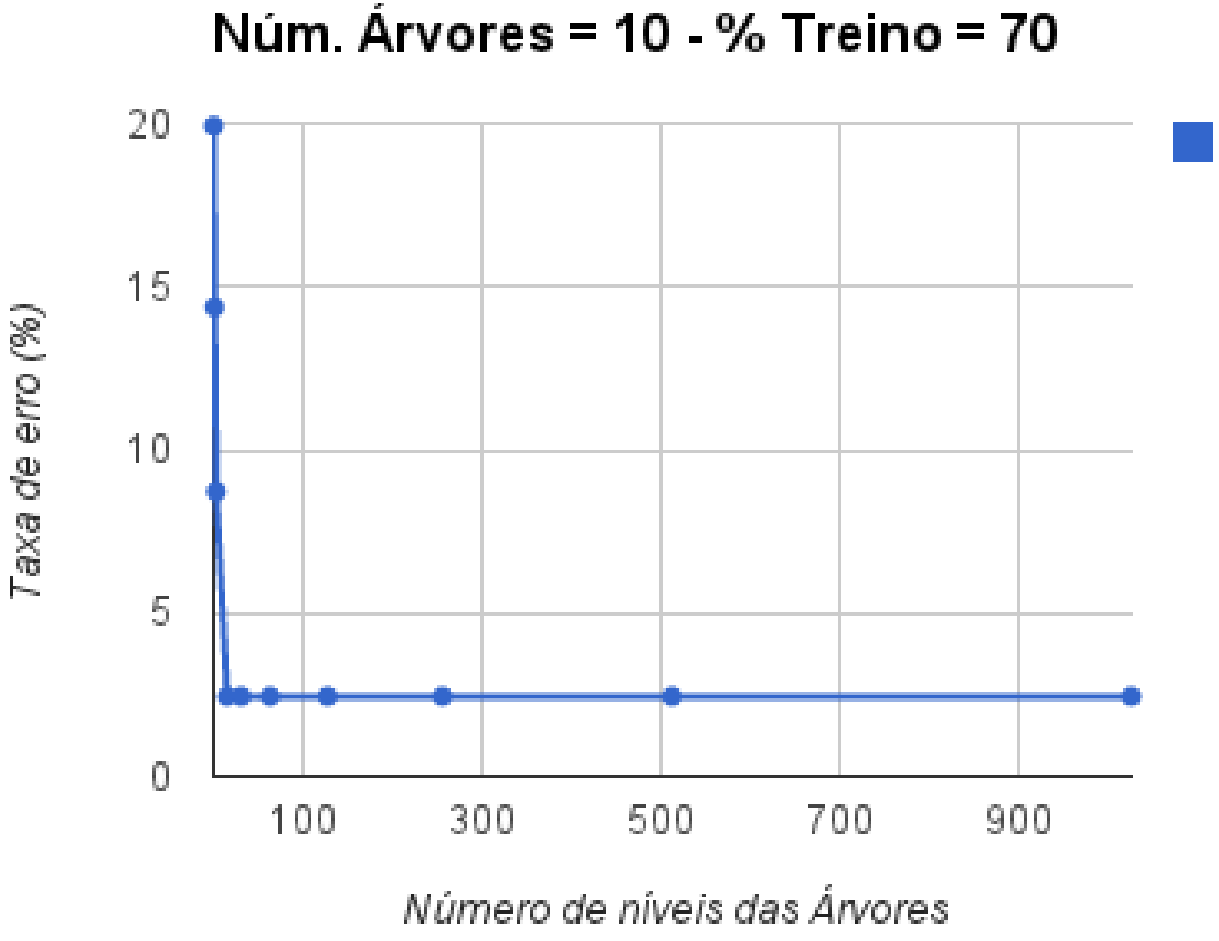
Tempo x Treino (Split)



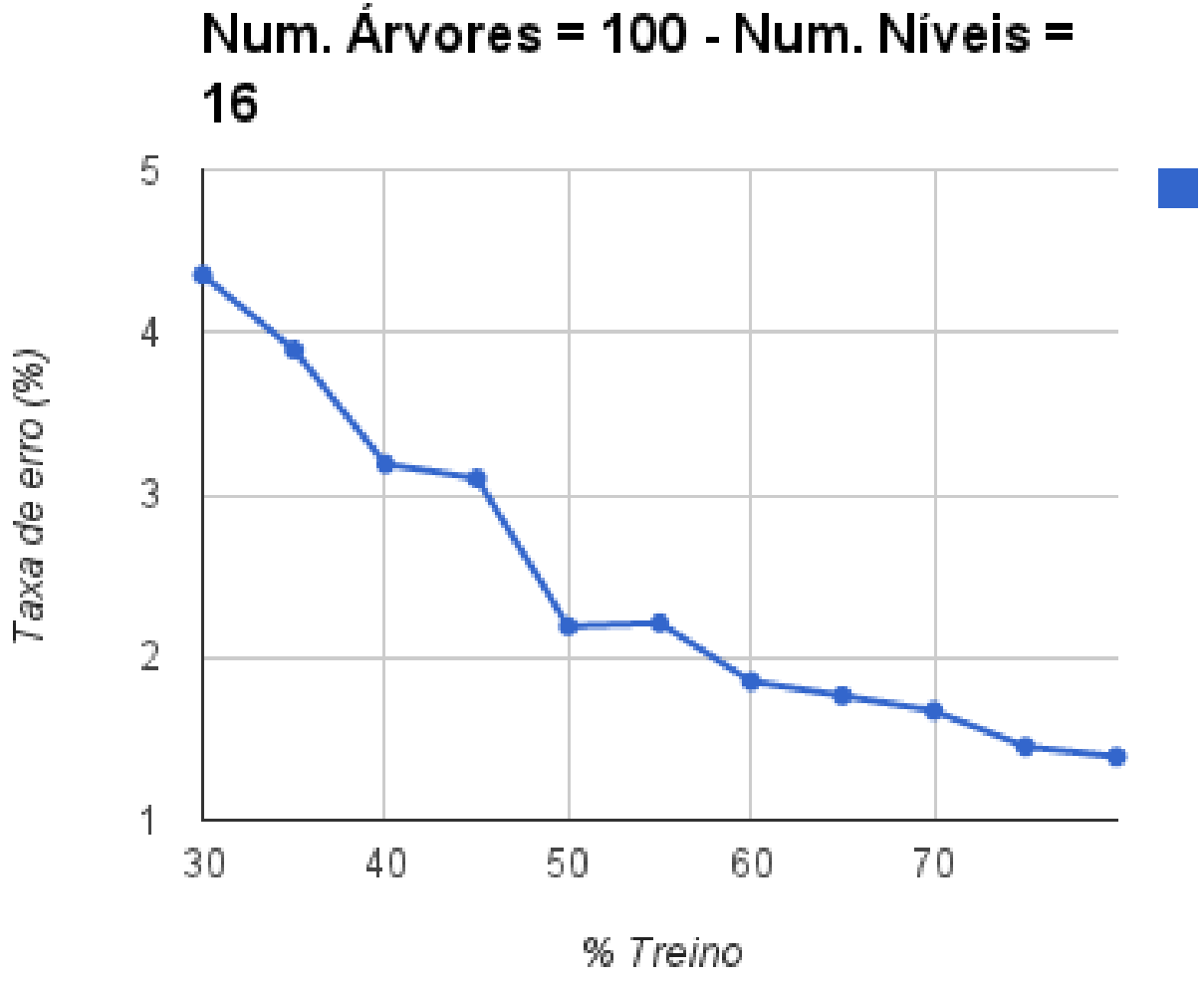
Tempo x Árvores



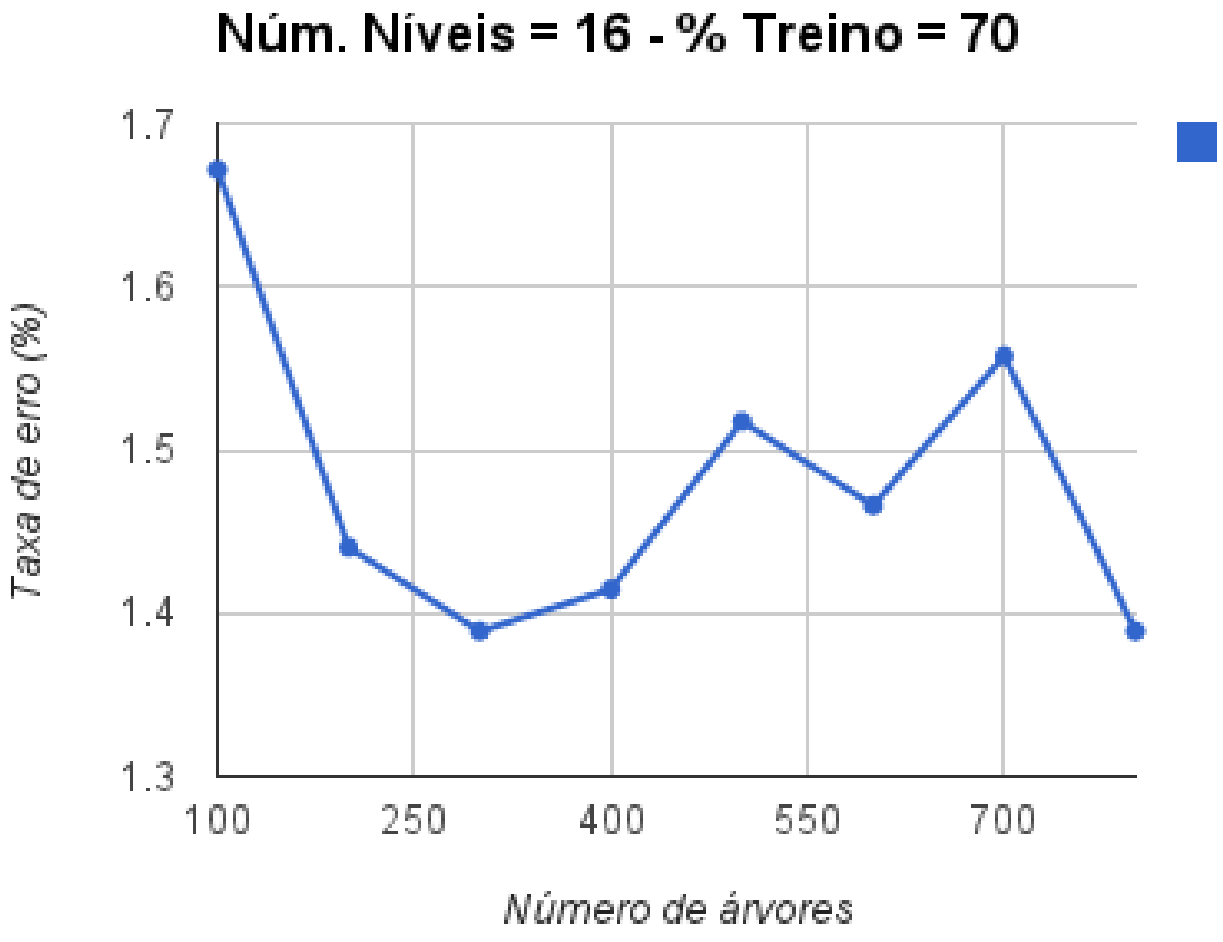
Erro x Profundidade



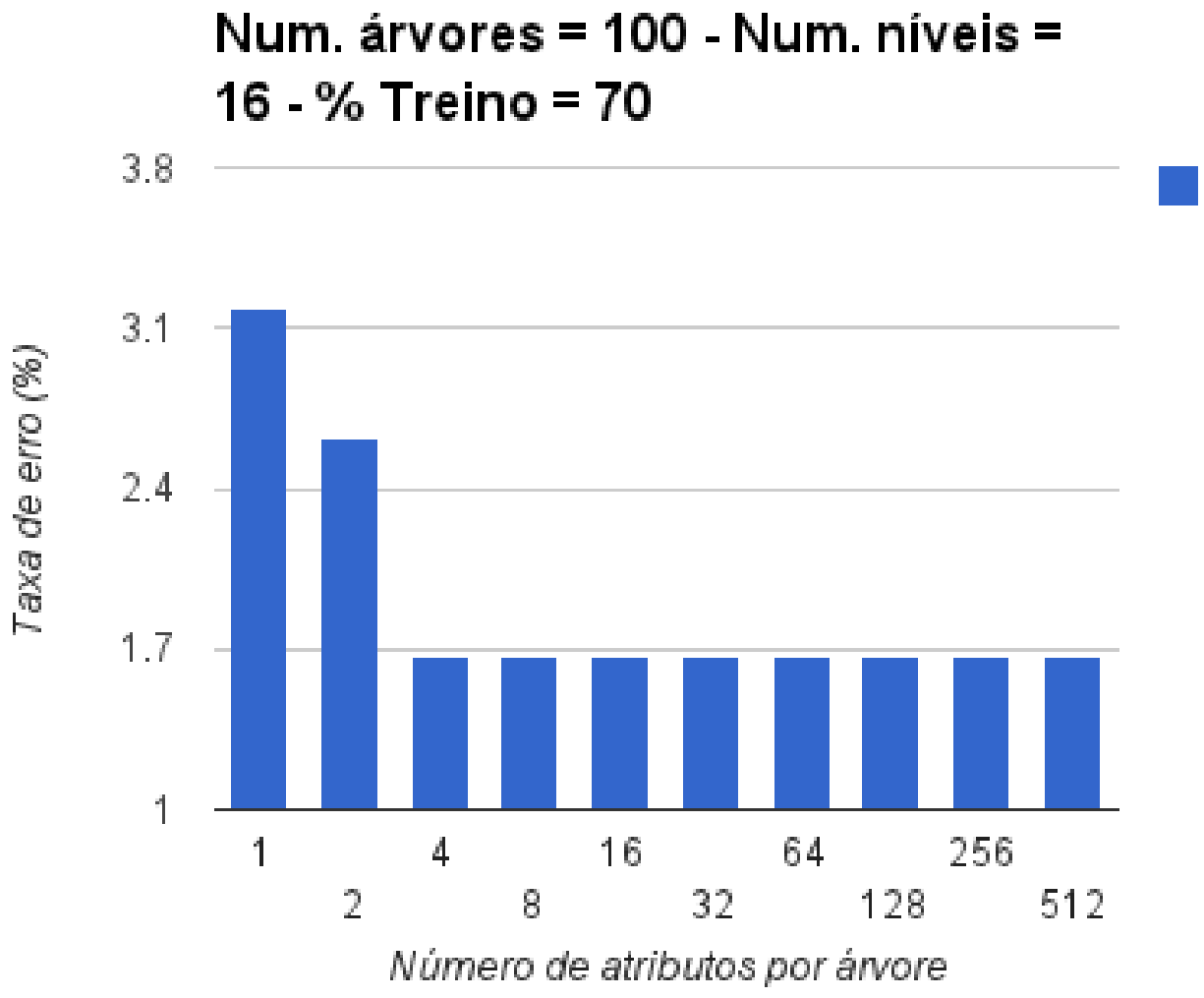
Erro x Treino (Split)



Erro x Árvores



Erro x Atributos selecionados



Prisma

- Para cada classe *c* de 1 a *n*:
  - **Passo 1** - Calcular a probabilidade de ocorrência da classe c para cada par-atributo-valor
  - **Passo 2** - Selecionar o par-atributo com a probabilidade máxima de ocorrência e crie um subconjunto de treinamento tomado como entrada compreendendo todas as instâncias que o par selecionado (para todas as classes)
  - **Passo 3** - Repetir os passos 1 e 2 para este subconjunto até o momento em que ele apresente apenas instâncias da classe *c*. A regra induzida é então a conjunção de todos os pares atributo-valor selecionados na criação deste subconjunto homogêneo.
  - **Passo 4** - Remover todas as instâncias, que satisfaçam a regra formada, do conjunto de treinamento.
- Repetir a sequência de 1 a 4 até que todas as instâncias da classe *c* tenham sido removidas.

Exemplo - parte I

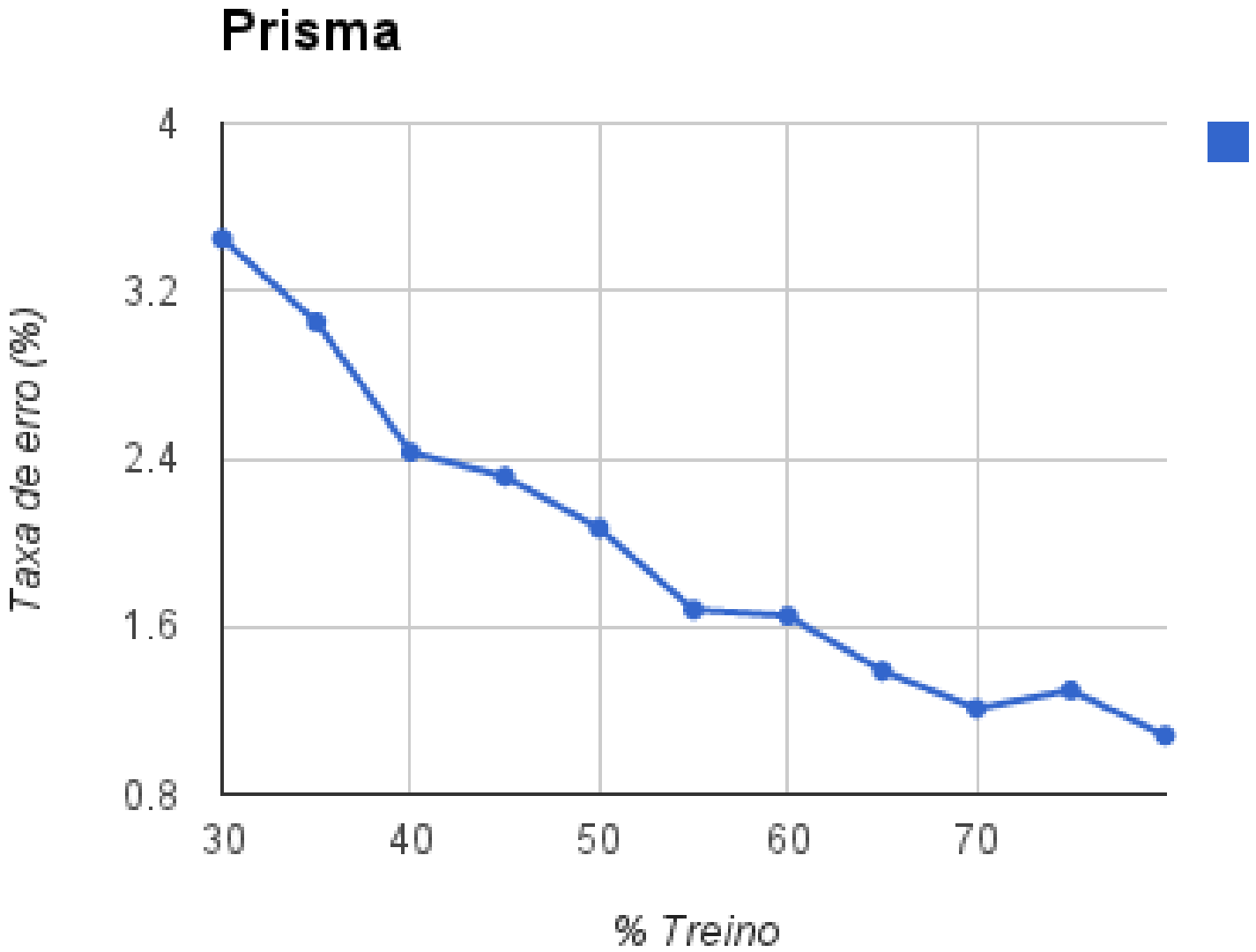
Atributo-valor	Frequência (P) para classe = recommend	Frequência (T) para atributo-valor	Probabilidade (P/T)
Finance = convenient	4	8	0.5
Finance = in-conven	1	8	0.125
Housing = convenient	1	8	0.125
Housing = less_conv	4	12	0.33
Housing = cri-tical	1	12	0.83
Children = 1	0	12	0
Children = 2	3	12	0.25

Exemplo - parte II

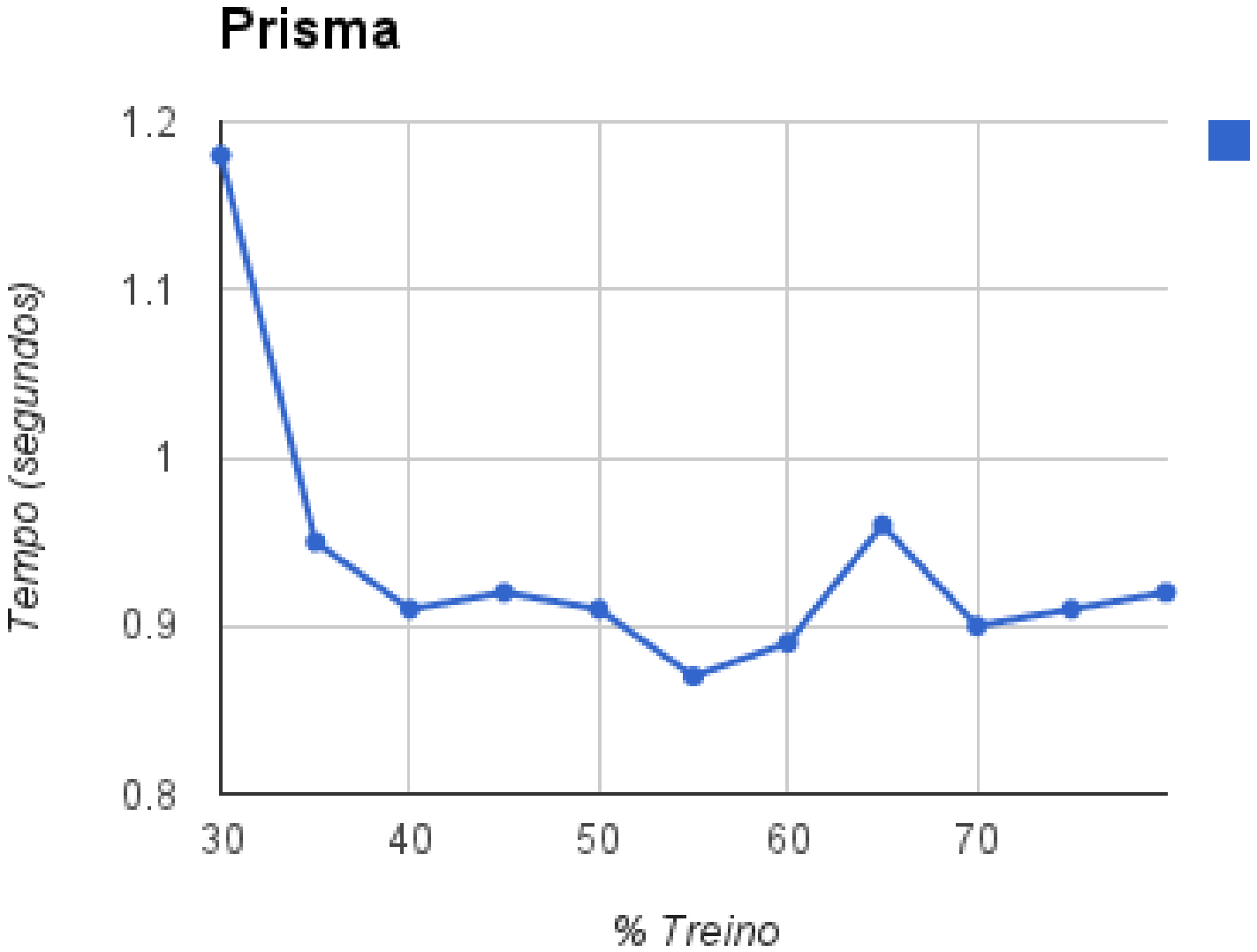
Atributo-valor AND Finance = convenient	Frequência(P) para classe = recommend	Frequência (T) para atributo-valor	Probabilidade (P/T)
Housing = convenient	4	4	1
Housing = less_conv	1	4	0.25
Children = 1	0	6	0
Children = 2	2	6	0.33
Children = 3	0	6	0
Children = more	3	6	0.5

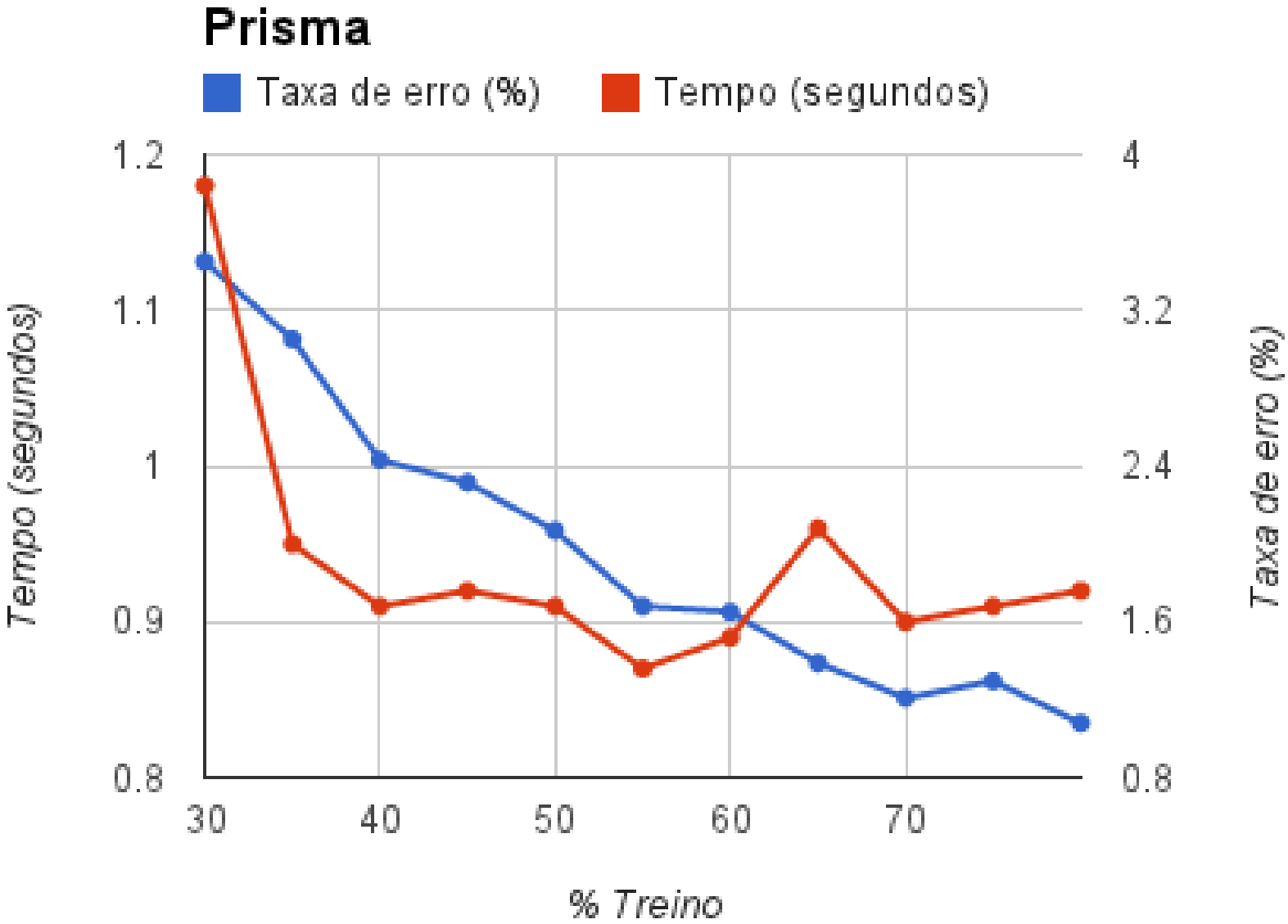
- Parâmetro variado na simulação
  - Porcentagem da base de dados para treinamento
- Parâmetros avaliados nas simulações
  - Taxa de erro de classificação
  - Tempo de construção do modelo

Resultados



Resultados

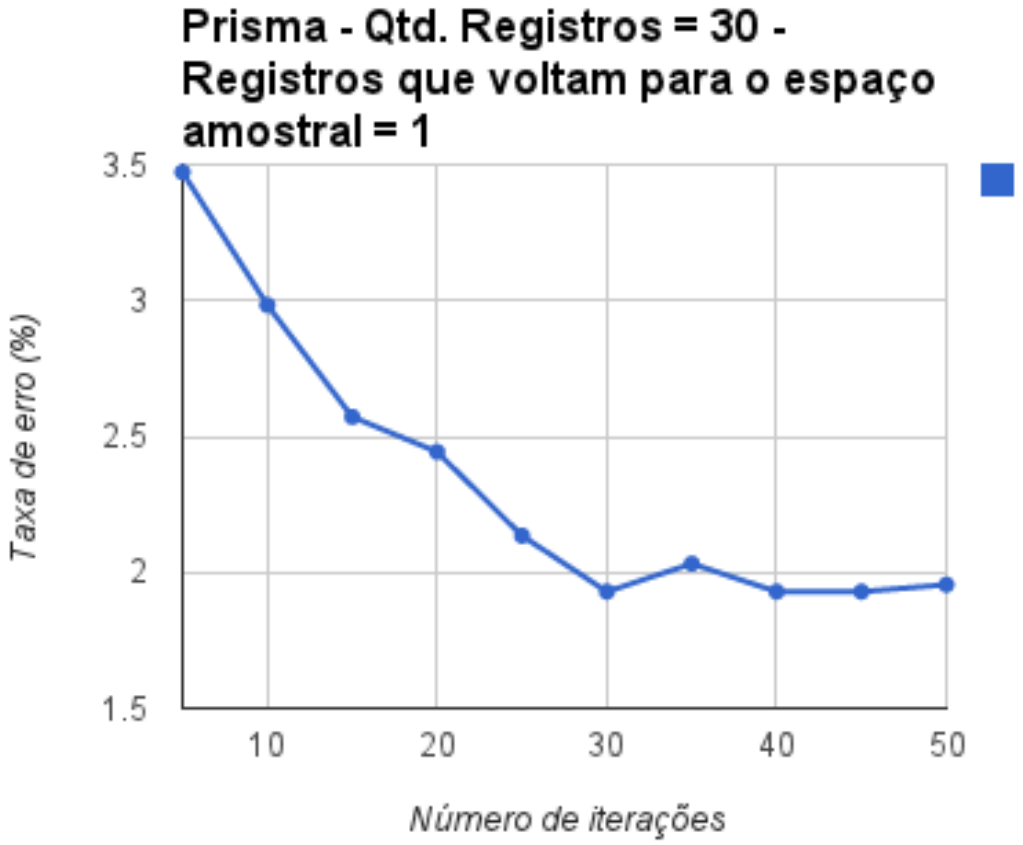



Prisma
Taxa de erro (%)
Tempo (segundos)
1.2
1.1
1
0.9
0.8
4
3.2
2.4
1.6
0.8
30
40
50
60
70
Tempo (segundos)
Taxa de erro (%)
% Treino

Testes

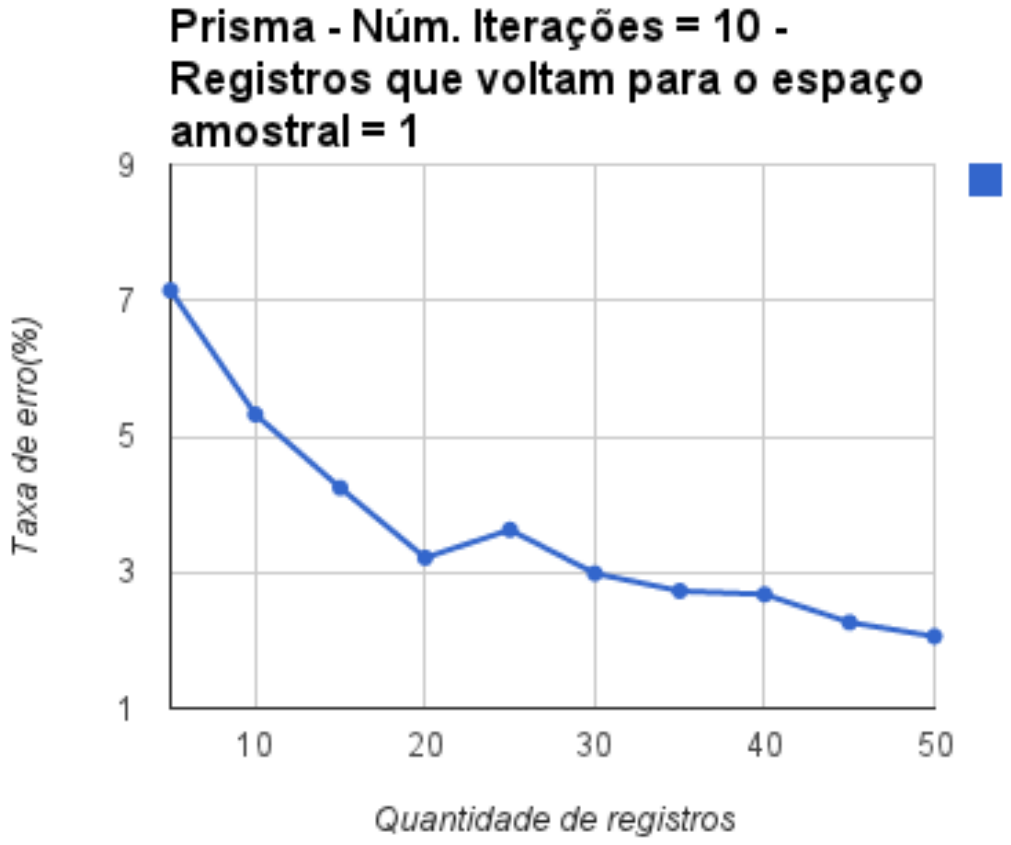
- Também foram realizados testes com bagging em conjunto com o Prisma e Rede Neural.
- No caso da Rede Neural foi utilizado o momento = 0.2, taxa de aprendizado = 0.2 e número de ciclos = 100.
- No exemplo utilizando o Weka, será variado a quantidade de registros utilizados(porcentagem) e o número de iterações.

Prisma - Variando o número de iterações



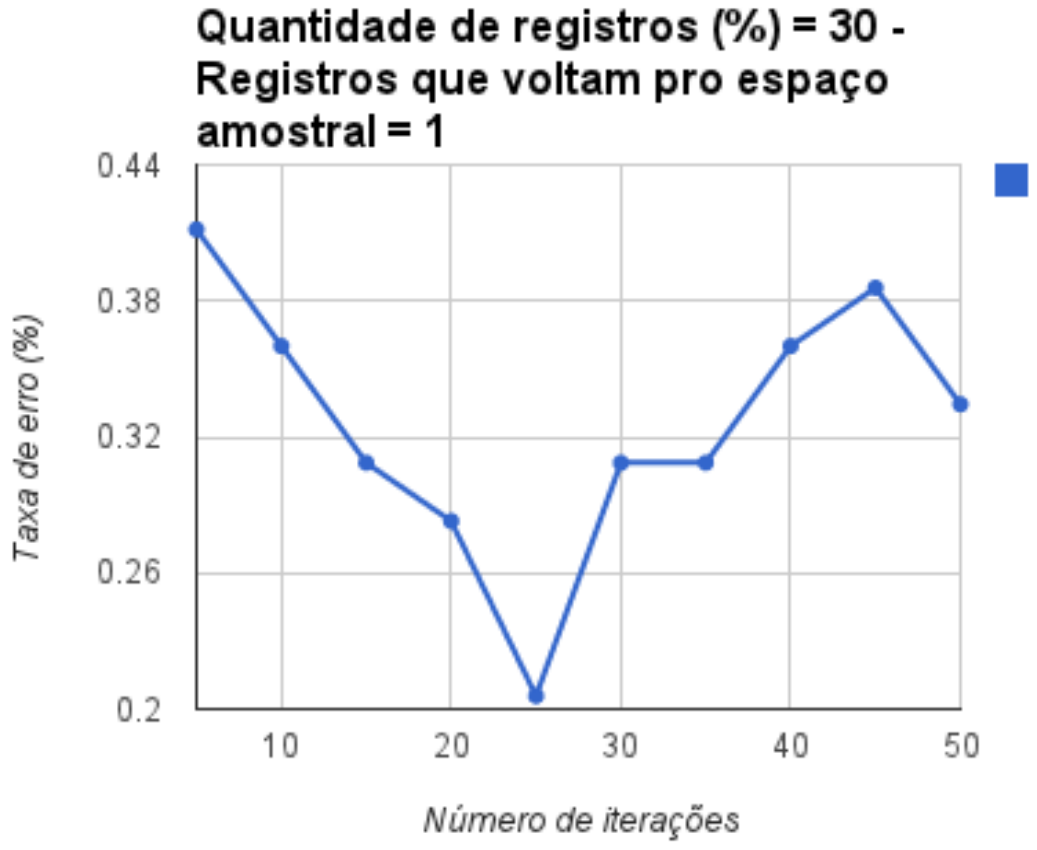
- utilizando prisma normal o erro = 1.2088%

Prisma - Variando a quantidade de registros



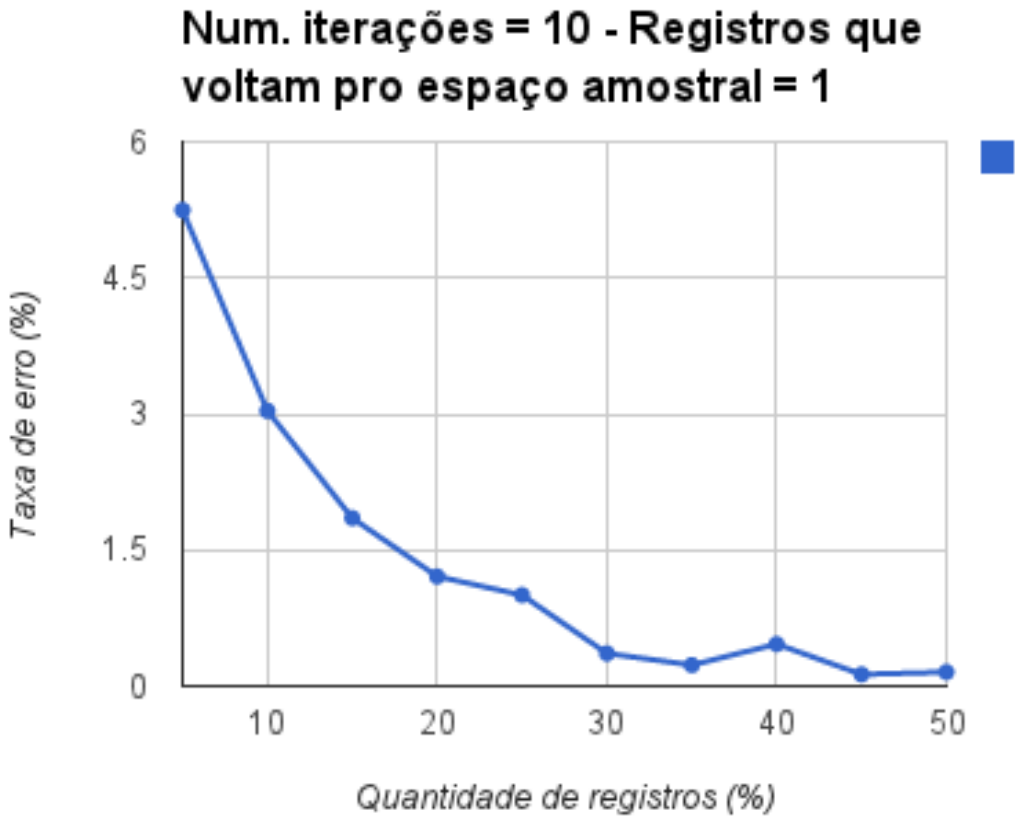
- utilizando prisma normal o erro = 1.2088%

Rede Neural - Variando o número de iterações



- utilizando a Rede neural normal o erro = 0.0257

Rede Neural - Variando a quantidade de registros



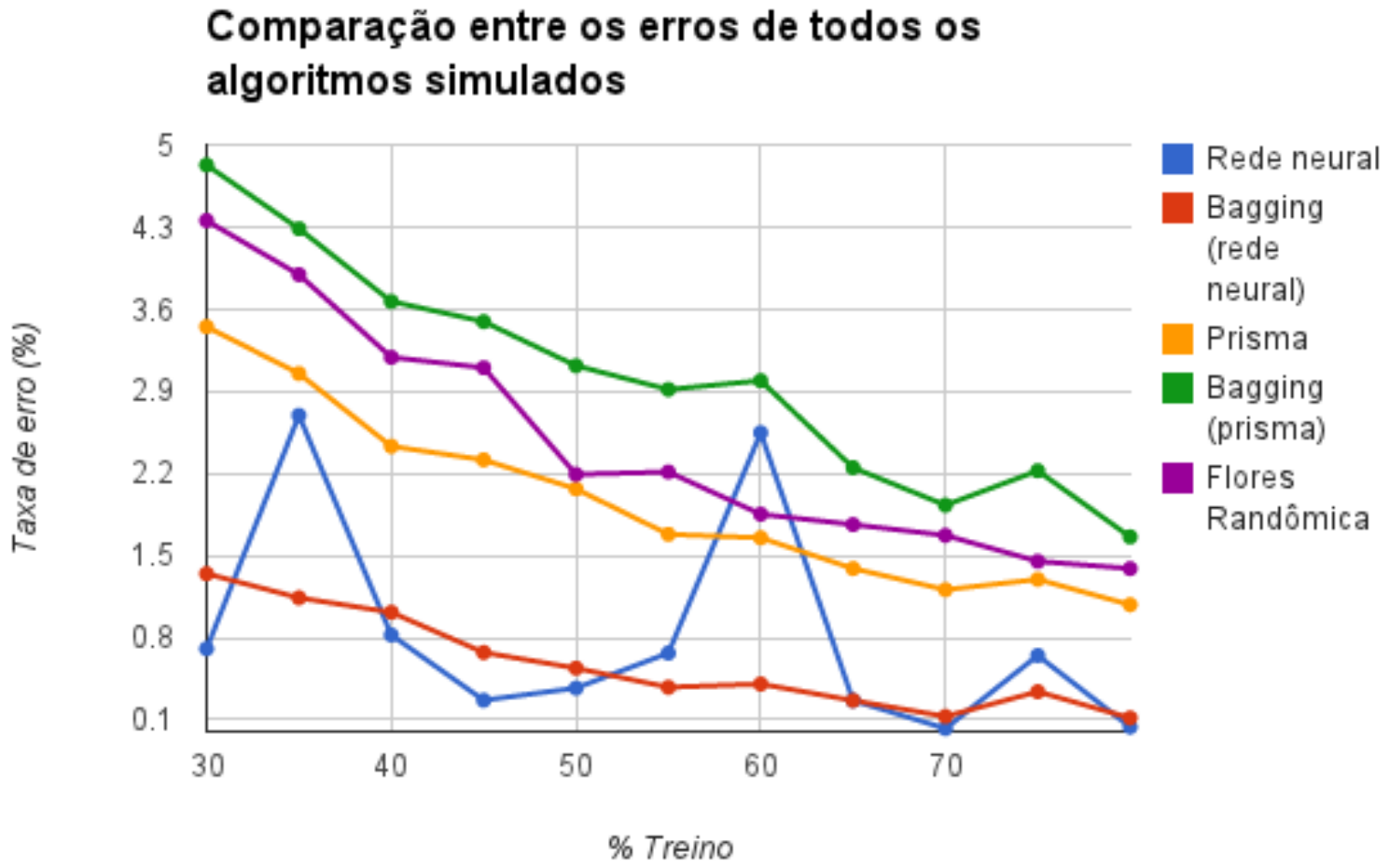
- utilizando a Rede neural normal o erro = 0.0257

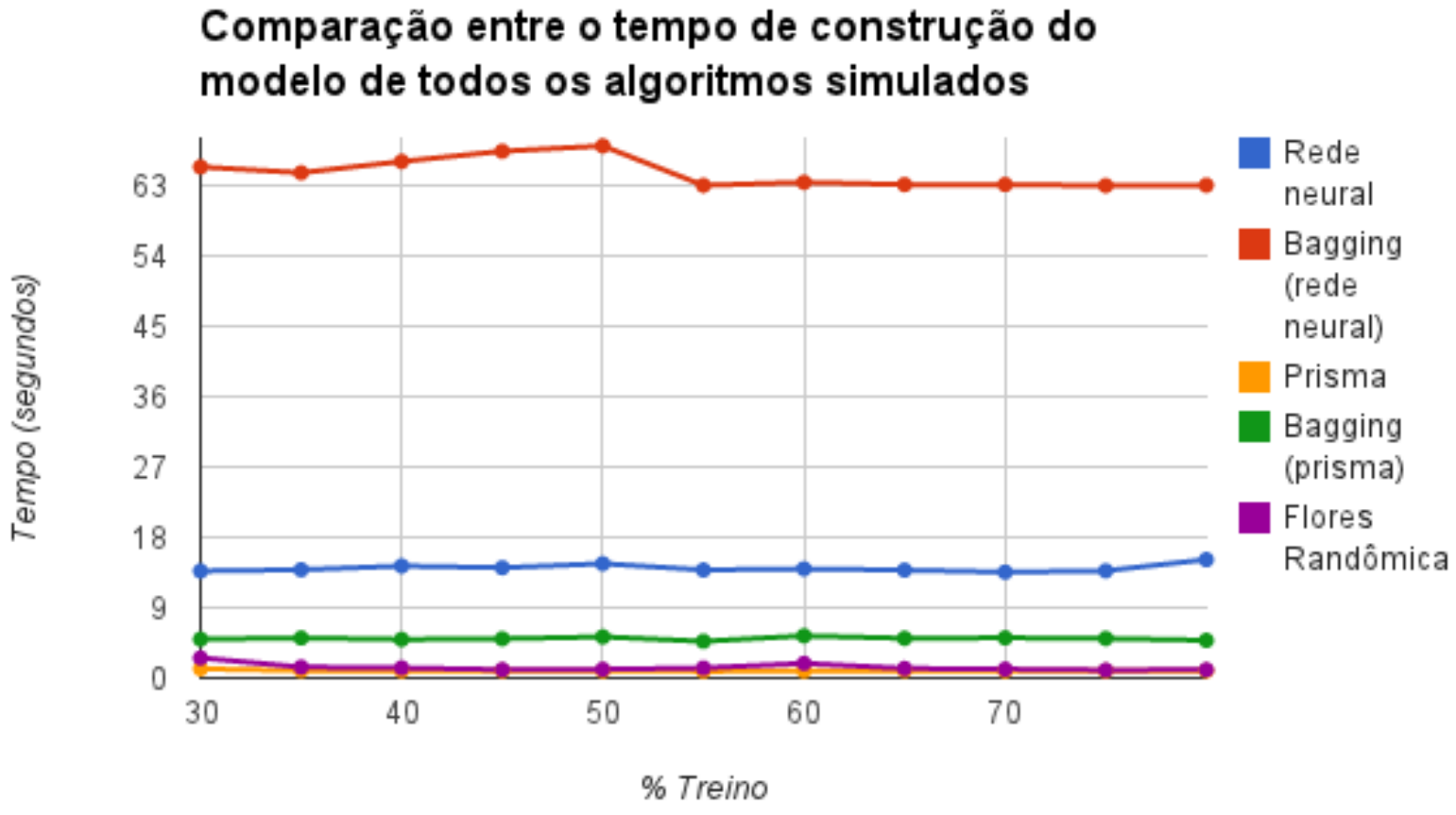
Conclusões

- Para uma base de dados pequena a mistura com o bagging não foi satisfatória.
- Não foi encontrada implementações que realizem a mistura entre os algoritmos como o Random Forest.

Conclusões gerais

- A rede neural encontrou o menor erro, porém com um tempo elevado
- O algoritimo prisma obteve um erro satisfatório com o menor tempo
- A utilização do algoritmo *bagging* não se mostrou vantajosa para uma base de dados pequena
- O algoritmo floresta randômica oferece um bom custo X benefício entre erro e tempo de execução


Comparação entre os erros de todos os algoritmos simulados
Taxa de erro (%)
5
4.3
3.6
2.9
2.2
1.5
0.8
0.1
30
40
50
60
70
% Treino
Rede neural
Bagging (rede neural)
Prisma
Bagging (prisma)
Flores Randômica


Comparação entre o tempo de construção do modelo de todos os algoritmos simulados
Tempo (segundos)
63
54
45
36
27
18
9
0
30
40
50
60
70
% Treino
Rede neural
Bagging (rede neural)
Prisma
Bagging (prisma)
Flores Randômica