Análise Comparativa de Algoritmos de *Machine Learning* na Detecção de *Malwares* em Dispositivos *Android*

Lucas Ferreira[1]

[1]Mestrando em Engenharia Eletrônica e Computação
Universidade Católica de Pelotas (UCPel)

Trabalho de Introdução à Inteligência Computacional

Sumário

Sumário

Introdução

- A popularização do Sistema Operacional *Android* e o advento e onipresença dos dispositivos móveis em nosso cotidiano fez com que as ameaças de segurança se tornassem maiores nestes dispositivos;

Introdução

- A popularização do Sistema Operacional *Android* e o advento e onipresença dos dispositivos móveis em nosso cotidiano fez com que as ameaças de segurança se tornassem maiores nestes dispositivos;
- A detecção de *malware* é necessária, levando em conta os mais diversos tipos de ataques cibernéticos hoje existentes;

Introdução

- A popularização do Sistema Operacional *Android* e o advento e onipresença dos dispositivos móveis em nosso cotidiano fez com que as ameaças de segurança se tornassem maiores nestes dispositivos;
- A detecção de *malware* é necessária, levando em conta os mais diversos tipos de ataques cibernéticos hoje existentes;
- A forma como os aplicativos são distribuídos no *Android* influencia na forma de infecção do SO;

Introdução

- A popularização do Sistema Operacional *Android* e o advento e onipresença dos dispositivos móveis em nosso cotidiano fez com que as ameaças de segurança se tornassem maiores nestes dispositivos;
- A detecção de *malware* é necessária, levando em conta os mais diversos tipos de ataques cibernéticos hoje existentes;
- A forma como os aplicativos são distribuídos no *Android* influencia na forma de infecção do SO;
- Algoritmos de *Machine Learning* são explorados na detecção destes códigos, utilizando para tal dados estáticos e/ou dinâmicos extraídos destes aplicativos.

Sumário

Metodologia

- Avaliação de quatro algoritmos de *Machine Learning* com o propósito de classificação;

Metodologia

- Avaliação de quatro algoritmos de *Machine Learning* com o propósito de classificação;
- Foram computados a avaliação no conjunto de treino e teste a acurácia, o Score F1 e a curva ROC;

Metodologia

- Avaliação de quatro algoritmos de *Machine Learning* com o propósito de classificação;
- Foram computados a avaliação no conjunto de treino e teste a acurácia, o Score F1 e a curva ROC;
- Os algoritmos utilizados foram:
  - Árvores de decisão (Decision Tree);

Metodologia

- Avaliação de quatro algoritmos de *Machine Learning* com o propósito de classificação;
- Foram computados a avaliação no conjunto de treino e teste a acurácia, o Score F1 e a curva ROC;
- Os algoritmos utilizados foram:
  - Árvores de decisão (Decision Tree);
  - Suporte de máquina de vetores (SVM);

Metodologia

- Avaliação de quatro algoritmos de *Machine Learning* com o propósito de classificação;
- Foram computados a avaliação no conjunto de treino e teste a acurácia, o Score F1 e a curva ROC;
- Os algoritmos utilizados foram:
  - Árvores de decisão (Decision Tree);
  - Suporte de máquina de vetores (SVM);
  - K-ésimo vizinho mais próximo (KNN); e

Metodologia

- Avaliação de quatro algoritmos de *Machine Learning* com o propósito de classificação;
- Foram computados a avaliação no conjunto de treino e teste a acurácia, o Score F1 e a curva ROC;
- Os algoritmos utilizados foram:
  - Árvores de decisão (Decision Tree);
  - Suporte de máquina de vetores (SVM);
  - K-ésimo vizinho mais próximo (KNN); e
  - Floresta aleatória (Random Forest).

Fluxograma de desenvolvimento

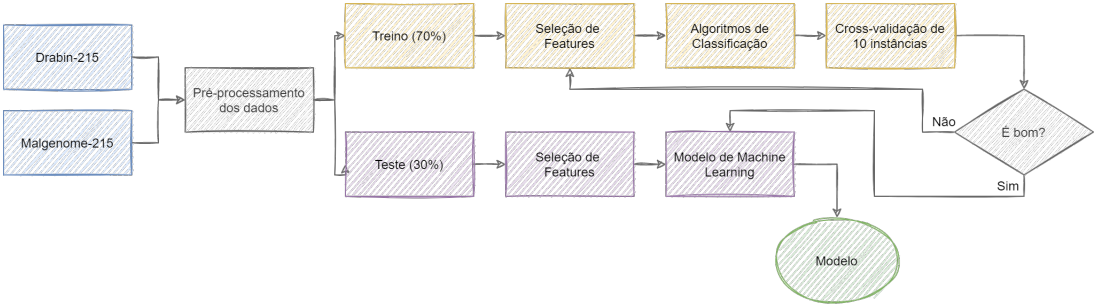

Figura 1: Fluxograma de desenvolvimento

Sumário

Árvores de decisão (Decision Tree)

- É um método usado em problemas de classificação e regressão, que particiona recursivamente um conjunto de treinamento, até que cada subconjunto resultante apresente casos de uma única classe;

Árvores de decisão (Decision Tree)

- É um método usado em problemas de classificação e regressão, que particiona recursivamente um conjunto de treinamento, até que cada subconjunto resultante apresente casos de uma única classe;
- O algoritmo examina e compara as distribuições de classes;

Árvores de decisão (Decision Tree)

- É um método usado em problemas de classificação e regressão, que particiona recursivamente um conjunto de treinamento, até que cada subconjunto resultante apresente casos de uma única classe;
- O algoritmo examina e compara as distribuições de classes;
- Os algoritmos utilizam a técnica de dividir para conquistar e o modelo Top-Down, ou seja, inicia do nó da raiz e vai até as suas folhas;

Árvores de decisão (Decision Tree)

- É um método usado em problemas de classificação e regressão, que particiona recursivamente um conjunto de treinamento, até que cada subconjunto resultante apresente casos de uma única classe;
- O algoritmo examina e compara as distribuições de classes;
- Os algoritmos utilizam a técnica de dividir para conquistar e o modelo Top-Down, ou seja, inicia do nó da raiz e vai até as suas folhas;
- É geralmente usada quando as instâncias são representadas por pares atributo-valor, a função objetivo assume apenas valores discretos, o conjunto de treinamento pode estar corrompido por ruído, etc.

Suporte de máquina de vetores (SVM)

- É um método de aprendizagem para problemas de reconhecimento de padrões;

Suporte de máquina de vetores (SVM)

- É um método de aprendizagem para problemas de reconhecimento de padrões;
- Foi introduzida por Vapnik (1995) através da teoria estatística de aprendizagem, no qual utiliza do fundamento de separação ótima entre classes, onde se as classes são separáveis, então o resultado é obtido de modo a separar o máximo as classes;

Suporte de máquina de vetores (SVM)

- É um método de aprendizagem para problemas de reconhecimento de padrões;
- Foi introduzida por Vapnik (1995) através da teoria estatística de aprendizagem, no qual utiliza do fundamento de separação ótima entre classes, onde se as classes são separáveis, então o resultado é obtido de modo a separar o máximo as classes;
- Sua proposta esta em resolver problemas de classificação e análise de regressão;

Suporte de máquina de vetores (SVM)

- É um método de aprendizagem para problemas de reconhecimento de padrões;
- Foi introduzida por Vapnik (1995) através da teoria estatística de aprendizagem, no qual utiliza do fundamento de separação ótima entre classes, onde se as classes são separáveis, então o resultado é obtido de modo a separar o máximo as classes;
- Sua proposta esta em resolver problemas de classificação e análise de regressão;
- Funciona muito bem com margem de separação clara, é eficaz nos casos em que o número de dimensões é maior que o número de amostras, além de ser eficiente em memória;

Suporte de máquina de vetores (SVM)

- É um método de aprendizagem para problemas de reconhecimento de padrões;
- Foi introduzida por Vapnik (1995) através da teoria estatística de aprendizagem, no qual utiliza do fundamento de separação ótima entre classes, onde se as classes são separáveis, então o resultado é obtido de modo a separar o máximo as classes;
- Sua proposta esta em resolver problemas de classificação e análise de regressão;
- Funciona muito bem com margem de separação clara, é eficaz nos casos em que o número de dimensões é maior que o número de amostras, além de ser eficiente em memória;
- Não tem bom desempenho quando há um grande conjunto de dados porque o tempo de treinamento é grande e também quando o conjunto de dados é ruidoso.

K-ésimo vizinho mais próximo (KNN)

- É um popular algoritmo de aprendizado não-supervisionado que gera k agrupamentos a partir de um conjunto de dados de treino;

K-ésimo vizinho mais próximo (KNN)

- É um popular algoritmo de aprendizado não-supervisionado que gera k agrupamentos a partir de um conjunto de dados de treino;
- A saída gerada pelo KNN pode ser vista como um diagrama de Vonoroi, que nada mais é que um particionamento do conjunto de dados com alguns pontos centrais;

K-ésimo vizinho mais próximo (KNN)

- É um popular algoritmo de aprendizado não-supervisionado que gera k agrupamentos a partir de um conjunto de dados de treino;
- A saída gerada pelo KNN pode ser vista como um diagrama de Vonoroi, que nada mais é que um particionamento do conjunto de dados com alguns pontos centrais;
- No seu funcionamento, para descobrir a classe de um elemento que não pertença ao conjunto de dados de treinamento, o KNN busca K elementos do conjunto que estejam mais próximos do elemento desconhecido, isto é, que tenham a menor distância;

K-ésimo vizinho mais próximo (KNN)

- É um popular algoritmo de aprendizado não-supervisionado que gera k agrupamentos a partir de um conjunto de dados de treino;
- A saída gerada pelo KNN pode ser vista como um diagrama de Vonoroi, que nada mais é que um particionamento do conjunto de dados com alguns pontos centrais;
- No seu funcionamento, para descobrir a classe de um elemento que não pertença ao conjunto de dados de treinamento, o KNN busca K elementos do conjunto que estejam mais próximos do elemento desconhecido, isto é, que tenham a menor distância;
- O classificador KNN possui o numero de K-vizinhos como único parâmetro livre, o qual é controlado pelo usuário para obter uma melhor classificação;

K-ésimo vizinho mais próximo (KNN)

- É um popular algoritmo de aprendizado não-supervisionado que gera k agrupamentos a partir de um conjunto de dados de treino;
- A saída gerada pelo KNN pode ser vista como um diagrama de Vonoroi, que nada mais é que um particionamento do conjunto de dados com alguns pontos centrais;
- No seu funcionamento, para descobrir a classe de um elemento que não pertença ao conjunto de dados de treinamento, o KNN busca K elementos do conjunto que estejam mais próximos do elemento desconhecido, isto é, que tenham a menor distância;
- O classificador KNN possui o numero de K-vizinhos como único parâmetro livre, o qual é controlado pelo usuário para obter uma melhor classificação;
- Ele pode ser exaustivo computacionalmente se utilizado um conjunto com muitos dados.

Floresta aleatória (Random Forest)

- É um metodo desenvolvido por Breiman (2001) para resolver problemas de classificação e regressão de métodos de aprendizagem em arvores, por meio do bootstrap dos dados de treinamento, aumentando o uso do algoritmo conhecido como CART (classification e Regression Trees);

Floresta aleatória (Random Forest)

- É um metodo desenvolvido por Breiman (2001) para resolver problemas de classificação e regressão de métodos de aprendizagem em arvores, por meio do bootstrap dos dados de treinamento, aumentando o uso do algoritmo conhecido como CART (classification e Regression Trees);
- Na grande maioria, os métodos de decisão em arvore possuem viés baixo, mas alta variância, resultando na produção de um sobre ajuste (overfitting);

Floresta aleatória (Random Forest)

- É um metodo desenvolvido por Breiman (2001) para resolver problemas de classificação e regressão de métodos de aprendizagem em arvores, por meio do bootstrap dos dados de treinamento, aumentando o uso do algoritmo conhecido como CART (classification e Regression Trees);
- Na grande maioria, os métodos de decisão em arvore possuem viés baixo, mas alta variância, resultando na produção de um sobre ajuste (overfitting);
- A precisão do classificador de RF é muito boa, possui forte capacidade de generalização, tem alta velocidade computacional e os parâmetros configuráveis são muito poucos.

Sumário

Resultados

Dataset	# de Instâncias	# de Infectados	# de Benignos	# de Features
Drebin-215	15,036	5,560	9,476	215
Malgenome-215	3,799	1,260	2,539	215

Tabela 1: Características dos datasets selecionados.

Resultados

- Os datasets foram unidos por possuírem features semelhantes, gerando um único com 18,830 instâncias combinadas;

Resultados

- Os datasets foram unidos por possuírem features semelhantes, gerando um único com 18,830 instâncias combinadas;
- As features que não eram comuns entre os dois datasets foram descartadas;

Resultados

- Os datasets foram unidos por possuírem features semelhantes, gerando um único com 18,830 instâncias combinadas;
- As features que não eram comuns entre os dois datasets foram descartadas;
- As instâncias em branco foram removidas;

Resultados

- Os datasets foram unidos por possuírem features semelhantes, gerando um único com 18,830 instâncias combinadas;
- As features que não eram comuns entre os dois datasets foram descartadas;
- As instâncias em branco foram removidas;
- Dos dados, 12,015 são malwares (63,8%) e 6,815 são limpos (36,2%);

Resultados

- Os datasets foram unidos por possuírem features semelhantes, gerando um único com 18,830 instâncias combinadas;
- As features que não eram comuns entre os dois datasets foram descartadas;
- As instâncias em branco foram removidas;
- Dos dados, 12,015 são malwares (63,8%) e 6,815 são limpos (36,2%);
- Alguns itens foram convertidos para conclusões binárias (0 ou 1);

Resultados

- Os datasets foram unidos por possuírem features semelhantes, gerando um único com 18,830 instâncias combinadas;
- As features que não eram comuns entre os dois datasets foram descartadas;
- As instâncias em branco foram removidas;
- Dos dados, 12,015 são malwares (63,8%) e 6,815 são limpos (36,2%);
- Alguns itens foram convertidos para conclusões binárias (0 ou 1);
- 208 features em comum na união dos datasets;

Resultados

- Os datasets foram unidos por possuírem features semelhantes, gerando um único com 18,830 instâncias combinadas;
- As features que não eram comuns entre os dois datasets foram descartadas;
- As instâncias em branco foram removidas;
- Dos dados, 12,015 são malwares (63,8%) e 6,815 são limpos (36,2%);
- Alguns itens foram convertidos para conclusões binárias (0 ou 1);
- 208 features em comum na união dos datasets;
- O resultado final foi dividido em datasets de treino (70%) e teste (30%), mantendo a % de distribuição das classes.

Resultados

- Os datasets foram unidos por possuírem features semelhantes, gerando um único com 18,830 instâncias combinadas;
- As features que não eram comuns entre os dois datasets foram descartadas;
- As instâncias em branco foram removidas;
- Dos dados, 12,015 são malwares (63,8%) e 6,815 são limpos (36,2%);
- Alguns itens foram convertidos para conclusões binárias (0 ou 1);
- 208 features em comum na união dos datasets;
- O resultado final foi dividido em datasets de treino (70%) e teste (30%), mantendo a % de distribuição das classes.

Resultados

- A partir do dataset resultante, foi feita a extração de features utilizando CfsSubsetEval;

Resultados

- A partir do dataset resultante, foi feita a extração de features utilizando CfsSubsetEval;
- Avalia o valor de um subconjunto de atributos, considerando a capacidade individual de cada feature junto com o grau de redundância entre eles.

Resultados

- A partir do dataset resultante, foi feita a extração de features utilizando CfsSubsetEval;
- Avalia o valor de um subconjunto de atributos, considerando a capacidade individual de cada feature junto com o grau de redundância entre eles.
- Foram selecionadas 27 features consideradas importantes;

Resultados

- A partir do dataset resultante, foi feita a extração de features utilizando CfsSubsetEval;
- Avalia o valor de um subconjunto de atributos, considerando a capacidade individual de cada feature junto com o grau de redundância entre eles.
- Foram selecionadas 27 features consideradas importantes;
- Foi observado um aumento de 0,15($\pm$0,14) para 0,35($\pm$0,14) e de 0,09($\pm$0,11) para 0,18($\pm$0,19) na correlação das features.

Resultados

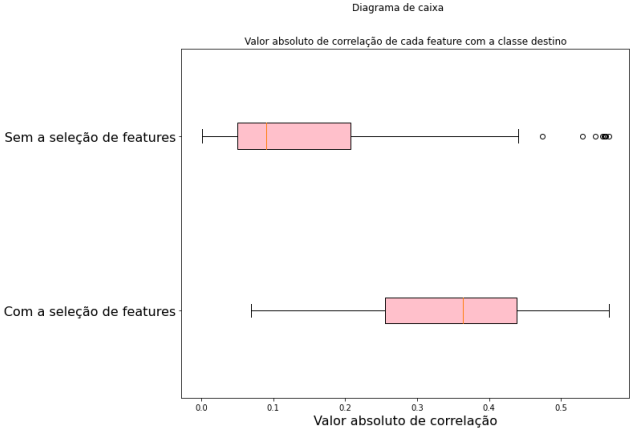

Figura 2: Gráfico de caixa da correlação absoluta de cada feature com a classe alvo

Resultados

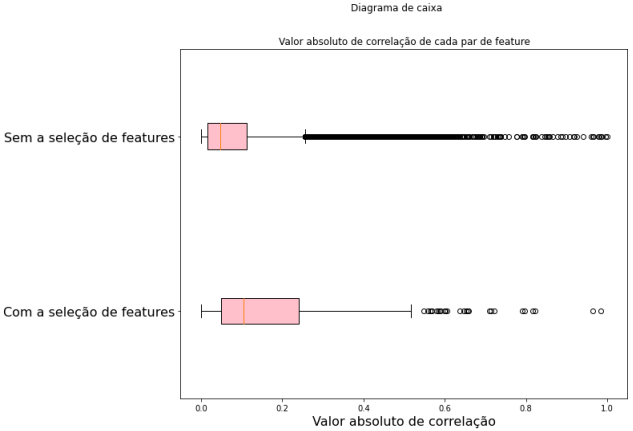

Figura 3: Gráfico de caixa da correlação absoluta de cada par de features

Resultados

- Os quatro algoritmos escolhidos foram aplicados com hiper parametrização para cada um deles e cross validação com 10-camadas;

Resultados

- Os quatro algoritmos escolhidos foram aplicados com hiper parametrização para cada um deles e cross validação com 10-camadas;
- Foi usada a acurácia, o score F1 e a curva ROC para avaliar a hiper parametrização;

Resultados

- Os quatro algoritmos escolhidos foram aplicados com hiper parametrização para cada um deles e cross validação com 10-camadas;
- Foi usada a acurácia, o score F1 e a curva ROC para avaliar a hiper parametrização;
- GridSearchCV foi usado para exaustivamente procurar entre um range definido de parâmetros e encontrar o melhor deles;

Resultados

- Os quatro algoritmos escolhidos foram aplicados com hiper parametrização para cada um deles e cross validação com 10-camadas;
- Foi usada a acurácia, o score F1 e a curva ROC para avaliar a hiper parametrização;
- GridSearchCV foi usado para exaustivamente procurar entre um range definido de parâmetros e encontrar o melhor deles;
- O resultado obtido foi levado em consideração no treino.

Resultados (Decision Tree)

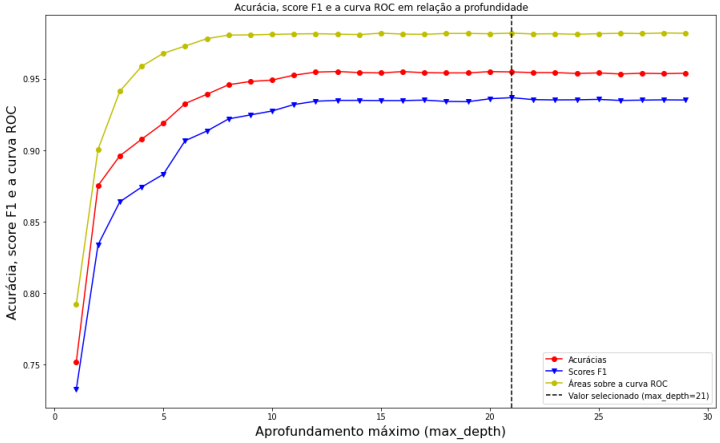

Figura 4: Gráfico de hiper parametrização da Decision Tree

Resultados (SVM)

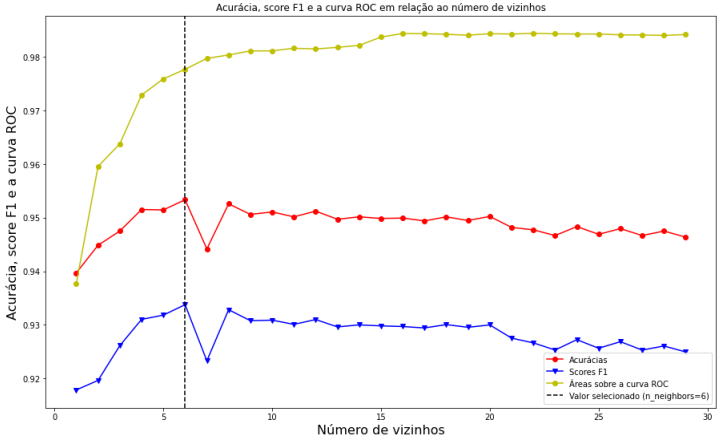

Figura 5: Gráfico de hiper parametrização da SVM

Resultados (KNN)

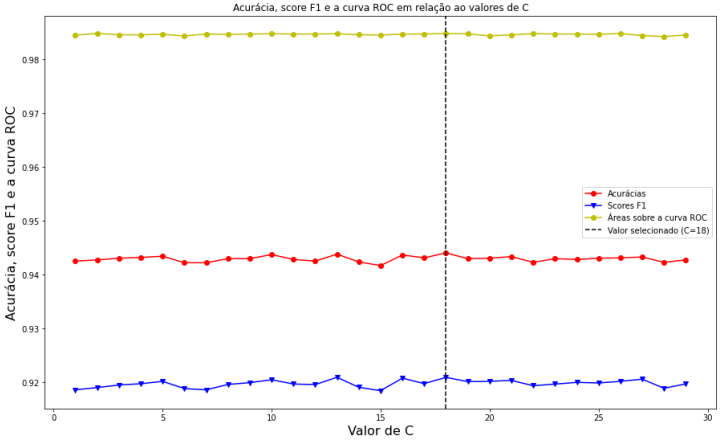

Figura 6: Gráfico de hiper parametrização da KNN

Resultados (Random Forest)

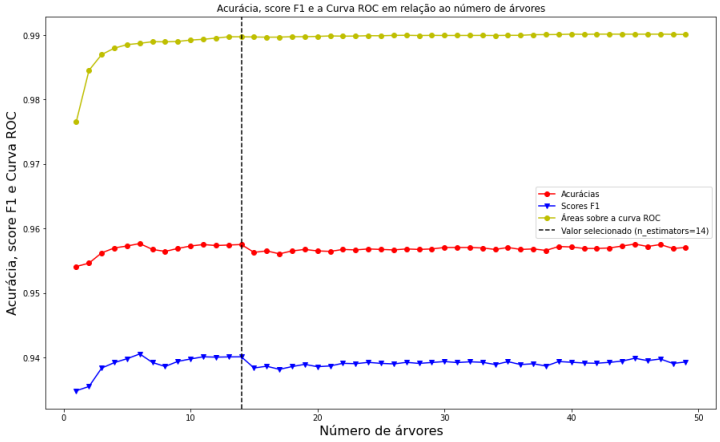

Figura 7: Gráfico de hiper parametrização da Random Forest

Resultados sem pré-processamento

	Acurácia	Score F1	Curva ROC
Decision Tree	**0,763**	0,758	0,795
KNN	0,742	0,741	0,786
SVM	0,751	0,743	0,777
Random Forest	0,762	**0,767**	**0,789**

Tabela 2: Resultados de treino de cada algoritmo sem pré-processamento

	Acurácia	Score F1	Curva ROC
Decision Tree	0,744	0,768	0,713
KNN	0,798	0,791	0,794
SVM	0,781	0,738	0,801
Random Forest	**0,803**	**0,793**	**0,812**

Tabela 3: Resultados de teste de cada algoritmo sem pré-processamento

Resultados com pré-processamento

	Acurácia	Score F1	Curva ROC
Decision Tree	**0,965**	**0,951**	**0,995**
KNN	0,956	0,938	0,986
SVM	0,946	0,923	0,985
Random Forest	**0,965**	**0,951**	0,994

Tabela 4: Resultados de treino de cada algoritmo com pré-processamento

	Acurácia	Score F1	Curva ROC
Decision Tree	**0,965**	**0,951**	**0,994**
KNN	0,956	0,938	0,986
SVM	0,945	0,924	0,984
Random Forest	0,962	0,948	0,991

Tabela 5: Resultados de teste de cada algoritmo com pré-processamento

Sumário

Considerações Finais

- O grande problema é como usar os algoritmos para resolver problemas difíceis;

Considerações Finais

- O grande problema é como usar os algoritmos para resolver problemas difíceis;
- A performance geral algoritmos depende do número de dados contidos no dataset;

Considerações Finais

- O grande problema é como usar os algoritmos para resolver problemas difíceis;
- A performance geral algoritmos depende do número de dados contidos no dataset;
- O uso do algoritmo correto pode fazer com que o desempenho em termos de poder computacional, uso de memória e tempo sejam reduzidos.

Trabalhos Futuros

- A comparação de outros parâmetros que não apenas os selecionados;

Trabalhos Futuros

- A comparação de outros parâmetros que não apenas os selecionados;
- Procurar outros trabalhos semelhantes e realizar a comparação dos resultados;

Trabalhos Futuros

- A comparação de outros parâmetros que não apenas os selecionados;
- Procurar outros trabalhos semelhantes e realizar a comparação dos resultados;
- Testar o modelo com parâmetros de versões mais recentes do sistema;

Trabalhos Futuros

- A comparação de outros parâmetros que não apenas os selecionados;
- Procurar outros trabalhos semelhantes e realizar a comparação dos resultados;
- Testar o modelo com parâmetros de versões mais recentes do sistema;
- Construir um analisador estático que possa analisar todas as chamadas de um APK sem que seja necessário um poder computacional grande.

Análise Comparativa de Algoritmos de *Machine Learning* na Detecção de *Malwares* em Dispositivos *Android*

Lucas Ferreira[1]

[1]Mestrando em Engenharia Eletrônica e Computação
Universidade Católica de Pelotas (UCPel)

Trabalho de Introdução à Inteligência Computacional