Análise de Algoritmos de Aprendizagem de Máquina para Identificação de Ataques Utilizando a Base NSL-KDD

Autor: Hugo Silveira Sousa
Coautor: Wendley Souza da Silva
Orientador: Luis Eduardo Araripe Gomes da Silva

Universidade Federal do Ceará - Campus Sobral
PET - Engenharia de Computação
Encontros Universitários 2018

31 de Outubro 2018

- Palavras-chave: IDS, Aprendizagem de Máquina, IoT.

Sumário

Introdução

- "Segundo pesquisadores da Kaspersky Lab, o número total de amostras de malware que visam dispositivos inteligentes chegou a mais de 7.000, sendo que mais da metade deles surgiu em 2017. Com mais de 6 bilhões de dispositivos inteligentes em uso no mundo, o risco de um malware atingir as vidas conectadas dos usuários é cada vez maior."[1] (Kaspersky Lab)

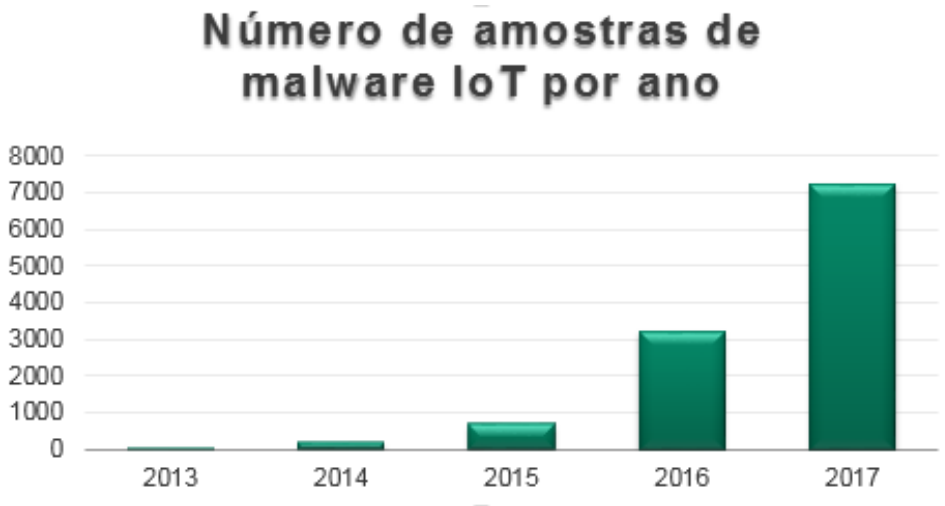

Figura 1: Número de Malwares [1] (Kaspersky Lab)

Introdução

- Os sistemas de detecção de intrusão(IDS) têm como objetivo reconhecer comportamentos intrusivos em uma rede e alertar os administradores ou realizar ações de contra medidas automaticamente [2] (Bace & Mell, 2001)

Introdução

- Por que usar Aprendizagem de Máquina?
  - K-Nearest Neighbour (KNN)
  - Support Vector Machine (SVM)
  - Naive Bayes (NB)
  - Árvore de Decisão (AD)
  - Random Forest (RF)
  - Regressão Logística (RL)

Objetivos

- Estudar Métodos de Aprendizagem de Máquina
- Implementar um IDS
- Aplicar em um cenário IoT

Base de Dados

- Base NSL-KDD
  - KDD-CUP-99[3]
  - MIT Lincoln Labs
  - Simulação de uma rede LAN da U.S. Air Force
  - 42 atributos

Deniel of service (DoS)	User to root (U2R)	Remote to local (R2L)	Probing (PROBE)
Back	Perl	FTP write	IP sweep
Ping of death	Buffer overflow	Guess password	NMAP
Neptune	Load module	IMAP	Port sweep
Smurf	Rootkit	Multi HOP	Satan
Land		Phf	
Teardrop		SPY	

Tabela 1: Tipos de ataques [4] (Ashfaq et al., 2017)

Biblioteca

- Biblioteca SciKitLearn

Figura 2: Logo da biblioteca ScikitLearn.[5] (Doc. Scikit-Learn)

Pré-Processamento

- Transformação das Variáveis Categóricas:

	0	1	2
1	A	39	a
2	B	50	a
3	C	38	b
4	C	53	c
5	C	28	a
6	C	37	d
7	C	49	e
8	B	52	b
9	C	31	d
10	C	42	a

	0	1	2
1	1	39	1
2	2	50	1
3	3	38	2
4	3	53	3
5	3	28	1
6	3	37	4
7	3	49	5
8	2	52	2
9	3	31	4
10	3	42	1

Pré-Processamento

- Normalização:

	A	B	C
1	34415.2	48.1172	6564.75
2	57317.2	63.108	8020.95
3	42709.5	45.752	6103.64
4	66952.7	18.5843	8770.1
5	24904.1	57.4716	15.4986
6	48430.4	26.8091	5722.58
7	24500.1	32.8975	2971
8	40654.9	55.4969	4755.83
9	25075.9	39.7764	1409.23
10	64131.4	25.6796	4351.03

	A	B	C
1	-0.557066	0.471545	0.642551
2	0.944933	1.51911	1.19416
3	-0.0130902	0.306265	0.467886
4	1.57687	-1.59222	1.47794
5	-1.18084	1.12524	-1.8383
6	0.362103	-1.01747	0.32354
7	-1.20733	-0.592006	-0.718755
8	-0.147841	0.98724	-0.0426663
9	-1.16957	-0.111311	-1.31035
10	1.39184	-1.0964	-0.196003

Pré-Processamento

- One-Hot Encoding

	0
1	A
2	B
3	C
4	C
5	C
6	C
7	C
8	B
9	C
10	C

	0
1	1
2	2
3	3
4	3
5	3
6	3
7	3
8	2
9	3
10	3

	1(A)	2(B)	3(C)
1	1	0	0
2	0	1	0
3	0	0	1
4	0	0	1
5	0	0	1
6	0	0	1
7	0	0	1
8	0	1	0
9	0	0	1
10	0	0	1

Experimento

- Computador do Experimento:
  - Processador Intel(R) Core(TM) i5-4460 CPU @ 3.20GHz
  - 16 GB de Memória RAM
  - Velocidade de Rotacão do HD 7200 RPM
  - Windows 10 x64
  - Anaconda Navigator 1.8.7
  - IDE Spyder 3.2.8
  - Python 3.6

Experimento

Estruturas dos Algoritmos (5 testes)

- Código 1
  1. Carregar Base
  2. Pré-Processamento
     1. Transformação das Variáveis Categóricas
     2. One-Hot Encoding
     3. Normalização
  3. Treino (85%)
  4. Teste (15%)
  5. Obtenção dos Resultados
- Código 2
  1. Carregar Base
  2. Pré-Processamento
     1. Transformação das Variáveis Categóricas
     2. Normalização
  3. Treino (85%)
  4. Teste (15%)
  5. Obtenção dos Resultados
- Código 3
  1. Carregar Base
  2. Pré-Processamento
     1. Transformação das Variáveis Categóricas
     2. One-Hot Encoding
  3. Treino (85%)
  4. Teste (15%)
  5. Obtenção dos Resultados
- Código 4
  1. Carregar Base
  2. Pré-Processamento
     1. Transformação das Variáveis Categóricas
  3. Treino (85%)
  4. Teste (15%)
  5. Obtenção dos Resultados

- https://github.com/hugosousa111/AnalisesAlgoritmos.git

Resultados

- Árvore de Decisão (AD)

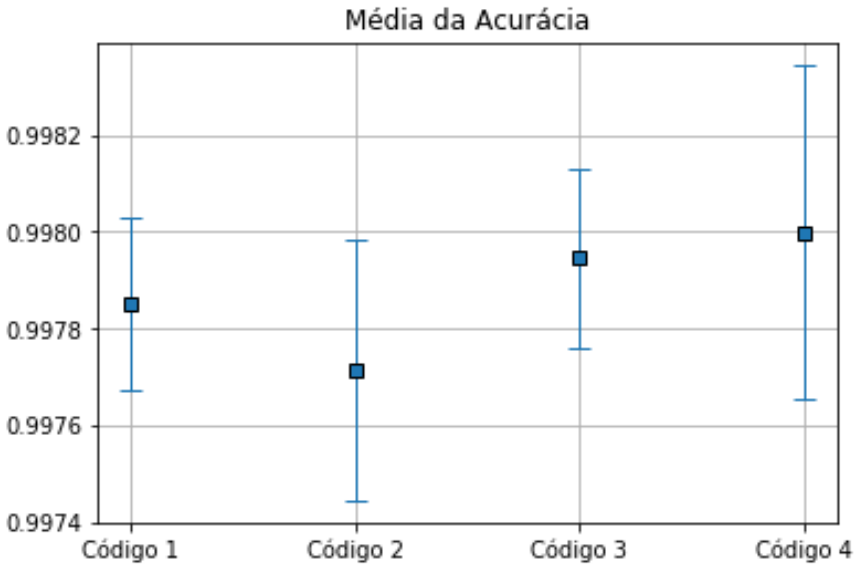

Figura 3: Gráfico Média da Acurácia AD

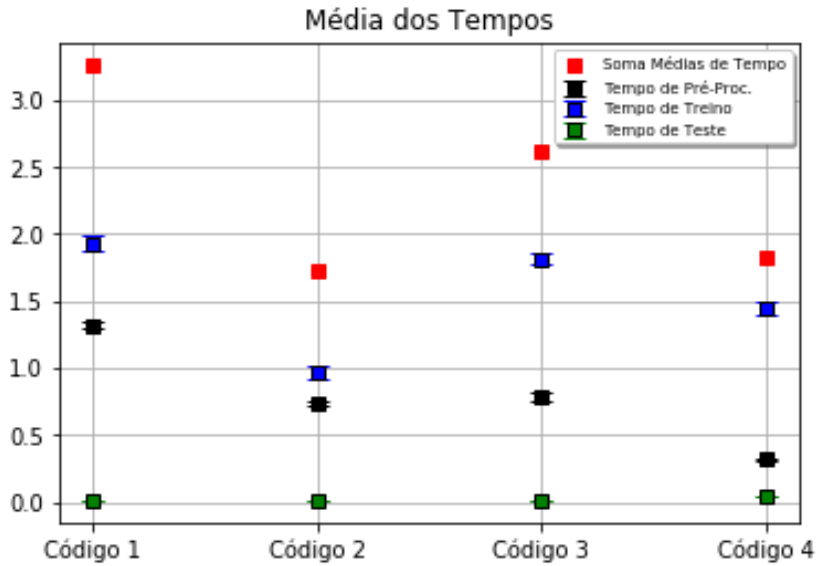

Figura 4: Gráfico Média dos Tempos AD

- Código 1: One-Hot + Normalização
- Código 2: Normalização
- Código 3: One-Hot
- Código 4: Sem One-Hot e Normalização

Resultados

- Random Forest (RF)

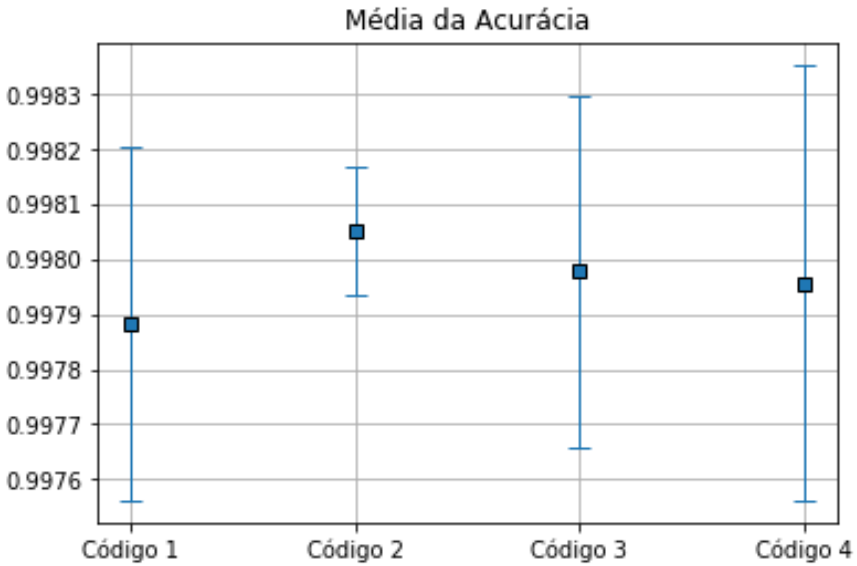

Figura 5: Gráfico Média da Acurácia RF

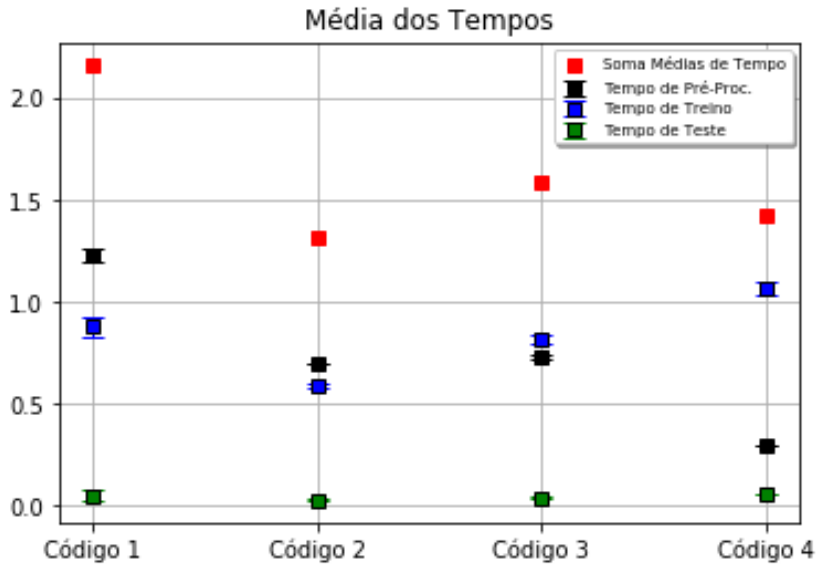

Figura 6: Gráfico Média dos Tempos RF

- Código 1: One-Hot + Normalização
- Código 2: Normalização
- Código 3: One-Hot
- Código 4: Sem One-Hot e Normalização

Resultados

- Naive Bayes (NB)

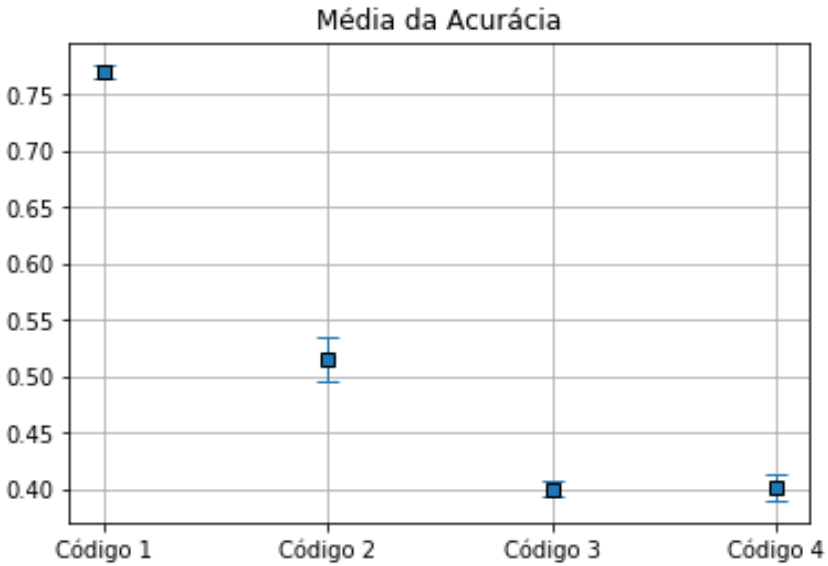

Figura 7: Gráfico Média da Acurácia NB

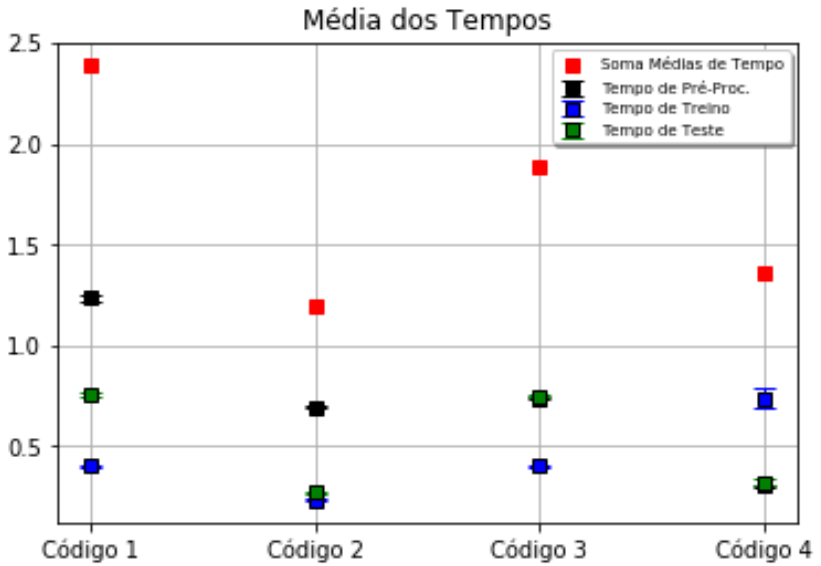

Figura 8: Gráfico Média dos Tempos NB

- Código 1: One-Hot + Normalização
- Código 2: Normalização
- Código 3: One-Hot
- Código 4: Sem One-Hot e Normalização

Resultados

- Regressão Logística (RL)

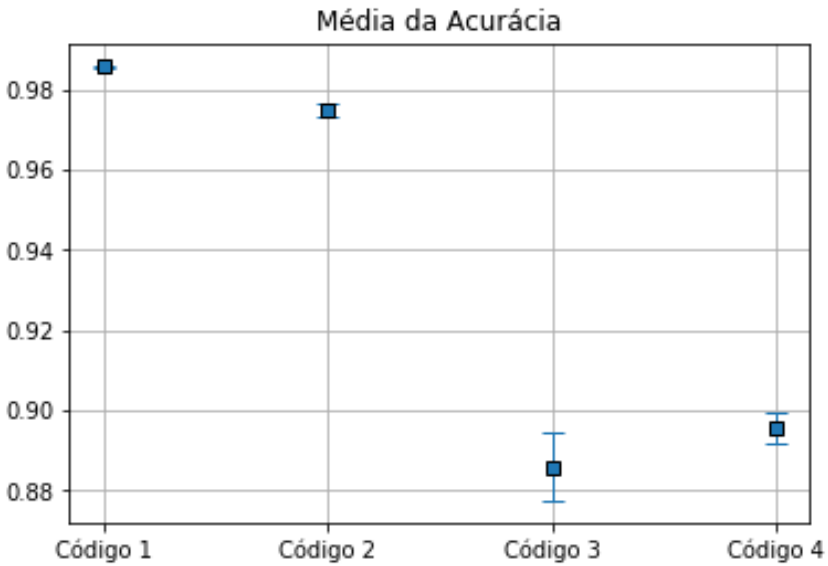

Figura 9: Gráfico Média da Acurácia RL

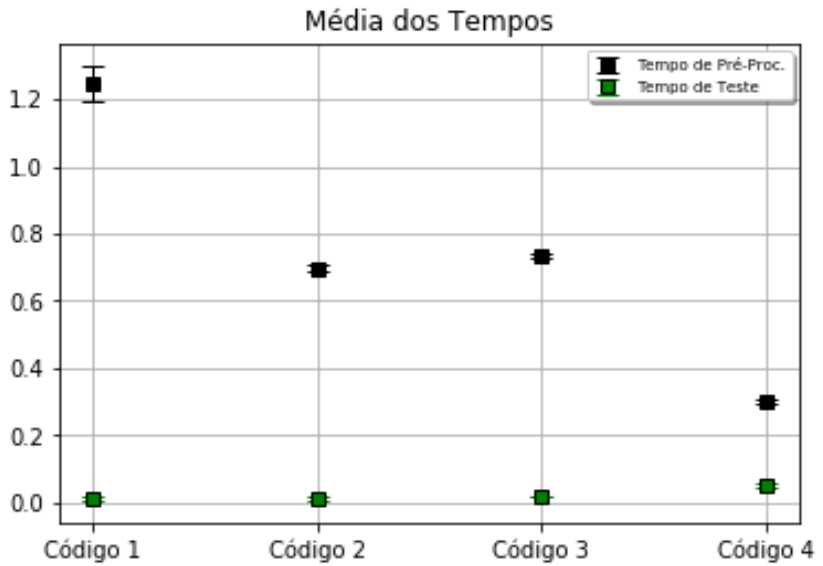

Figura 10: Gráfico Média dos Tempos RL

- Código 1: One-Hot + Normalização
- Código 2: Normalização
- Código 3: One-Hot
- Código 4: Sem One-Hot e Normalização

Resultados

- Regressão Logística (RL)

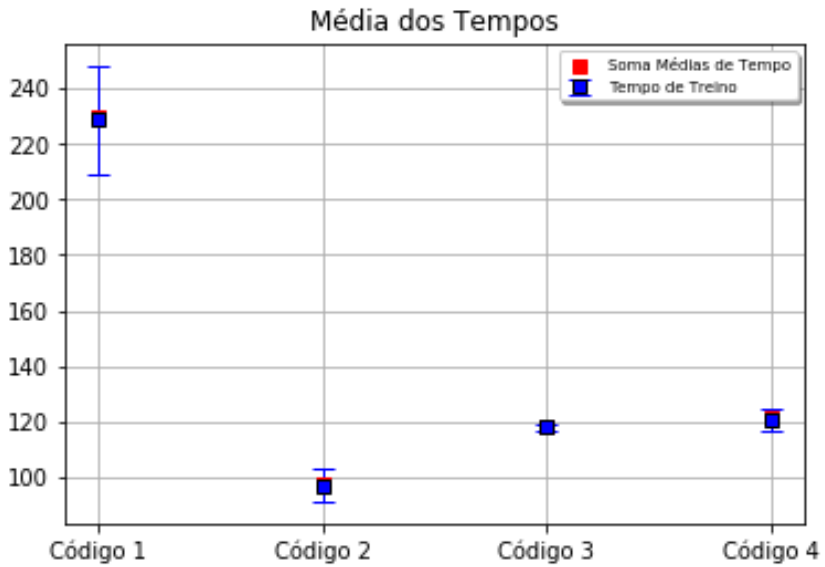

Figura 11: Gráfico Média dos Tempos RL

- Código 1: One-Hot + Normalização
- Código 2: Normalização
- Código 3: One-Hot
- Código 4: Sem One-Hot e Normalização

Resultados

- Support Vector Machine (SVM)

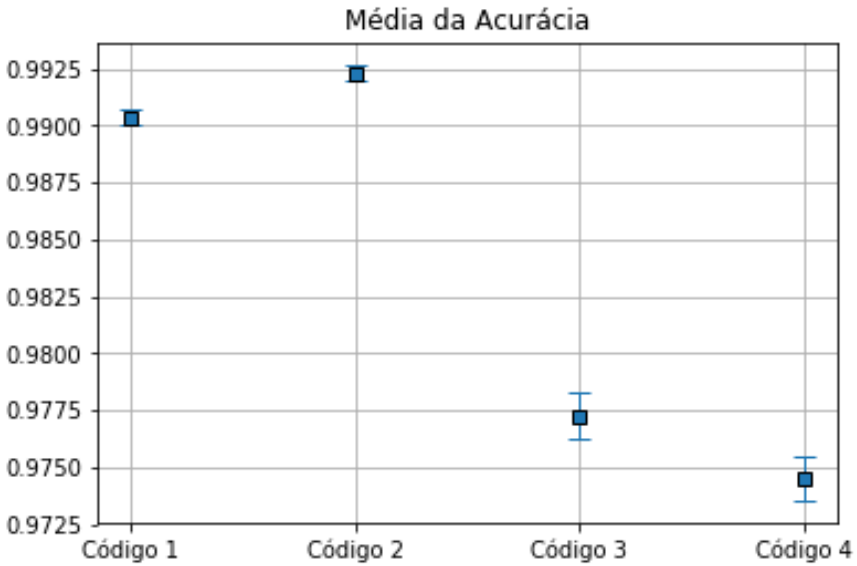

Figura 12: Gráfico Média da Acurácia SVM

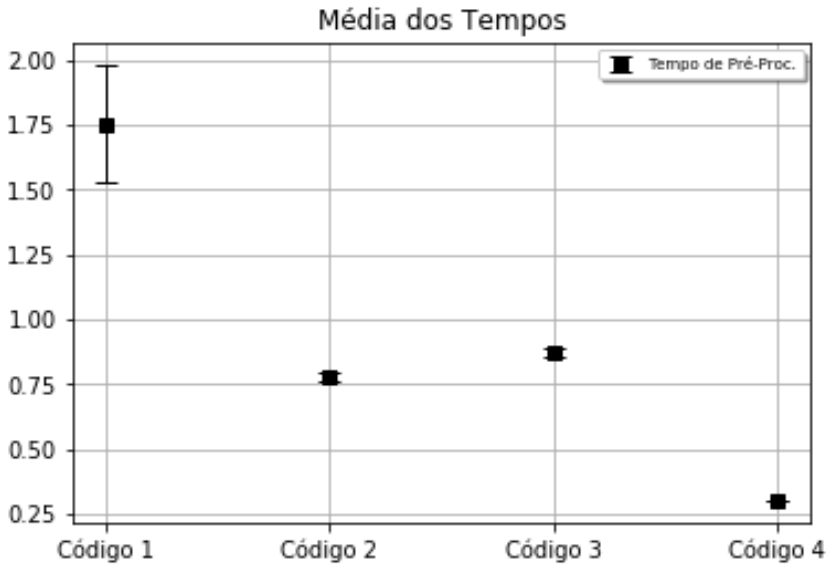

Figura 13: Gráfico Média dos Tempos SVM

- Código 1: One-Hot + Normalização
- Código 2: Normalização
- Código 3: One-Hot
- Código 4: Sem One-Hot e Normalização

Resultados

- Support Vector Machine (SVM)

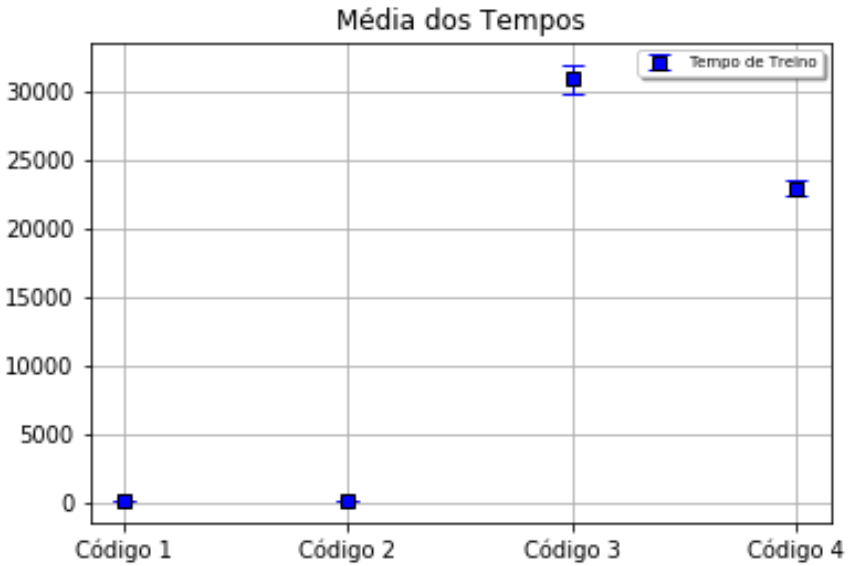

Figura 14: Gráfico Média dos Tempos SVM

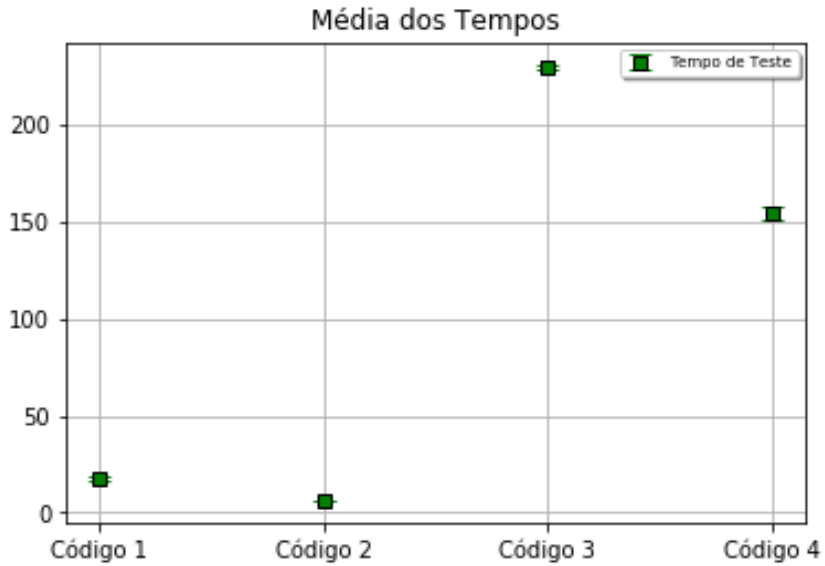

Figura 15: Gráfico Média dos Tempos SVM

- Código 1: One-Hot + Normalização
- Código 2: Normalização
- Código 3: One-Hot
- Código 4: Sem One-Hot e Normalização

Resultados

- Support Vector Machine (SVM)

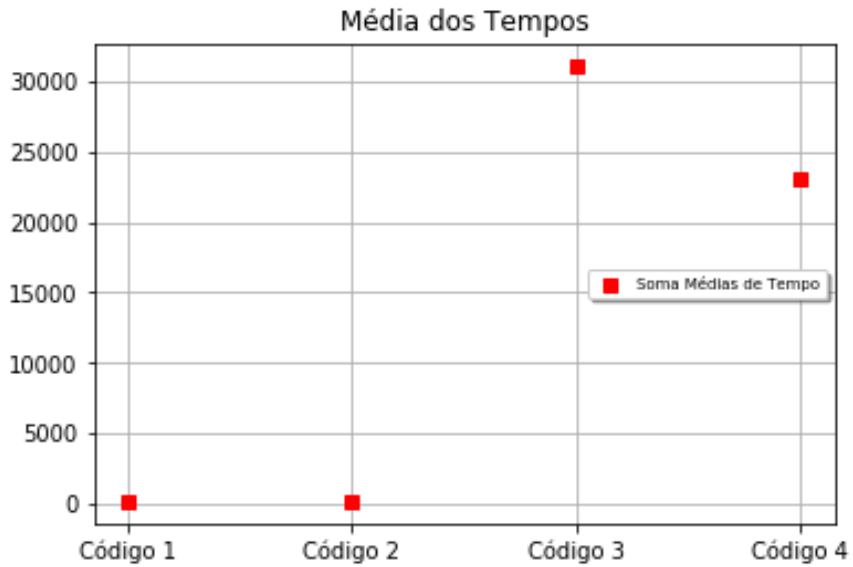

Figura 16: Gráfico Média dos Tempos SVM

- Código 1: One-Hot + Normalização
- Código 2: Normalização
- Código 3: One-Hot
- Código 4: Sem One-Hot e Normalização

Resultados

- K-Nearest Neighbour (KNN)

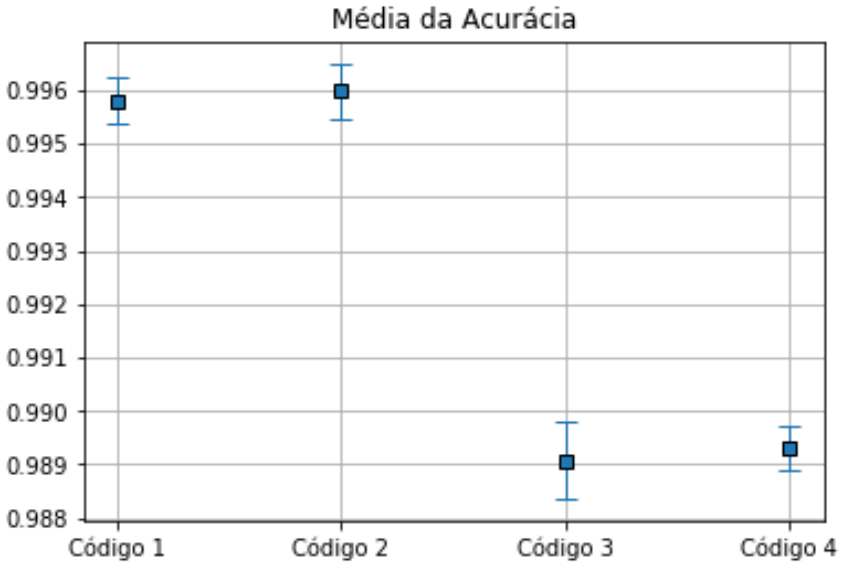

Figura 17: Gráfico Média da Acurácia KNN

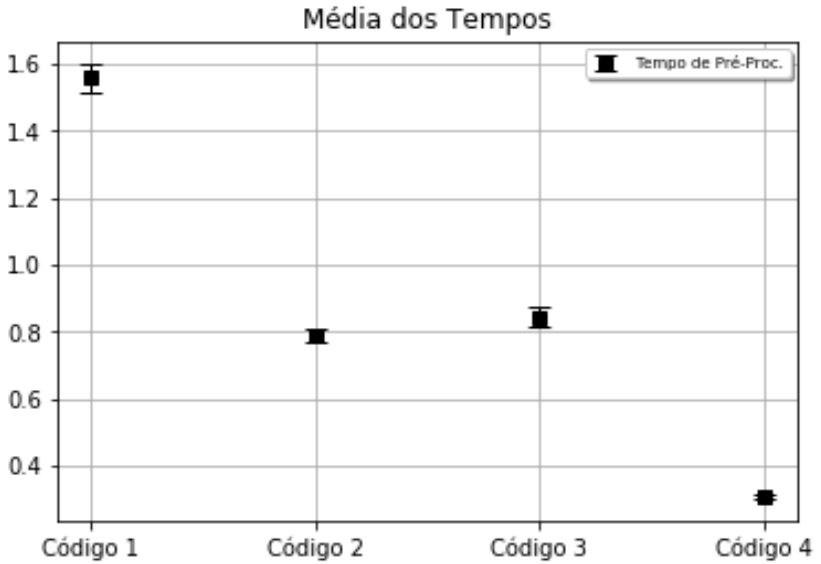

Figura 18: Gráfico Média dos Tempos KNN

- Código 1: One-Hot + Normalização
- Código 2: Normalização
- Código 3: One-Hot
- Código 4: Sem One-Hot e Normalização

Resultados

- K-Nearest Neighbour (KNN)

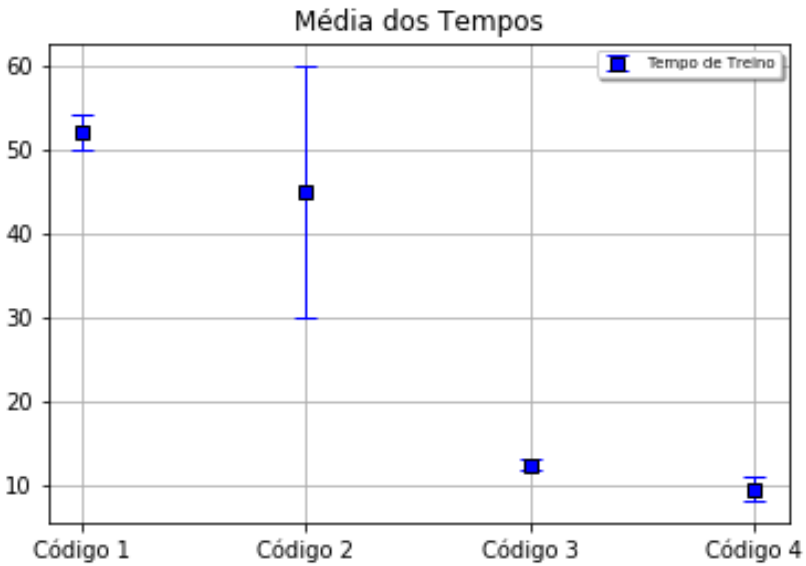

Figura 19: Gráfico Média dos Tempos KNN

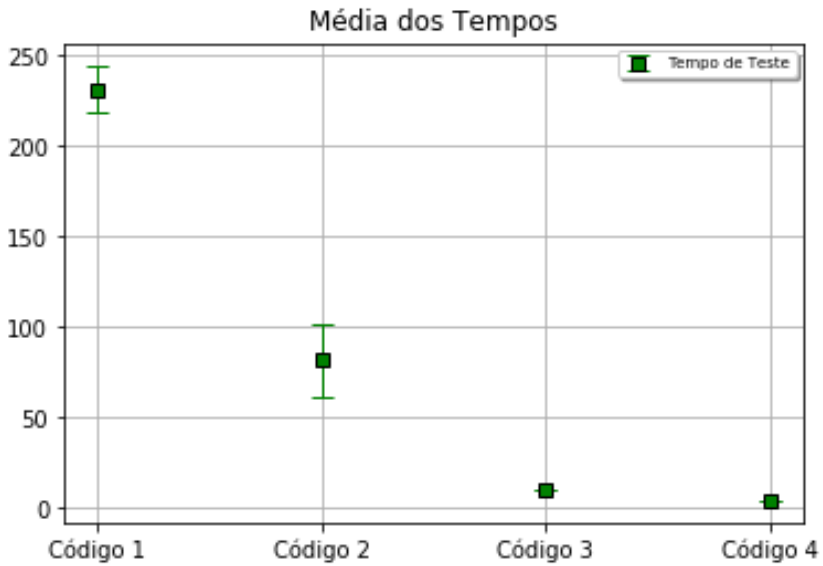

Figura 20: Gráfico Média dos Tempos KNN

- Código 1: One-Hot + Normalização
- Código 2: Normalização
- Código 3: One-Hot
- Código 4: Sem One-Hot e Normalização

Resultados

- K-Nearest Neighbour (KNN)

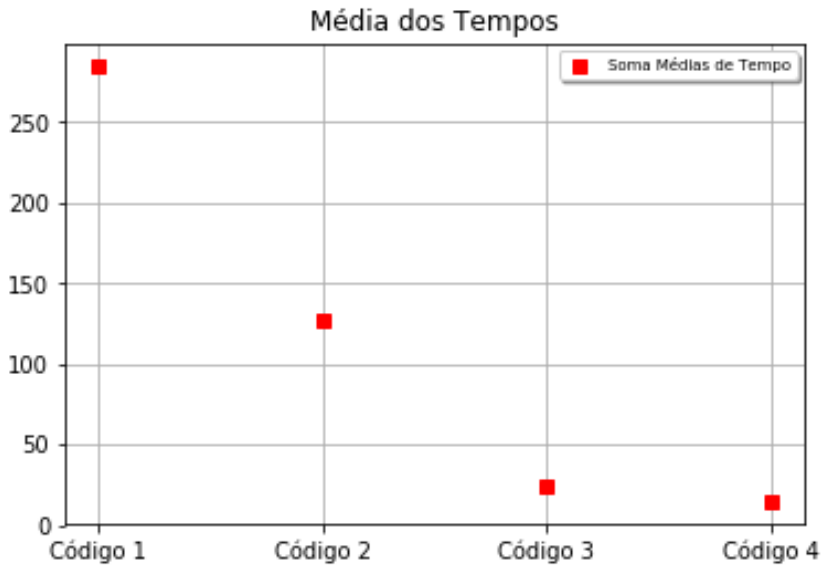

Figura 21: Gráfico Média dos Tempos KNN

- Código 1: One-Hot + Normalização
- Código 2: Normalização
- Código 3: One-Hot
- Código 4: Sem One-Hot e Normalização

Resultados

- Melhores Resultados

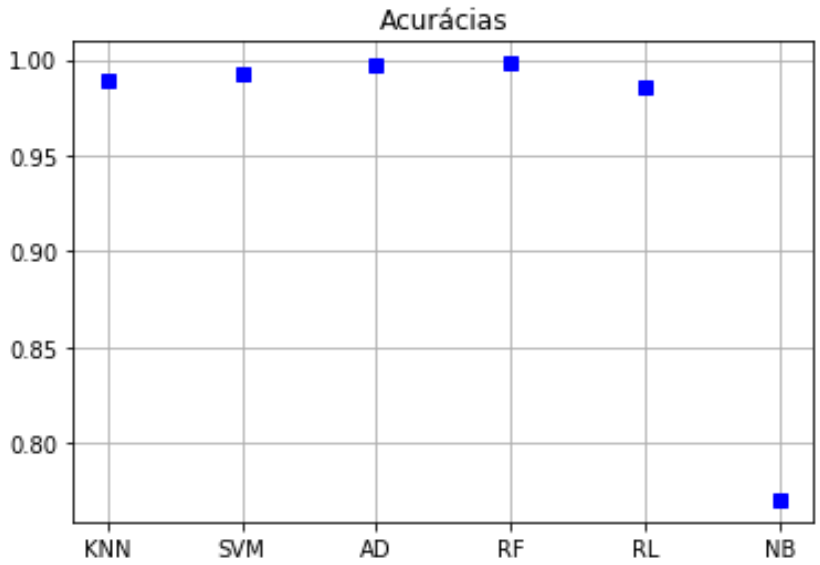

Figura 22: Gráfico Acurácias

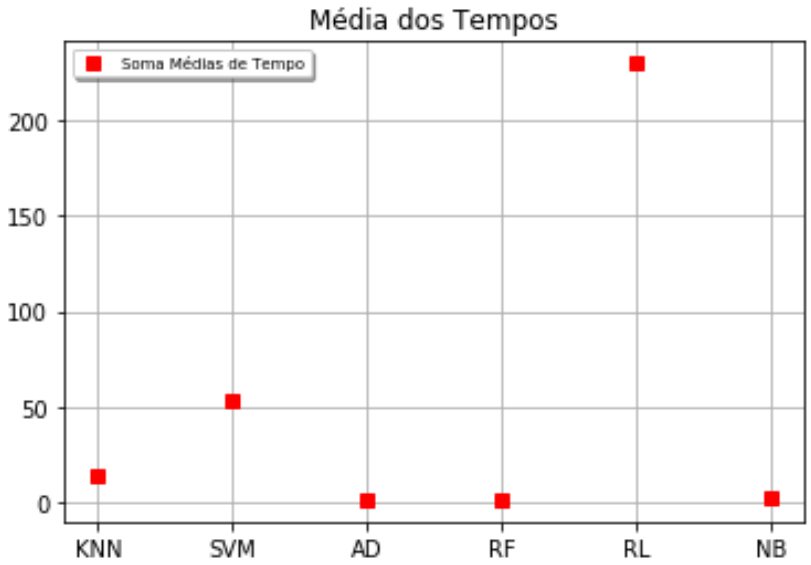

Figura 23: Gráfico Tempos

Condições das Melhores Acurácias:

- KNN: Sem Normalização + One-Hot
- SVM: Normalização
- NB: Normalização + One-Hot
- AD: Normalização
- RF: Normalização
- RL: Normalização + One-Hot

Resultados

- Melhores Resultados

	Md. Acurácia	Tm. de Pré-P.	Tm. de Treino	Tm. de Teste	Soma dos Tms.
AD	0.997713	0.745769	0.969451	0.006002	1.721223
RF	0.998052	0.703075	0.590857	0.028117	1.322050
KNN	0.989320	0.306482	9.582092	3.994016	13.882591
NB	0.770522	1.234288	0.399970	0.759322	2.393581
SVM	0.992305	0.777155	47.118665	5.899022	53.794843
RL	0.985594	1.246486	228.682299	0.012503	229.941289

Tabela 2: Resultados Finais

Condições das Melhores Acurácias:

- KNN: Sem Normalização + One-Hot
- SVM: Normalização
- NB: Normalização + One-Hot
- AD: Normalização
- RF: Normalização
- RL: Normalização + One-Hot

Considerações Finais

- Alterações no Código
- Implementação em um Cenário IoT

Referências I

[1] Kaspersky Lab. Aumentou mais do que duas vezes o número de malware visando dispositivos inteligentes em 2017.
`https://www.kaspersky.com.br/about/press-releases/2017_aumentou-mais-do-que-duas-vezes-o-numero-de-malware-\visando-dispositivos-inteligentes-em-2017`.
Acesso: 07/10/2018.

[2] Rebecca Bace and Peter Mell.
Nist special publication on intrusion detection systems.
Technical report, BOOZ-ALLEN AND HAMILTON INC MCLEAN VA, 2001.

[3] KDD-CUP-99.
`http://kdd.ics.uci.edu/databases/kddcup99/task.html`.
Acesso: 24/08/2018.

[4] Rana Aamir Raza Ashfaq, Xi-Zhao Wang, Joshua Zhexue Huang, Haider Abbas, and Yu-Lin He.
Fuzziness based semi-supervised learning approach for intrusion detection system.
Information Sciences, 378:484–497, 2017.

Referências II

[5] Documentation of scikit-learn.
`http://scikit-learn.org/stable/documentation.html`.
Acesso: 24/08/2018.

[6] Git Hub NSL-KDD dataset.
`https://github.com/thinline72/nsl-kdd`.
Acesso: 24/08/2018.

[7] Cristiano Antonio de Souza et al.
Método híbrido de detecção de intrusão aplicando inteligência artificial.
Universidade Estadual do Oeste do Paraná, 2018.