

---

Características inerentes a medidas de  
centralidade e uso de algoritmos de aprendizado  
de máquina para classificação de *bridging nodes*

---

Getúlio de Moraes Pereira



UNIVERSIDADE FEDERAL DE UBERLÂNDIA  
FACULDADE DE COMPUTAÇÃO  
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO



**Getúlio de Moraes Pereira**

**Características inerentes a medidas de  
centralidade e uso de algoritmos de aprendizado  
de máquina para classificação de *bridging nodes***

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Inteligência Artificial

Orientador: Prof. Dr. Anderson Rodrigues dos Santos

Uberlândia

2018

Dados Internacionais de Catalogação na Publicação (CIP)  
Sistema de Bibliotecas da UFU, MG, Brasil.

---

- P436c  
2018      Pereira, Getúlio de Moraes, 1976-  
Características inerentes a medidas de centralidade e uso de algoritmos de aprendizado de máquina para classificação de bridging nodes / Getúlio de Moraes Pereira. - 2018.  
135 f. : il.
- Orientador: Anderson Rodrigues dos Santos.  
Dissertação (mestrado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação.  
Disponível em: <http://dx.doi.org/10.14393/ufu.di.2018.287>  
Inclui bibliografia.
1. Computação - Teses. 2. Aprendizado do computador - Teses. 3. Algoritmos de computador - Teses. I. Santos, Anderson Rodrigues dos. II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. III. Título.

---

CDU: 681.3



UNIVERSIDADE FEDERAL DE UBERLÂNDIA  
FACULDADE DE COMPUTAÇÃO  
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Computação a aceitação da dissertação intitulada “**Características inerentes a medidas de centralidade e uso de algoritmos de aprendizado de máquina para classificação de *bridging nodes***” por **Getúlio de Moraes Pereira** como parte dos requisitos exigidos para a obtenção do título de **Mestre em Ciência da Computação**.

Uberlândia, \_\_\_\_ de \_\_\_\_\_ de \_\_\_\_

Orientador: \_\_\_\_\_

Prof. Dr. Anderson Rodrigues dos Santos  
Universidade Federal de Uberlândia

Banca Examinadora: \_\_\_\_\_

Profª. Dra. Márcia Aparecida Fernandes  
Universidade Federal de Uberlândia

\_\_\_\_\_  
Prof. Dr. Sérgio Vale Aguiar Campos  
Universidade Federal de Minas Gerais



*Dedico este trabalho primeiramente a meus pais, João e Delcina, pelo sacrifício e dedicação na minha formação; à minha esposa Vanessa pelo amor, amizade, paciência e cumplicidade e a meu filho Vinícius que veio a nascer durante o mestrado.*



---

# Agradecimentos

Agradeço primeiramente à Universidade Federal de Uberlândia e ao Programa de Pós Graduação em Ciência da Computação (PPGCO) da Faculdade de Computação (FACOM) por proporcionarem meu retorno à este ambiente acadêmico, no qual fiz parte há alguns anos, e que atualmente me trouxe ainda mais conhecimento e aprendizado.

Agradeço também ao Instituto Federal do Triângulo Mineiro (IFTM) por ter concedido o afastamento para poder terminar este mestrado. Agradeço aos professores do Curso de Ciência da Computação e Análise e Desenvolvimento de Sistemas do IFTM/Campus Ituiutaba por suprirem minha ausência, também pelo companheirismo e por entenderem a necessidade desta qualificação.

Agradeço ainda o Prof. Dr. Anderson Rodrigues dos Santos pela oportunidade e pela confiança em me orientar, seus ensinamentos me permitiram concluir esta etapa, bem como sua amizade e os puxões de orelha que me fizeram persistir nesta caminhada.

Agradeço à banca examinadora pelas correções, críticas e apontamentos que certamente contribuirão para a melhoria deste trabalho.

Aqui ressalto minha gratidão aos colegas que me acompanharam nas disciplinas, aos docentes que me ensinaram e que fortaleceram a certeza de que escolhi a profissão certa e ao Erisvaldo Araújo Fialho que sempre me auxiliou em relação aos trâmites do PPGCO.



*“Desistir é a saída dos fracos.  
Persistir é a alternativa dos fortes!”  
(Desconhecido)*





---

## Resumo

As redes de interação proteína-proteína (PPI), com frequência, comportam números expressivos de nós (proteínas) e arestas (interações) na ordem dos milhares, possivelmente milhões. Os nós promissores em redes PPI de grande porte, passíveis a serem utilizados na produção de fármacos, podem ser identificados através de métodos exatos como *bridging centrality*, no entanto, isto pode se tornar um problema computacional a ser superado devido à complexidade destas redes. Como alternativa, se sugere o uso de Inteligência Artificial, sendo o objetivo desta pesquisa analisar algoritmos de aprendizado de máquina (ML) aplicadas ao problema de determinação de *bridging centrality* em redes PPI, modeladas por meio de Rede Complexa, e identificar o algoritmo que ofereça resultados próximos ao gerado pelo algoritmo exato tido como referência, mas com esforço computacional menor. Foram selecionadas redes PPI de nove diferentes bactérias, sobre as quais foi gerado um conjunto de métricas estruturais usando o *software* Gephi. Em seguida, cada arquivo de PPI contendo as métricas geradas foi submetido à análise de 15 algoritmos selecionados para a ML, disponíveis no *software* WEKA. Os arquivos de métricas de predição foram submetidos ao melhor modelo preditivo identificado e, a seguir, os nós foram classificados em *weak* ou *strong*. Por fim, houve a avaliação do desempenho do classificador, utilizando-se o *software* R e o pacote ROCR, obtendo-se a curva ROC, o valor *Area Under the Curve* (AUC), a acurácia e o *threshold* correspondentes. Os melhores modelos de aprendizagem identificados foram gerados pelos algoritmos *Bagging* e *Random Forest*, e os piores foram gerados pelos algoritmos *Naïve Bayes* e *OneR*. Em termos gerais, a acurácia média da predição foi de  $74,38\% \pm 5,84\%$ , com limiar médio de 96%, e AUC médio de  $65,09\% \pm 4,48\%$ . Os nós preditos corretamente pelo classificador foram, em média, 24,33% sendo 2,75% verdadeiros positivos e 21,58% verdadeiros negativos. Por outro lado, 75,66% foram incorretamente classificados, sendo 23,67% falsos positivos e 51,99% falsos negativos. Comparando os 2,75% verdadeiros positivos com os identificados pelo algoritmo exato, obteve-se uma taxa de acerto médio de  $77,16\% \pm 20,23\%$ . O resultado preditivo gerado pelo processo de ML aproximou-se do obtido pelo algoritmo exato, apresentando eficácia, no entanto, com considerável taxa de erro. Dessa forma,

nossos resultados corroboram o conhecimento da literatura sobre uso de ML em redes complexas, ou seja, algoritmos de ML aplicados a medidas de centralidade complexas como a *bridging centrality* não são eficazes. O *plugin* implementado como um dos produtos deste trabalho para o *software* GEPHI, versão 0.9.1 ou superior, encontra disponível no site [sourceforge.net](https://sourceforge.net), sob o nome de BridgingCentralityPlugin.

**Palavras-chave:** PPI, Bridging Centrality, Aprendizado de Máquina, Redes Complexas.

---

# Abstract

Protein-protein interaction (PPI) networks often carry expressive numbers of proteins and interactions in the order of thousands, possibly millions. The promising nodes in large PPI networks, which can be used in drug production, can be identified through exact methods such as bridging centrality, however, this can become computationally expensive to be overcome due to the complexity of these networks. As an alternative, the use of machine learning is suggested. The objective of this study was analyzed machine learning algorithms (ML) applied to the problem of determining bridging centrality in PPI networks, modeled by Complex Network, and identify the algorithm which offers results closest to generated by the exact algorithm taken as a reference, but with less computational effort. PPI networks were selected from nine different bacteria, on which a set of structural metrics were generated using Gephi. Then, each PPI file containing the generated metrics was submitted to the analysis of 15 algorithms selected for ML available in the WEKA. The prediction metrics files were submitted to the best predictive model identified and then the nodes were classified as weak or strong. Finally, the performance of the classifier was evaluated using the R software and the ROCR package, obtaining the ROC curve, the area under ROC curve (AUC), the corresponding accuracy and threshold. The best learning models identified correspond to the algorithms Bagging and Random Forest and the worst were the NaiveBayes and OneR. In general terms, the mean prediction accuracy of the nodes of the PPIs was  $74.38\% \pm 5.84\%$ , with a mean threshold of 96%, and mean AUC of  $65.09\% \pm 4.48\%$ . The nodes correctly predicted by the classifier were, on average, 24.33%, with 2.75% true positive and 21.58% true negative. On the other hand, 75.66% were incorrectly classified, being 23,67% false positive and 51.99% false negative. Comparing the 2.75% true positive with those identified by the exact algorithm, an average hit rate of  $77.16\% \pm 20.23\%$  was obtained. The predictive result generated by the ML process approached that obtained by the exact algorithm, presenting efficacy, however, with a considerable error rate. Thus, our results corroborate the literature knowledge about the use of ML in complex networks, that is, ML algorithms applied to complex centrality measures such as bridging centrality are not effective. The

plugin implemented as one of the products of this work for Gephi software, version 0.9.1 or higher, is available on [sourceforge.net](https://sourceforge.net) under the name of BridgingCentralityPlugin.

**Keywords:** PPI, Bridging Centrality, Machine Learning, Complex Networks.

---

## Lista de ilustrações

Figura 1 – A rede metabólica da levedura com 359 nós e 435 arestas. Os nós com os maiores valores de <i>bridging centrality</i> são destacados em círculos azuis.	30
Figura 2 – Grafo representando o problema das 7 pontes de Königsburg.	34
Figura 3 – Exemplo esquemático de rede Protein-Protein Interaction (PPI) em sua participação num processo metabólico	34
Figura 4 – Exemplo de grafo com auto loop.	35
Figura 5 – Exemplo de grafo direcionado e sua matriz de adjacências.	36
Figura 6 – Representação de grafo ponderado direcionado e sua matriz de pesos.	37
Figura 7 – Exemplo de grafo não direcionado e sua matriz de adjacências.	37
Figura 8 – Exemplo de grafo não direcionado ponderado e sua matriz de pesos.	38
Figura 9 – Exemplo de grafo direcionado ponderado e sua matriz de pesos.	39
Figura 10 – Resumo esquemático de algoritmos de multiplicação de matrizes utilizados na contagem e listagem de triângulos presentes em redes complexas.	42
Figura 11 – Matriz de contingências e algumas métricas comuns.	48
Figura 12 – Exemplo de curva Receiver Operating Characteristic (ROC) obtida a partir de um conjunto de testes. A tabela ao lado corresponde aos dados gerados por um classificador em que se observa a classe predita e o score atribuído a cada instância. Os pontos no gráfico estão rotulados segundo o valor threshold correspondente.	48
Figura 13 – Exemplos de curvas ROC, onde <b>(a)</b> apresenta Area under ROC curve (AUC) para as curvas ROC A e B, respectivamente, e <b>(b)</b> apresenta curvas ROC de dois classificadores.	49
Figura 14 – Diagrama esquemático das etapas da pesquisa.	53
Figura 15 – Rede PPI do organismo <i>Corynebacterium diphtheriae</i> NC 002935 contendo 379 nós e 2.585 arestas, gerada pelo software Gephi.	55
Figura 16 – Rede PPI do organismo <i>Bacillus anthrax</i> contendo 792 nós e 19.518 arestas, gerada pelo software Gephi.	55
Figura 17 – Fragmento do arquivo de métricas exatas da PPI <i>Bacillus anthrax</i> .	59

Figura 18 – Curva ROC gerada pelo algoritmo <i>Random Forest</i> na fase de teste do treinamento aplicado à PPI <i>Bacillus anthrax</i> . . . . .	60
Figura 19 – Arquivo de log do treinamento do algoritmo <i>RandomForest</i> aplicado à PPI <i>Bacillus anthrax</i> . . . . .	61
Figura 20 – Fragmento do arquivo de métricas de predição correspondente à PPI <i>Bacillus anthrax</i> . . . . .	62
Figura 21 – Exemplo de gráficos de desempenho do modelo preditivo aplicado sobre uma PPI, onde: <b>(a)</b> mostra a curva ROC com valor da AUC, <b>(b)</b> exhibe a curva de acurácia em que se destaca a linha de corte e o valor de threshold, e <b>(c)</b> exhibe a proporção de verdadeiro positivos (TP), verdadeiro negativos (TN), falso positivos (FP) e falso negativos (FN) para o valor de threshold adotado . . . . .	64
Figura 22 – Resultado da geração de modelos de classificação da PPI <i>Corynebacterium diphtheriae NC 002935</i> , onde <b>(a)</b> apresenta os dados tabulados, e <b>(b)</b> representação gráfica em boxplot. . . . .	68
Figura 23 – Resultado da geração de modelos de classificação da PPI <i>Bacillus anthrax</i> , onde <b>(a)</b> apresenta os dados tabulados, e <b>(b)</b> representação gráfica em boxplot. . . . .	69
Figura 24 – Resultado da geração de modelos de classificação da PPI <i>Clostridium botulinum F Langeland</i> , onde <b>(a)</b> dados tabulados, e <b>(b)</b> representação gráfica em boxplot. . . . .	70
Figura 25 – Resultado da geração de modelos de classificação da PPI <i>Clostridium tetani E88</i> , onde <b>(a)</b> dados tabulados, e <b>(b)</b> representação gráfica em boxplot. . . . .	71
Figura 26 – Resultado da geração de modelos de classificação da PPI <i>Clostridium botulinum A2 Kyoto</i> , onde <b>(a)</b> dados tabulados, e <b>(b)</b> representação gráfica em boxplot. . . . .	72
Figura 27 – Resultado da geração de modelos de classificação da PPI <i>Escherichia coli ED1a</i> , onde <b>(a)</b> dados tabulados, e <b>(b)</b> representação gráfica em boxplot. . . . .	73
Figura 28 – Resultado da geração de modelos de classificação da PPI <i>Streptococcus pneumoniae Taiwan19F14</i> , onde <b>(a)</b> dados tabulados, e <b>(b)</b> representação gráfica em boxplot. . . . .	74
Figura 29 – Resultado da geração de modelos de classificação da PPI <i>Clostridium perfringens</i> , onde <b>(a)</b> dados tabulados, e <b>(b)</b> representação gráfica em boxplot. . . . .	75
Figura 30 – Resultado da geração de modelos de classificação da PPI <i>Mycobacterium tuberculosis</i> , onde <b>(a)</b> dados tabulados, e <b>(b)</b> representação gráfica em boxplot. . . . .	76

Figura 31 – Gráfico consolidado de boxplots, agrupado por PPI, exibindo a dispersão dos resultados obtidos pelos algoritmos estudados. . . . .	78
Figura 32 – Desempenho do modelo preditivo aplicado sobre a PPI <i>Clostridium botullinum</i> A2 Kyoto, sendo que (a) apresenta a curva ROC com AUC = 0,736620, (b) exibe a curva de acurácia em que destaca o valor de corte threshold = 0,96, e (c) exibe o percentual de verdadeiro positivos (TP), verdadeiro negativos (TN), falso positivos (FP) e falso negativos (FN) para o valor de threshold = 0,96 . . . . .	79
Figura 33 – Desempenho do modelo preditivo aplicado sobre a PPI <i>Bacillus anthrax</i> , sendo que (a) apresenta a curva ROC com AUC = 0,695160, (b) exibe a curva de acurácia em que destaca o valor de corte threshold = 0,96, e (c) exibe o percentual de verdadeiro positivos (TP), verdadeiro negativos (TN), falso positivos (FP) e falso negativos (FN) para o valor de threshold = 0,96 . . . . .	80
Figura 34 – Desempenho do modelo preditivo aplicado sobre a PPI <i>Mycobacterium tuberculosis</i> , sendo que (a) apresenta a curva ROC com AUC = 0,662467, (b) exibe a curva de acurácia em que destaca o valor de corte threshold = 0,96, e (c) exibe o percentual de verdadeiro positivos (TP), verdadeiro negativos (TN), falso positivos (FP) e falso negativos (FN) para o valor de threshold = 0,96 . . . . .	81
Figura 35 – Desempenho do modelo preditivo aplicado sobre a PPI <i>Clostridium botullinum</i> F Langeland, sendo que (a) apresenta a curva ROC com AUC = 0,639140, (b) exibe a curva de acurácia em que destaca o valor de corte threshold = 0,96, e (c) exibe o percentual de verdadeiro positivos (TP), verdadeiro negativos (TN), falso positivos (FP) e falso negativos (FN) para o valor de threshold = 0,96 . . . . .	82
Figura 36 – Desempenho do modelo preditivo aplicado sobre a PPI <i>Clostridium perfringens</i> , sendo que (a) apresenta a curva ROC com AUC = 0,627972, (b) exibe a curva de acurácia em que destaca o valor de corte threshold = 0,96, e (c) exibe o percentual de verdadeiro positivos (TP), verdadeiro negativos (TN), falso positivos (FP) e falso negativos (FN) para o valor de threshold = 0,96 . . . . .	83
Figura 37 – Desempenho do modelo preditivo aplicado sobre a PPI <i>Clostridium tetani</i> E88, sendo que (a) apresenta a curva ROC com AUC = 0,619899, (b) exibe a curva de acurácia em que destaca o valor de corte threshold = 0,96, e (c) exibe o percentual de verdadeiro positivos (TP), verdadeiro negativos (TN), falso positivos (FP) e falso negativos (FN) para o valor de threshold = 0,96 . . . . .	84

- Figura 38 – Desempenho do modelo preditivo aplicado sobre a PPI *Streptococcus pneumoniae Taiwan19F14*, sendo que **(a)** apresenta a curva ROC com  $AUC = 0,618804$ , **(b)** exibe a curva de acurácia em que destaca o valor de corte  $\text{threshold} = 0,96$ , e **(c)** exibe o percentual de verdadeiro positivos (TP), verdadeiro negativos (TN), falso positivos (FP) e falso negativos (FN) para o valor de  $\text{threshold} = 0,96$  . . . . . 85
- Figura 39 – Desempenho do modelo preditivo aplicado sobre a PPI *Escherichia coli ED1a*, sendo que **(a)** apresenta a curva ROC com  $AUC = 0,606802$ , **(b)** exibe a curva de acurácia em que destaca o valor de corte  $\text{threshold} = 0,96$ , e **(c)** exibe o percentual de verdadeiro positivos (TP), verdadeiro negativos (TN), falso positivos (FP) e falso negativos (FN) para o valor de  $\text{threshold} = 0,96$  . . . . . 86
- Figura 40 – Resultado consolidado proporcional da predição em termos de verdadeiro positivos (TP), verdadeiro negativos (TN), falso positivos (FP) e falso negativos (FN) . . . . . 87



---

## Lista de tabelas

Tabela 1 – Exemplos de redes representando sistemas reais. . . . .	33
Tabela 2 – Complexidade dos algoritmos relacionados ao cálculo de métricas de centralidade . . . . .	44
Tabela 3 – Algoritmos Machine Learning (ML) selecionados para a pesquisa . . .	45
Tabela 3 – Algoritmos ML selecionados para a pesquisa . . . . .	46
Tabela 4 – Redes PPI adotadas para testes da pesquisa . . . . .	54
Tabela 5 – Fragmento do arquivo de métricas relacionado à PPI <i>Bacillus anthrax</i>	56
Tabela 6 – Métricas selecionadas para esta pesquisa . . . . .	57
Tabela 7 – Algoritmos ML selecionados para a pesquisa . . . . .	58
Tabela 8 – Valores estatísticos da métrica <i>Bridging Centrality</i> calculados a partir do arquivo de métricas da PPI <i>Bacillus anthrax</i> . . . . .	59
Tabela 9 – Fragmento do arquivo “RandomForest.csv”gerado pelo <i>script</i> de predição a partir do arquivo de métricas de predição Attribute-Relation File Format (ARFF) correspondente à PPI <i>Bacillus anthrax</i> . . . . .	63
Tabela 10 – Exemplo de comparativo Real vs Predição para o percentual de nós classificados como verdadeiros positivos (TP) . . . . .	65
Tabela 11 – Modelos de aprendizado das redes PPI, seus respectivos algoritmos geradores e os valores de área sob a curva ROC (AUC) . . . . .	77
Tabela 12 – Resultado consolidado da predição em termos de AUC, ACC. Threshold = 0,96 . . . . .	87
Tabela 13 – Comparativo Real vs Predição para os 2,75% dos nós classificados como verdadeiros positivos (TP) . . . . .	88
Tabela 14 – Tempo de execução dos algoritmos geradores das métricas, em milisegundos . . . . .	99
Tabela 15 – Tempo de execução dos scripts de otimização de parâmetros dos algoritmos ML . . . . .	100
Tabela 16 – Parâmetros testados para o algoritmo Bagging . . . . .	100
Tabela 17 – Parâmetros testados para o algoritmo J48 . . . . .	101

Tabela 18 – Parâmetros testados para o algoritmo JRip . . . . .	101
Tabela 19 – Parâmetros testados para o algoritmo KStar . . . . .	102
Tabela 20 – Parâmetros testados para o algoritmo LMT . . . . .	102
Tabela 21 – Parâmetros testados para o algoritmo MultiLayerPerceptron . . . . .	103
Tabela 22 – Parâmetros testados para o algoritmo OneR . . . . .	103
Tabela 23 – Parâmetros testados para o algoritmo PART . . . . .	104
Tabela 24 – Parâmetros testados para o algoritmo RandomCommittee . . . . .	104
Tabela 25 – Parâmetros testados para o algoritmo RandomForest . . . . .	104
Tabela 26 – Parâmetros testados para o algoritmo RandomSubSpace . . . . .	105
Tabela 27 – Parâmetros testados para o algoritmo RandomTree . . . . .	105
Tabela 28 – Parâmetros testados para o algoritmo REPTree . . . . .	106
Tabela 29 – Resultado da comparação Strong Real vs Strong Predito usando a PPI <i>Bacillus anthrax</i> como modelo preditivo para o classificador Random- Forest . . . . .	128
Tabela 30 – Resultado da comparação Strong Real vs Strong Predito usando a PPI <i>Clostridium botulinum A2 Kyoto</i> como modelo preditivo para o clas- sificador RandomForest . . . . .	128
Tabela 31 – Resultado da comparação Strong Real vs Strong Predito usando a PPI <i>Clostridium botulinum F Langeland</i> como modelo preditivo para o classificador RandomForest . . . . .	129
Tabela 32 – Resultado da comparação Strong Real vs Strong Predito usando a PPI <i>Clostridium perfringens</i> como modelo preditivo para o classificador RandomForest . . . . .	130
Tabela 33 – Resultado da comparação Strong Real vs Strong Predito usando a PPI <i>Clostridium tetani E88</i> como modelo preditivo para o classificador RandomForest . . . . .	130
Tabela 34 – Resultado da comparação Strong Real vs Strong Predito usando a PPI <i>Corynebacterium diphtheriae NC 002935</i> como modelo preditivo para o classificador RandomForest . . . . .	131
Tabela 35 – Resultado da comparação Strong Real vs Strong Predito usando a PPI <i>Escherichia coli ED1a</i> como modelo preditivo para o classificador Ran- domForest . . . . .	132
Tabela 36 – Resultado da comparação Strong Real vs Strong Predito usando a PPI <i>Mycobacterium tuberculosis</i> como modelo preditivo para o classificador RandomForest . . . . .	132
Tabela 37 – Resultado da comparação Strong Real vs Strong Predito usando a PPI <i>Streptococcus pneumoniae Taiwan19F14</i> como modelo preditivo para o classificador RandomForest . . . . .	133

---

## Lista de siglas

**ARFF** : Attribute-Relation File Format

**AUC** : Area under ROC curve

**CSV** : Comma-Separated Values

**FN** : False Negative

**FP** : False Positive

**ML** : Machine Learning

**PPI** : Protein-Protein Interaction

**ROC** : Receiver Operating Characteristic

**TN** : True Negative

**TP** : True Positive



---

## Lista de símbolos

$\mathcal{V}(G)$	Conjunto de vértices do grafo $G$
$\mathcal{E}(G)$	Conjunto de arestas do grafo $G$
$ \mathcal{X} $	Cardinalidade do conjunto $\mathcal{X}$
$N$	Número de vértices, $ \mathcal{V}(G) $
$M$	Número de arestas, $ \mathcal{E}(G) $
$W$	Matriz de pesos
$w_{ij}$	elemento da Matriz de pesos
$A$	Matriz de adjacências
$a_{ij}$	elemento da Matriz de adjacências
$k_i$	grau do vértice $i$
$k_i^{out}$	grau de saída do vértice $i$
$k_i^{in}$	grau de entrada do vértice $i$
$s_i$	força do vértice $i$
$\nu(i)$	conjunto dos vizinhos do vértice $i$
$  X  $	Soma dos elementos da matriz $X$
$\mathcal{N}(v)$	Vizinhos de um vértice
$\bar{k}$	Grau médio do grafo
$d(\mathcal{W})$	Distância de uma caminhada ou de um caminho
$d_{uv}$	Menor caminho entre dois vértices

$C_D(i)$	Degree Centrality
$C_D^w(i)$	Weighted Degree Centrality
$C_B(v)$	Betweenness Centrality
$BC(v)$	Bridging coefficient
$C_R(v)$	Bridging Centrality

---

# Sumário

<b>1</b>	<b>INTRODUÇÃO . . . . .</b>	<b>27</b>
<b>1.1</b>	<b>Motivação . . . . .</b>	<b>29</b>
<b>1.2</b>	<b>Objetivos e Desafios da Pesquisa . . . . .</b>	<b>30</b>
1.2.1	Objetivo Geral . . . . .	30
1.2.2	Objetivos Específicos . . . . .	30
<b>1.3</b>	<b>Hipótese . . . . .</b>	<b>31</b>
<b>1.4</b>	<b>Organização da Dissertação . . . . .</b>	<b>31</b>
<b>2</b>	<b>REVISÃO DA LITERATURA . . . . .</b>	<b>33</b>
<b>2.1</b>	<b>Grafos . . . . .</b>	<b>35</b>
2.1.1	Medidas de Centralidade . . . . .	40
2.1.2	Outras Medidas de Centralidade . . . . .	44
<b>2.2</b>	<b>Aprendizado de Máquina . . . . .</b>	<b>44</b>
<b>2.3</b>	<b>Avaliação de Classificadores - curva ROC e AUC . . . . .</b>	<b>47</b>
<b>2.4</b>	<b>Trabalhos Correlatos . . . . .</b>	<b>49</b>
<b>3</b>	<b>METODOLOGIA . . . . .</b>	<b>53</b>
<b>3.1</b>	<b>Fase 1: Algoritmo Exato . . . . .</b>	<b>54</b>
<b>3.2</b>	<b>Fase 2: Aprendizado de Máquina . . . . .</b>	<b>57</b>
<b>3.3</b>	<b>Fase 3: Predição . . . . .</b>	<b>62</b>
<b>3.4</b>	<b>Fase 4: Conclusão . . . . .</b>	<b>65</b>
<b>4</b>	<b>EXPERIMENTOS E ANÁLISE DOS RESULTADOS . . . . .</b>	<b>67</b>
<b>4.1</b>	<b>Aplicação do modelo preditivo . . . . .</b>	<b>78</b>
<b>4.2</b>	<b>Comparativo Real vs Estimado . . . . .</b>	<b>88</b>
<b>5</b>	<b>CONCLUSÕES . . . . .</b>	<b>89</b>
<b>5.1</b>	<b>Síntese dos Resultados . . . . .</b>	<b>89</b>
<b>5.2</b>	<b>Dificuldades Encontradas . . . . .</b>	<b>90</b>

5.3	Principais Contribuições . . . . .	90
5.4	Trabalhos Futuros . . . . .	91
REFERÊNCIAS . . . . .		93

## APÊNDICES 97

APÊNDICE A – OTIMIZAÇÃO DE PARÂMETROS . . . . .		99
A.1	Bagging . . . . .	100
A.2	J48 . . . . .	101
A.3	JRip . . . . .	101
A.4	KStar . . . . .	102
A.5	LMT . . . . .	102
A.6	MultiLayerPerceptron . . . . .	103
A.7	OneR . . . . .	103
A.8	PART . . . . .	104
A.9	RandomCommittee . . . . .	104
A.10	RandomForest . . . . .	104
A.11	RandomSubSpace . . . . .	105
A.12	RandomTree . . . . .	105
A.13	REPTree . . . . .	106
APÊNDICE B – SCRIPT GERADOR DE MODELOS PREDITI- VOS . . . . .		107
APÊNDICE C – SCRIPT CLASSIFICADOR . . . . .		119
APÊNDICE D – EXPERIMENTOS ADICIONAIS . . . . .		127
D.1	PPI: Bacillus anthrax . . . . .	127
D.2	PPI: Clostridium botulinum A2 Kyoto . . . . .	128
D.3	PPI: Clostridium botulinum F Langeland . . . . .	129
D.4	PPI: Clostridium perfringens . . . . .	129
D.5	PPI: Clostridium tetani E88 . . . . .	130
D.6	PPI: Corynebacterium diphtheriae NC 002935 . . . . .	131
D.7	PPI: Escherichia coli ED1a . . . . .	131
D.8	PPI: Mycobacterium tuberculosis . . . . .	132
D.9	PPI: Streptococcus pneumoniae Taiwan19F14 . . . . .	133



---

## Introdução

Conforme apontam Oliveira e Santos (2016), as proteínas são consideradas os blocos fundamentais de construção da vida, pois desempenham vários papéis em funções estruturais e enzimáticas de uma célula, por meio de interações proteicas. No entanto, a determinação do papel das proteínas é uma tarefa difícil de se realizar. Como exemplo desse fato tem-se que quase metade dos genes procariontes têm função ainda desconhecida (HANSON et al., 2010). Todavia, apesar da falta de informações quanto ao papel de novos genes, pode-se vislumbrar possíveis interações entre proteínas desconhecidas e proteínas conhecidas. Tal anotação poderia ser útil ao permitir a concentração de esforços em um conjunto particular de genes na tentativa de se entender os processos biológicos. Para alcançar este objetivo, usa-se certas características possíveis de evidência do gene, como por exemplo: evolução, vizinhança conservada, expressão, um processo biológico envolvido, função e componente celular (SNEL et al., 2000). Cada característica cria uma aresta entre os nós representando os genes em uma rede denominada Rede de Interação Proteína-Proteína (PPI). Deve-se notar que tal rede é apenas uma previsão automatizada, pois somente observando-se o genoma de um organismo não se pode garantir a expressão de seus genes previstos. Mesmo assim, é uma previsão valiosa para estudos da Biologia. Redes PPI foram desenvolvidas em vários organismos, permitindo a compreensão de numerosos processos biológicos. O conjunto de associações montado permite a investigação das singularidades proteicas estruturais e funcionais dentro das vias de interesse. Os mapas de PPI ajudaram a investigar novas proteínas relacionadas à doença (STELZL et al., 2005).

Redes de interação proteína-proteína são um ingrediente importante para a compreensão em nível de sistema de processos celulares. Essas redes podem ser utilizadas para filtrar e avaliar dados de genômica funcional e, também, para fornecer uma plataforma intuitiva para se anotar propriedades estruturais, funcionais e evolutivas de proteínas. Por meio da exploração das redes de interação previstas pode-se sugerir novos rumos para futuras investigações experimentais, além de fornecer previsões inter-espécies para o mapeamento de interações. (SCHWARTZ et al., 2008)

A identificação de proteínas alvo para a produção de fármacos, conforme Ananthasubramanian et al. (2012) e Liu et al. (2012), utiliza frequentemente a análise PPI (do inglês *Protein-Protein Interaction*) em que as relações de interação proteína-proteína consideram características como: (i) a proximidade de seus genes em um cromossomo; (ii) seus genes estão fundidos de modo a facilitar a produção e interação de seus produtos proteicos; (iii) evolução conjunta; (iv) são expressas conjuntamente; (v) existem no mesmo local subcelular; (vi) estão associados à mesma função molecular; e (vii) participam do mesmo processo biológico, como afirmam Consortium et al. (2004) e Mering et al. (2007).

Uma das potencialidades de PPIs é identificar quais seriam as consequências de se ativar/desativar a expressão de um determinado gene em um organismo. No entanto, as técnicas *in vitro* e *in vivo* demandam grande quantidade de recursos, produzindo apenas uma fração das PPIs de um organismo sob estudo, como afirma Braun (2012). Para Ananthasubramanian et al. (2012) e Larsen, Hamada e Gilbert (2012), uma alternativa é a predição *in silico* de PPIs, em que modelos computacionais de predição de PPIs podem ser reaproveitados entre organismos evolutivamente próximos. Esse resultado abre a perspectiva de reutilizar o conhecimento a respeito de organismos para os quais as PPIs sejam conhecidas para outros sobre os quais ainda não exista informação experimental. Tal análise baseia-se na hipótese de que as proteínas envolvidas em um mesmo processo biológico possuem algumas das características acima relatadas (i até vi) com grande similaridade. Nessa configuração, quando tais proteínas são representadas graficamente em um espaço multidimensional (por meio de uma abstração em Rede Complexa), estariam propensas a formarem aglomerados mais compactos do que aquelas que não pertencem ao mesmo processo biológico. Atualmente, existe uma gama de algoritmos para identificar tais aglomerados como, por exemplo, a *Clusterização Baseada na Densidade* e a *Clusterização Hierárquica* que apresentam boa acurácia preditiva em redes PPIs, como exposto por Zhang (2009).

As redes produzidas como consequência da geração de PPIs de um proteoma são complexas por possuírem uma quantidade relevante de caminhamentos circulares, tornando difícil perceber a origem e a finalização do possível processo biológico. Nas redes construídas a partir de milhares de proteínas a quantidade de ciclos aumenta consideravelmente, e a análise de suas interações mais simples tende a tornar-se inviável. Essa complexidade dificulta o uso de redes oriundas de PPIs para o mapeamento de relações menos óbvias, aquelas que não estão nas vizinhanças imediatas das proteínas, segundo Zhang (2009).

Para diminuir a complexidade da análise de redes PPI, Hwang et al. (2006) propõe a estatística *bridging centrality* a fim de definir um arcabouço central de vértices do PPI em estudo. Esse cálculo é o produto de duas partes: *bridging coefficient* e *betweenness centrality*, conforme Breitling et al. (2004) (detalhes no capítulo 2, seção 2.1.1.6) e a multiplicação dessa apuração garante uma normalização do item *bridging centrality* prevenindo, assim, a ocorrência de erros devido a diferenças de escala, segundo Zhang (2009). O valor

*bridging coefficient* de um dado vértice determina a extensão do quão bem posicionado ele está em relação a outros, de alto grau, ou seja, vértices que possuem elevadas quantidades de arestas. A estatística *betweenness centrality* leva em consideração a quantidade de caminhos mais curtos entre dois vértices quaisquer ponderando a quantidade desses caminhos que passam pelo vértice sob análise, como esclarece Zhang (2009). Essa estatística implica no mapeamento dos caminhos mais curtos entre todos os pares de vértices da rede. De acordo com Brandes (2001), tal análise, apesar de ser computacionalmente intensiva, pode ser executada em um tempo polinomial, com ordens de grandeza entre  $O(n^3)$  e  $O(nm)$  em grafos com arestas não direcionadas e não ponderadas, em que  $n$  representa a quantidade de vértices e  $m$  a quantidade de arestas. Uma vez computado o valor da estatística *bridging centrality*, todas as proteínas podem ser listadas em ordem decrescente de tal forma que os *bridging nodes* apresentem os maiores valores de *bridging centrality*.

O algoritmo apresentado por Brandes (2001) apresenta o cálculo exato<sup>1</sup> de *betweenness centrality* e requer espaço da ordem  $O(n + m)$ , e executa em tempo  $O(nm)$  para grafos não-ponderados e em  $O(nm + n^2 \log n)$  para grafos ponderados. Segundo Akhtar (2014), por apresentar tal desempenho, esse algoritmo encontra-se disponível em diversos *softwares* utilizados em análises de redes complexas como R<sup>2</sup>, Pajek<sup>3</sup>, Cytoscape<sup>4</sup>, SocNetV<sup>5</sup>, Gephi<sup>6</sup>, dentre outros.

## 1.1 Motivação

Com frequência as redes PPI apresentam elevado número de proteínas, podendo chegar aos milhares, assim como suas interações. Como exemplo, a Figura 1 apresenta uma rede PPI de levedura em que os *bridging nodes* são destacados por círculos pretos. Nesse sentido, o cálculo de *betweenness centrality* para grandes redes torna-se um problema computacional considerável.

Para contornar tal problema, tem-se como alternativa a utilização de técnicas de aprendizado de máquina (ML) sobre características estruturais da rede PPI em estudo, buscando-se prever quais de seus nós estariam mais propensos a serem classificados como *bridging nodes*.

Nesse sentido, este trabalho de pesquisa analisou a aplicação da técnica *bridging centrality* proposta por Hwang et al. (2006) em redes PPI, e também estudou a aplicação de algoritmos de aprendizado de máquina a fim de avaliar a viabilidade de utilizá-la como

<sup>1</sup> Por esta razão, pode ser chamado de *algoritmo exato*

<sup>2</sup> <http://www.r-project.org>, acessado em: 23/02/2017

<sup>3</sup> <http://mrvar.fdv.uni-lj.si/pajek/>, acesso em: 23/02/2017

<sup>4</sup> <http://www.cytoscape.org>, acesso em: 23/02/2017

<sup>5</sup> <http://socnetv.org>, acesso em: 23/02/2017

<sup>6</sup> <http://www.gephi.org>, acessado em: 23/02/2017

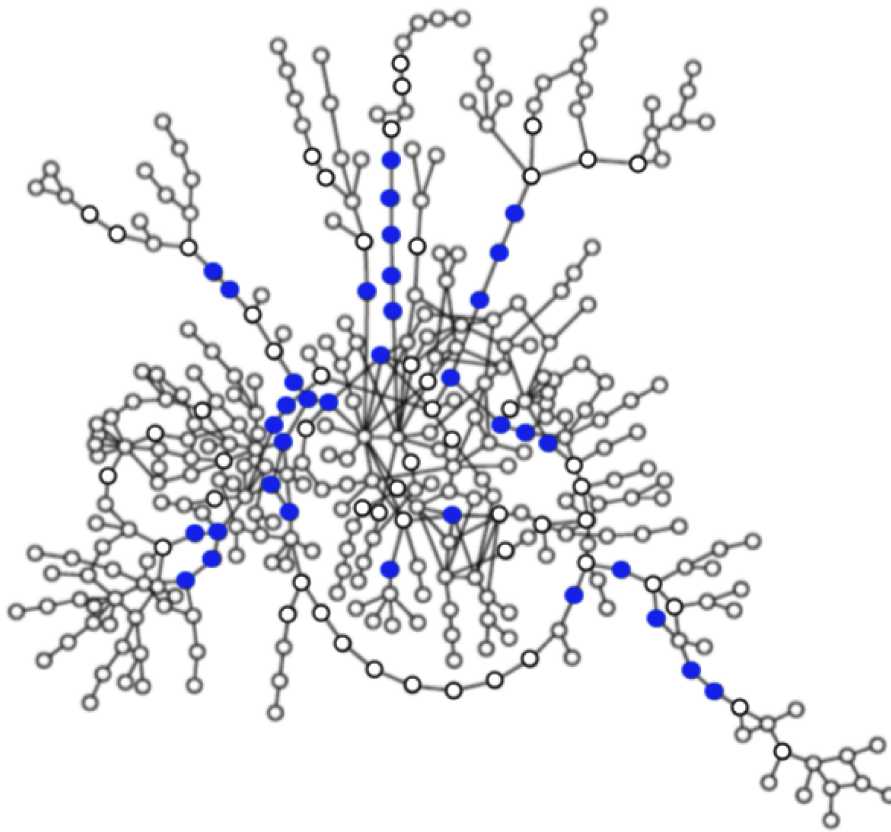


Figura 1 – A rede metabólica da levedura com 359 nós e 435 arestas. Os nós com os maiores valores de *bridging centrality* são destacados em círculos azuis.

Fonte: adaptado de Hwang et al. (2006)

método alternativo para predição de *bridging nodes* com esforço computacional menor que o do algoritmo exato.

## 1.2 Objetivos e Desafios da Pesquisa

### 1.2.1 Objetivo Geral

O objetivo geral deste trabalho foi analisar a aplicação de algoritmos de aprendizado de máquina ao problema de determinação de *bridging centrality* em redes PPI, modeladas por meio de Rede Complexa, e identificar aquele que ofereça resultados próximos ao do algoritmo exato tido como referência, mas com esforço computacional menor.

### 1.2.2 Objetivos Específicos

Para alcançar o objetivo geral, são apresentados os seguintes objetivos específicos:

- o realizar o levantamento de dados de redes PPI conhecidas para serem utilizados nos testes dos algoritmos;

- implementar o algoritmo exato de *bridging centrality* proposto por Hwang et al. (2006), na forma de um *plugin* para o *Gephi*;
- realizar testes com o algoritmo *bridging centrality* implementado (item anterior), coletando dados para análise posterior;
- para cada rede PPI adotada como objeto desse estudo:
  - ❑ gerar métricas para utilização nos algoritmos de aprendizado de máquina;
  - ❑ extrair as métricas cujos algoritmos tenham complexidade computacional inferiores à da *betweenness centrality*;
  - ❑ gerar o arquivo ARFF a ser submetido ao WEKA<sup>7</sup> para análise;
- utilizando o WEKA, executar algoritmos de aprendizado sobre as redes PPI, gerando os modelos de aprendizado correspondentes;
- com os modelos de aprendizado obtidos (item anterior), utilizar o WEKA para prever os nós mais promissores, para cada rede PPI do conjunto de testes;
- selecionar o modelo de aprendizado, assim como seu algoritmo, que apresentar o melhor resultado mensurado pelo valor AUC;
- implementar um *plugin* para o *Gephi* utilizando o algoritmo de aprendizado escolhido (item anterior);

### 1.3 Hipótese

A hipótese levantada para este trabalho é a existência de um algoritmo de aprendizado de máquina que, ao ser aplicado às métricas extraídas de uma rede PPI, gere resultados que se aproximem dos resultados gerados pelo algoritmo exato no cálculo de *bridging centrality*, mas com esforço computacional menor.

### 1.4 Organização da Dissertação

A apresentação do trabalho está estruturada como se segue.

O Capítulo 2 apresenta os conceitos teóricos que dão fundamento ao presente trabalho, com atenção especial a complexidade de algoritmos geradores de métricas de centralidade e uma breve análise de trabalhos correlatos.

O Capítulo 3 descreve a estrutura desta pesquisa, destacando-se o algoritmo exato para cálculo da *bridging centrality* e a implementação de seu plugin para o software Gephi.

<sup>7</sup> <https://www.cs.waikato.ac.nz/ml/weka/>, acesso em: 23/02/2017

Descreve, ainda, as etapas do estudo avaliativo dos algoritmos de aprendizado de máquina e a determinação do melhor método para predição de *bridging nodes*.

O Capítulo 4 apresenta os resultados obtidos pelos experimentos realizados utilizando os métodos descritos no capítulo 3.

Por fim, o Capítulo 5 apresenta as considerações finais desta pesquisa, com destaque para os principais resultados obtidos, relaciona dificuldades encontradas, contribuições para a literatura da área, aponta limitações da pesquisa e apresenta propostas para trabalhos futuros.

## Revisão da Literatura

De acordo com Dehmer e Basak (2012, p. 45), os fenômenos do mundo real ocorrem por meio de intrincadas interações entre vários de seus elementos. Para representá-los de forma abstrata, as redes são instrumentos que possibilitam descrever esses relacionamentos e são, assim, instrumentos poderosos para descrever sistemas complexos. Nesse sentido, o conceito de redes é universal e pode ser aplicado a estudos de áreas como Matemática, Ciência da Computação, Economia, Sociologia, Química, Física, Biologia, dentre outras. Logo, as redes tornaram-se um importante instrumento de compreensão dos sistemas do mundo real e da extração de conhecimento de suas relações. A Tabela 1 relaciona alguns exemplos de redes descrevendo sistemas reais.

Tabela 1 – Exemplos de redes representando sistemas reais.

Rede	Arestas	Nós
Internet	Conexão de dados	Computadores, roteadores, servidores
World Wide Web	Hyperlinks	Páginas web
Rede de citações	Citação	Artigos, patentes, or processos legais
Matriz energética	Linhas de transmissão	Estações e subestações geradoras
Redes sociais	Relacionamentos	Pessoas
Metabolismo	Reações metabólicas	Metabólitos
Cadeia alimentar	Predação	Espécies

Fonte: adaptado de Dehmer e Basak (2012, p. 46)

De acordo com Wang (2015), para que seja possível descrever as características comuns e as propriedades inerentes a diferentes tipos de redes, faz-se necessária uma ferramenta eficiente e igualmente rigorosa de análise, a qual é oferecida pela Matemática na forma da Teoria dos Grafos. Sua origem é atribuída a Leonhard Euler (1707 - 1783) quando este estudou e resolveu o famoso *Problema das Sete Pontes de Königsburg*. A Figura 2 representa a abstração utilizada por Euler.

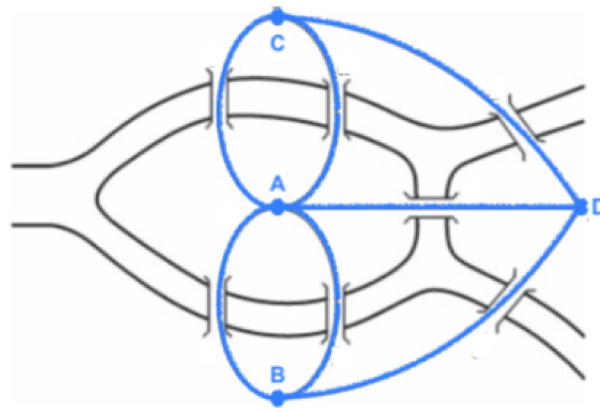


Figura 2 – Grafo representando o problema das 7 pontes de Königsburg.

Fonte: adaptado de Wang (2015, p. 6)

Nessa linha de raciocínio, como apresenta Wang (2015, p. 4), uma rede complexa é uma representação gráfica de um conjunto de elementos físicos diferentes (ou estruturas abstratas) representados por nós (vértices), interligados entre si por meio de *links* (arestas), segundo uma determinada regra de conexão, e formando uma certa estrutura topológica. Em contrapartida, um grafo, em sua representação pura, apresenta a noção matemática desse conceito considerando apenas sua estrutura, sem considerar seu significado físico.

Como mencionado no capítulo 1 e apontado na Tabela 1, interações entre proteínas (redes PPI) podem ser modeladas por redes complexas, cujos nós representam proteínas e os *links* descrevem as relações entre elas. Estas redes, de acordo com Erciyes (2014), atuam como interface entre o genoma e o metabolismo do organismo, como ilustra a Figura 3, e cumprem funções metabólicas sob controle dos genes.

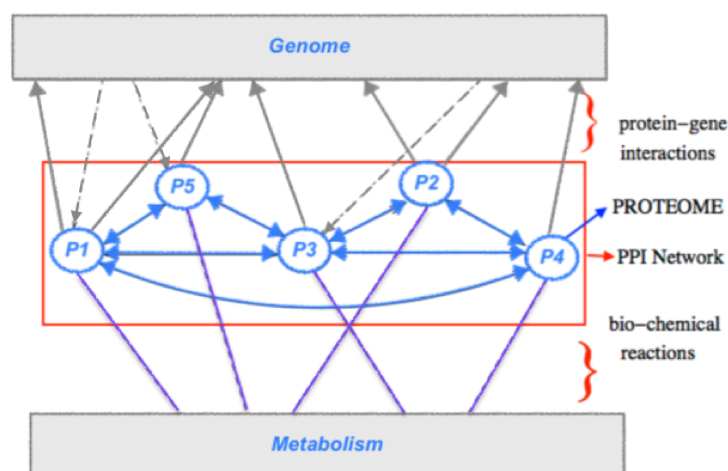


Figura 3 – Exemplo esquemático de rede PPI em sua participação num processo metabólico

Fonte: adaptado de Erciyes (2014, p. 204).



Na seção 2.1 a seguir são apresentados conceitos sobre grafos, relevantes ao escopo deste trabalho de pesquisa, tendo como referências a organização didática, assim como o conteúdo, apresentada por Silva e Zhao (2016, p. 16) e a notação adotada por Costa et al. (2007, p. 7). Os termos **grafo** e **rede** serão tratados, no decorrer do texto, como sinônimos e sua distinção será explicitada caso necessário.

## 2.1 Grafos

**Definição 2.1.1** (Grafo). *Um grafo  $\mathcal{G}$  é definido como um par ordenado  $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ , em que  $\mathcal{V}$  é um conjunto finito, não vazio de vértices (ou nós), e  $\mathcal{E}$  é o conjunto das arestas existentes entre os vértices de  $\mathcal{V}$ , ou seja,  $\mathcal{E} \subseteq \{(u, v) \mid u, v \in \mathcal{V}\}$ .*

A quantidade de vértices de  $\mathcal{G}$  é dada por  $n = |\mathcal{V}|$  e a quantidade de arestas, por  $m = |\mathcal{E}|$ .

Do grafo da Figura 4, tem-se  $\mathcal{V} = \{1, 2, 3, 4, 5\}$ ,  $\mathcal{E} = \{(1, 2), (1, 3), (2, 3), (3, 3), (3, 4)\}$ ,  $n = 5$  e  $m = 5$ , respectivamente.

Quando o grafo não possuir arestas ligando um mesmo vértice, ou seja,  $\forall u \in \mathcal{V} \mid (u, u) \notin \mathcal{E}$ , esse grafo é dito ser livre de auto loop. Informalmente, isso significa que não é possível sair de um vértice e voltar a ele mesmo passando por exatamente uma aresta. Por outro lado, quando houver pelo menos uma aresta ligando o mesmo vértice ( $\exists u \in \mathcal{V} \mid (u, u) \in \mathcal{E}$ ), o grafo é dito possuir auto loop. Este é o caso do grafo da Figura 4 que apresenta um loop no vértice 3.

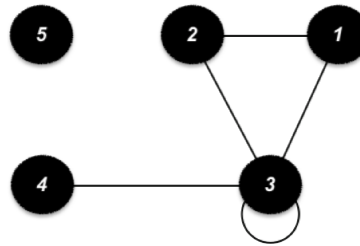


Figura 4 – Exemplo de grafo com auto loop.

Fonte: adaptado de Silva e Zhao (2016, p. 16)

**Definição 2.1.2** (Grafo não direcionado). *Quando o conjunto  $\mathcal{E}$  é simétrico, ou seja,  $\forall (u, v) \in \mathcal{E} \Rightarrow (v, u) \in \mathcal{E}$ , diz-se que  $\mathcal{G}$  é um grafo não direcionado.*

Em outras palavras, dada uma aresta entre os vértices  $u$  e  $v$ , deve haver uma aresta entre  $v$  e  $u$ . Graficamente, a existência dessas duas arestas ( $(u, v)$  e  $(v, u)$ ) é representada por única linha sem indicação de sentido (setas), como ilustra a Figura 4.

**Definição 2.1.3** (Grafo direcionado). *Quando o conjunto  $\mathcal{E}$  satisfaz a restrição  $\exists (u, v) \in \mathcal{E} \mid (v, u) \notin \mathcal{E}$ , diz-se que o grafo  $\mathcal{G}$  é direcionado. Esse grafo deve, pois, possuir pelo menos uma aresta de  $u$  para  $v$ , com a ausência de seu oposto.*

A Figura 5 ilustra um grafo cujas arestas possuem indicação de sentido (setas).

**Definição 2.1.4** (Vértices adjacentes). *Dois vértices  $u, v \in \mathcal{V}$  são ditos adjacentes se eles possuem uma aresta em comum.*

No caso de grafos não direcionados, se  $u$  é adjacente a  $v$  então  $v$  é adjacente a  $u$ . Mas, para grafos direcionados,  $u$  adjacente a  $v$  não implica que  $v$  seja adjacente a  $u$ , ou seja, se  $(u, v) \in \mathcal{E}$  e  $(v, u) \notin \mathcal{E}$ , então  $v$  é adjacente a  $u$ , mas o oposto não se aplica. Na Figura 5 o nó 3 é adjacente ao nó 5, mas o nó 5 não é adjacente a nó 3.

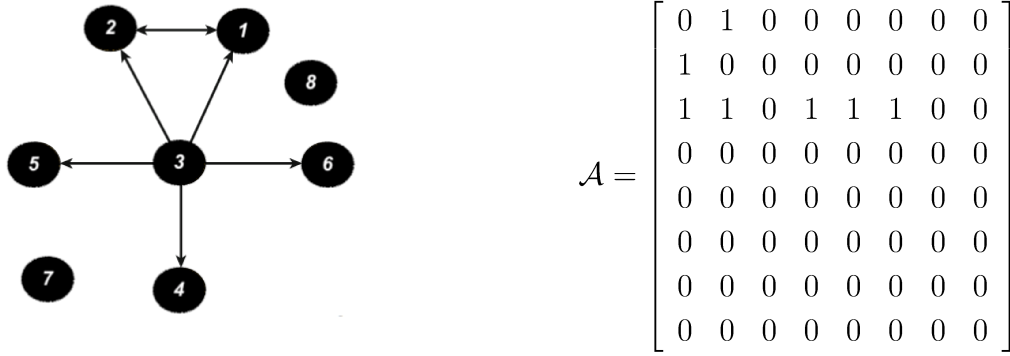


Figura 5 – Exemplo de grafo direcionado e sua matriz de adjacências.

Fonte: adaptado de Silva e Zhao (2016, p. 21)

**Definição 2.1.5** (Matriz de Adjacências). *Seja  $\mathcal{G} = \langle \mathcal{V}, \mathcal{E}, \mathcal{A} \rangle$  um grafo genérico.  $\mathcal{A}$  é denominada matriz de adjacências e é definida a seguir:*

- $\mathcal{A}$  é sempre uma matriz quadrada, em que a ordem  $n$  é dada pela quantidade de vértices de  $\mathcal{G}$  ( $n = |\mathcal{V}|$ )
- as linhas e colunas da matriz  $\mathcal{A}$  são indexadas pelos vértices de  $\mathcal{G}$
- a  $(i, j)$ -ésima entrada de  $\mathcal{A}$ , denotada por  $\mathcal{A}_{ij} = a_{i,j}$ , corresponde à aresta  $(i, j) \in \mathcal{E}$  do grafo. Formalmente,  $\forall (i, j) \in \mathcal{E} : a_{i,j} \neq 0$  e  $\forall (i, j) \notin \mathcal{E} : a_{i,j} = 0$

A matriz de adjacências assume a seguinte forma geral:

$$\mathcal{A} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

**Definição 2.1.6** (Grafo não ponderado). *Um grafo  $\mathcal{G}$  é dito não ponderado quando suas arestas não possuem indicação de peso.*

Para um grafo  $\mathcal{G}$  não ponderado, os elementos de sua matriz  $\mathcal{A}$  são 0's e 1's, ou seja,  $\mathcal{A}_{ij} \in \{0, 1\}, \forall i, j \in \mathcal{V}$ . Se  $\mathcal{G}$  é não direcionado, a matriz  $\mathcal{A}$  é simétrica ( $\mathcal{A}_{ij} = \mathcal{A}_{ji}$ ), o que pode não ocorrer quando  $\mathcal{G}$  é direcionado. A Figura 5 apresenta a matriz de adjacências  $\mathcal{A}$  correspondente ao grafo direcionado à sua esquerda. Cabe observar que os elementos dessa matriz não são simétricos em relação à sua diagonal principal e que os elementos dessa diagonal são nulos, indicado ausência de loop no grafo.

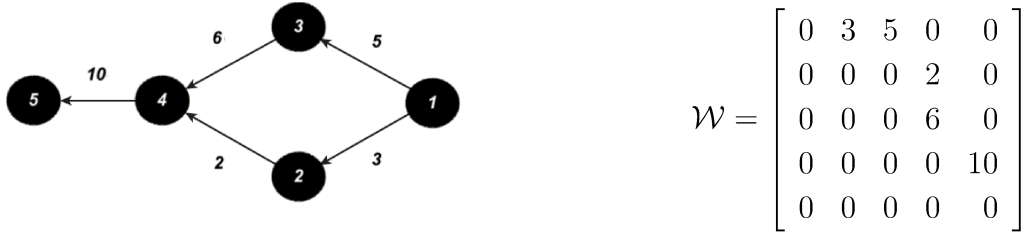


Figura 6 – Representação de grafo ponderado direcionado e sua matriz de pesos.

Fonte: adaptado de Silva e Zhao (2016, p. 18)

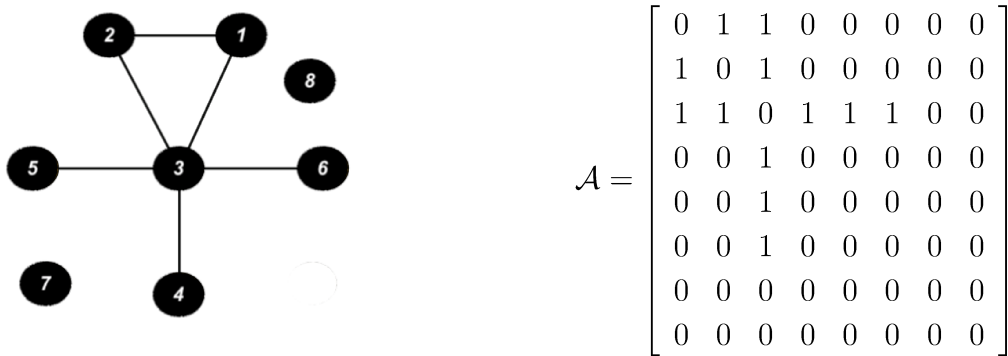


Figura 7 – Exemplo de grafo não direcionado e sua matriz de adjacências.

Fonte: adaptado de Silva e Zhao (2016, p. 21)

**Definição 2.1.7** (Grafo ponderado). *Um grafo  $\mathcal{G}$  é dito ponderado quando suas arestas possuem indicação de peso.*

A Figura 6 ilustra um grafo ponderado direcionado. Nota-se, em sua matriz de pesos denotada por  $\mathcal{W}$ , que os elementos não nulos indicam a existência de aresta partindo do vértice de origem dirigindo-se ao vértice de destino (convencionado, respectivamente, pela  $i$ -ésima linha e  $j$ -ésima coluna de  $\mathcal{W}$ ), ou seja,  $\mathcal{W}_{ij} > 0 \Leftrightarrow (i, j) \in \mathcal{E}$  e, ainda, que o valor de cada posição de  $\mathcal{W}$  representa o peso daquela aresta.

**Definição 2.1.8** (Vizinhos de um vértice). *Os vizinhos de um vértice  $v \in \mathcal{V}$ , no grafo  $\mathcal{G}$ , é o conjunto de todos os vértices adjacentes a  $v$ , denotado por*

$$\mathcal{N}(v) = \{u \mid (v, u) \in \mathcal{E}\} \quad (1)$$

Na Figura 7,  $\mathcal{N}(3) = \{1, 2, 4, 5, 6\}$ ,  $\mathcal{N}(1) = \{2, 3\}$ ,  $\mathcal{N}(7) = \{\}$ . Para o grafo da Figura 5,  $\mathcal{N}(3) = \{1, 2, 4, 5, 6\}$ ,  $\mathcal{N}(1) = \{2\}$ ,  $\mathcal{N}(7) = \{\}$ .

**Definição 2.1.9** (Grau de um vértice). *Em um grafo não direcionado  $\mathcal{G}$ , o grau de um vértice  $v \in \mathcal{V}$  (denotado por  $k_v$ ) é o total de vértices adjacentes a  $v$ . Formalmente,  $k_v = |\mathcal{N}(v)|$ , ou seja, o grau de  $v$  é a quantidade de seus vizinhos.*

Na Figura 7,  $k_3 = |\mathcal{N}(3)| = 5$ ,  $k_1 = |\mathcal{N}(1)| = 2$ ,  $k_7 = |\mathcal{N}(7)| = 0$ .

Em grafos direcionados tem-se que o grau de um vértice é a soma de arestas que nele chegam com a soma das arestas que dele saem. Formalmente, tem-se  $k_v = k_v^{in} + k_v^{out}$ , onde  $k_v^{in}$  indica o grau de entrada e  $k_v^{out}$ , o grau de saída.

Na Figura 5,  $k_3 = k_3^{in} + k_3^{out} = 0 + 5 = 5$ ,  $k_1 = k_1^{in} + k_1^{out} = 2 + 1 = 3$ ,  $k_7 = k_7^{in} + k_7^{out} = 0 + 0 = 0$ .

**Definição 2.1.10** (Grau médio do grafo). *O grau médio de um grafo é calculado como a média aritmética dos graus de todos os seus vértices, formalizado por*

$$\bar{k} = \frac{1}{n} \sum_{i \in \mathcal{V}} k_i \quad (2)$$

Adicionalmente, Chapela et al. (2015, p. 14 - 15) relaciona o grau médio com a quantidade de vértices ( $n = |\mathcal{V}|$ ) e arestas ( $m = |\mathcal{E}|$ ), formalizado pela expressão

$$\bar{k} = \frac{1}{n} \sum_{i \in \mathcal{V}} k_i = \frac{2m}{n}$$

simplificando o cálculo do grau médio do grafo.

Da Figura 7, tem-se  $m = 6$  arestas e  $n = 8$  vértices. Logo:

$$\bar{k} = \frac{1}{n} \sum_{i \in \mathcal{V}} k_i = \frac{2m}{n} = \frac{2 * 6}{8} = 1.5$$

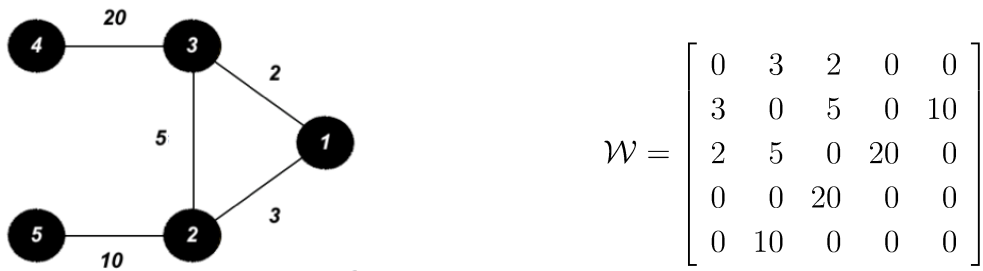


Figura 8 – Exemplo de grafo não direcionado ponderado e sua matriz de pesos.

Fonte: adaptado de Silva e Zhao (2016, p. 23)



Figura 9 – Exemplo de grafo direcionado ponderado e sua matriz de pesos.

Fonte: adaptado de Silva e Zhao (2016, p. 23)

**Definição 2.1.11** (Passeio ou Walk). *Um passeio em um grafo  $\mathcal{G}$ , contendo vértices  $v_1, v_2, \dots, v_k \in \mathcal{V}$  e  $k \geq 2$ , é uma sequência ordenada de vértices*

$$\mathcal{W} = \{(v_1, v_2), (v_2, v_3), \dots, (v_{k-1}, v_k) \mid \{2, \dots, k\} : (v_{k-1}, v_k) \in \mathcal{E}\}$$

Em um passeio, os vértices que o constituem podem ser revisitados. Ainda,  $v_1$  e  $v_k$  são denominados vértices de origem e de destino, respectivamente. Quando  $v_1 = v_k$ , tem-se um passeio fechado e, se  $v_1 \neq v_k$ , um passeio em aberto. Quando um passeio é constituído por um único vértice ele é dito trivial.

**Definição 2.1.12** (Trajeto ou Trail). *Um trajeto é um passeio cujas arestas não se repetem.*

**Definição 2.1.13** (Circuito ou Tour). *Um circuito é um trajeto fechado.*

**Definição 2.1.14** (Caminho ou Path). *Um caminho  $\mathcal{P}$  é um passeio não trivial em que todos os vértices intermediários são distintos.*

**Definição 2.1.15** (Caminho Hamiltoniano). *Um caminho Hamiltoniano é um caminho que passa por todos os vértices do grafo uma única vez.*

**Definição 2.1.16** (Caminho Euleriano). *Um caminho Euleriano é um caminho que passa por todas as arestas do grafo uma única vez.*

**Definição 2.1.17** (Ciclo). *Um ciclo é um caminho fechado.*

**Definição 2.1.18** (Ciclo Hamiltoniano). *Um ciclo Hamiltoniano é um ciclo que passa por todos os vértices do grafo uma única vez.*

**Definição 2.1.19** (Ciclo Euleriano). *Um ciclo Euleriano é um ciclo que passa por todas as arestas do grafo uma única vez.*

**Definição 2.1.20** (Comprimento de um passeio). *Seja um passeio*

$$\mathcal{W} = \{(v_1, v_2), (v_2, v_3), \dots, (v_{k-1}, v_k) \mid k \geq 2 : (v_{k-1}, v_k) \in \mathcal{E}\}$$

O comprimento de  $\mathcal{W}$  é a quantidade de arestas ao longo de seu percurso, ou seja,  $|\mathcal{W}| = k - 1 \geq 1$ .

**Definição 2.1.21** (Distância de um passeio ou caminho). *Seja um passeio*

$$\mathcal{W} = \{(v_1, v_2), (v_2, v_3), \dots, (v_{k-1}, v_k) \mid k \geq 2 : (v_{k-1}, v_k) \in \mathcal{E}\}$$

A distância  $d$  de  $\mathcal{W}$  é dada por:

$$d(\mathcal{W}) = \sum_{k=2}^K |(v_{k-1}, v_k)| = \sum_{k=2}^K w_{k-1,k} \quad (3)$$

onde:

$|(v_{k-1}, v_k)|$  : é o peso da aresta entre os vértices  $v_{k-1}$  e  $v_k$

$w_{i,j}$  : é a  $(i,j)$ -ésima entrada na matriz de pesos  $\mathcal{W}$

**Definição 2.1.22** (Caminho mais curto entre dois vértices). *O caminho mais curto entre dois vértices  $u \in \mathcal{V}$  e  $v \in \mathcal{V}$ , denotado por  $d_{uv}$ , é o caminho iniciado no vértice  $u$ , finalizado em  $v$  e que possui a menor distância. Formalmente:*

$$d_{uv} = \min_{\mathcal{W}_{u \rightarrow v}} d(\mathcal{W}_{u \rightarrow v}) \quad (4)$$

onde:

$\mathcal{W}_{u \rightarrow v}$  : corresponde ao caminho iniciado em  $u$  e finalizado em  $v$

Na Figura 4,  $\mathcal{W}_1 = \{(3, 3)\}$  é um passeio trivial,  $\mathcal{W}_2 = \{(4, 3), (3, 3), (3, 2)\}$  é um passeio em aberto,  $\mathcal{W}_3 = \{(4, 3), (3, 3), (3, 2), (2, 1), (1, 3), (3, 4)\}$  é um passeio fechado,  $\mathcal{W}_4 = \{(4, 3), (3, 2), (2, 1)\}$  é um trajeto aberto,  $\mathcal{W}_5 = \{(3, 2), (2, 1), (1, 3)\}$  é um trajeto fechado. Nenhum passeio alcança o nó 5. Os comprimentos desses passeios são:  $|\mathcal{W}_1| = 1$ ,  $|\mathcal{W}_2| = 3$ ,  $|\mathcal{W}_3| = 6$ ,  $|\mathcal{W}_4| = 3$ ,  $|\mathcal{W}_5| = 3$

## 2.1.1 Medidas de Centralidade

Para Erciyes (2014, p. 70), centralidade é uma métrica que quantifica a importância de um nó ou aresta, baseado em sua posição topológica em uma rede complexa. Dehmer e Basak (2012, p. 69) complementam ao afirmar que centralidade é um conceito importante para análises de redes, pois ajuda a localizar os nós centrais (e, portanto, importantes) em uma rede complexa. A seguir, são apresentadas algumas métricas de centralidade relevantes para esta pesquisa.

### 2.1.1.1 Degree Centrality

Brandes e Erlebach (2005, p. 21), Costa et al. (2007, p. 8), Dehmer e Basak (2012, p. 70) e Erciyes (2014, p. 86) definem que o grau de um nó é considerado a maneira mais simples de se especificar a sua importância em uma rede. Logo, espera-se que o nó possuidor do grau mais elevado (portanto, com mais vizinhos) esteja envolvido em

comunicações mais frequentes do que o nó que possua grau mais baixo. Para grafos não direcionados  $\mathcal{G}(\mathcal{V}, \mathcal{E})$ , a *degree centrality* de um vértice  $i$  ( $C_D(i)$ ) é simplesmente o valor de seu grau, denotado por  $k(i)$ . Para grafos direcionados, a *degree centrality* é dada pela soma de *in-degree centrality* ( $k^{in}(i)$ ) e *out-degree centrality* ( $k^{out}(i)$ ), ou seja,  $k(i) = k^{in}(i) + k^{out}(i)$ . Formalmente, dado um grafo  $\mathcal{G}(\mathcal{V}, \mathcal{E})$  composto por  $n$  vértices e uma matriz de adjacências  $\mathcal{A}_{n \times n}$ , a *degree centrality* de um vértice  $i$  é:

$$C_D(i) = \sum_{j \in \mathcal{V}} a_{ij} \quad (5)$$

onde  $a_{ij}$  é  $ij$ -ésima entrada da matriz  $\mathcal{A}$  e  $n$  é a quantidade de nós da rede.

O cálculo das *degree centrality* de todos os nós da rede é atribuído a um vetor  $C_D$ , em que cada posição corresponde à soma dos valores da linha correspondente de  $\mathcal{A}$  em  $n$  passos, totalizando  $\Theta(n^2)$  em tempo de execução.

### 2.1.1.2 Weighted Degree Centrality

Costa et al. (2007, p. 9) e Opsahl, Agneessens e Skvoretz (2010, p. 246) definem a *weighted degree centrality* ( $C_D^w(i)$ ) como uma extensão da *degree centrality* para redes ponderadas, em que consideram a soma dos pesos das arestas ligadas ao nó de interesse ( $s_i$ ). Em redes ponderadas direcionadas, essa métrica formaliza-se por:

$$C_D^w(i) = s^{in}(i) + s^{out}(i) \quad (6)$$

$$s^{in}(i) = \sum_j^n w_{ji} \quad (7)$$

$$s^{out}(i) = \sum_j^n w_{ij} \quad (8)$$

Para redes ponderadas não direcionadas, tem-se:

$$C_D^w(i) = s(i) = \sum_j^n w_{ij} \quad (9)$$

onde  $w_{ij}$  é a  $ij$ -ésima entrada da matriz de pesos  $\mathcal{W}_{n \times n}$  da rede.

A complexidade do cálculo, em tempo de execução, das *weighted degree centrality* de todos os nós da rede é idêntico ao realizado para *degree centrality*, resultando em  $\Theta(n^2)$ .

### 2.1.1.3 Triangles

O cálculo do número de triângulos de uma rede tem sido objeto de pesquisas frequentes desenvolvidas por diversos autores nos últimos anos e utiliza multiplicação de matrizes. Dentre esses autores destaca-se o trabalho realizado por Coppersmith e Winograd (1987), que determina o valor do expoente  $\gamma$  da multiplicação de matrizes como sendo 2.376.

O trabalho realizado por Schank e Wagner (2005) compreende um estudo experimental rigoroso sobre algoritmos conhecidos para contagem e listagem de triângulos presentes

em redes complexas, e contribui apresentando melhorias de performance em um desses algoritmos. A Figura 10 apresenta resumidamente os algoritmos estudados por esses pesquisadores, juntamente com suas estimativas de tempo de execução.

Considerando o algoritmo “*ayz with fast matrix-multiplication*” proposto por Schank e Wagner (2005), sua ordem de complexidade ( $O(m^{\frac{2\gamma}{\gamma+1}})$ ) em tempo de execução, o valor de  $\gamma = 2.376$ , e  $m$  a quantidade de arestas da rede, tem-se:

$$O(m^{\frac{2\gamma}{\gamma+1}}) = O(m^{\frac{2 \cdot 2.376}{2.376+1}}) = O(m^{1.408})$$

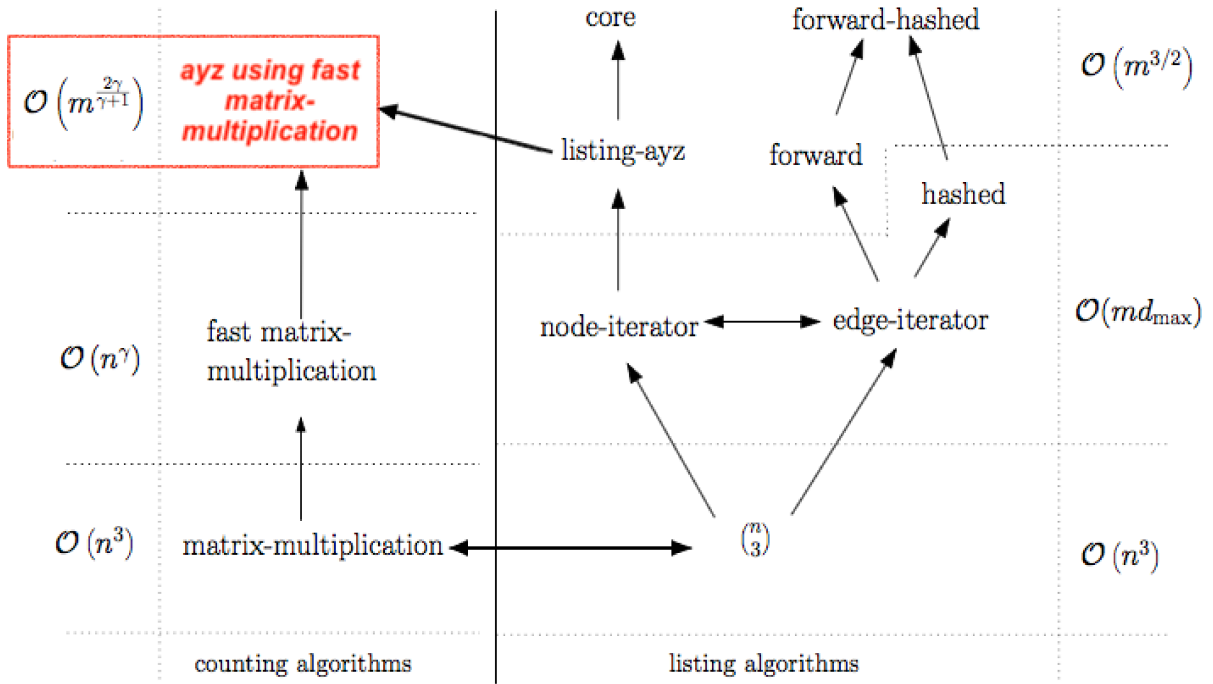


Figura 10 – Resumo esquemático de algoritmos de multiplicação de matrizes utilizados na contagem e listagem de triângulos presentes em redes complexas.

Fonte: adaptado de Schank e Wagner (2005, p. 6)

#### 2.1.1.4 Betweenness Centrality

Esta métrica, como descrevem Koschützki (2007, p.73), Silva e Zhao (2016, p. 48)) e Zhang (2009, p. 73), quantifica a participação de um dado vértice nos caminhos mais curtos entre cada par de vértices da rede, e é calculada pela seguinte expressão:

$$C_B(v) = \sum_{\substack{s \neq v \neq t \\ s, v, t \in V}} \frac{\rho_{st}(v)}{\rho_{st}} \quad (10)$$

onde:

$\rho_{st}(v)$  : quantidade de menores caminhos entre os vértices  $s$  e  $t$  que contém o vértice  $v$



$\rho_{st}$  : quantidade de menores caminhos entre os vértices  $s$  e  $t$

O algoritmo desenvolvido por Brandes e Erlebach (2005, p. 69) para cálculo da *betweenness centrality* é considerado o mais rápido disponível, com tempo de execução da ordem  $O(mn + n^2 \log n)$  para grafos ponderados, e da ordem  $O(mn)$  para grafos não ponderados.

#### 2.1.1.5 Bridging Coefficient

Hwang et al. (2006) definem que esta métrica visa determinar o quão bem posicionado um nó está em relação aos nós de mais alto grau da rede. Formalmente, é definida por:

$$BC(v) = \frac{\frac{1}{k_v}}{\sum_{i \in \mathcal{N}(v)} \frac{1}{k_i}} \quad (11)$$

onde:

$k_v$  : é o grau do nó  $v$

$\mathcal{N}(v)$  : é o conjunto dos vizinhos do nó  $v$ .

Em trabalho posterior, Hwang et al. (2008, p. 343), afirmam que a complexidade média para cálculo da *bridging coefficient* é da ordem  $O(n(\log n)^2)$ , em tempo de execução.

#### 2.1.1.6 Bridging Centrality

Ainda no trabalho desenvolvido por Hwang et al. (2006) tem-se que um *bridging node* é um nó que interliga componentes densamente conectadas de uma rede. A métrica *bridging centrality* é utilizada para identificá-los quantitativamente, atribuindo-lhes um valor que pode ser calculado formalmente por:

$$C_R(v) = BC(v) \times C_B(v) \quad (12)$$

onde:

$BC(v)$  : é o valor da *bridging coefficient* do nó  $v$

$C_B(v)$  : é o valor da *betweenness centrality* do nó  $v$

Dessa forma, os nós da rede que possuem os maiores valores de *bridging centrality* são considerados *bridging nodes*.

Em trabalho posterior, Hwang et al. (2008, p. 343), afirmam que a complexidade total, em tempo de execução, da *bridging centrality* varia entre  $O(nm)$  e  $O(nm + n^2 \log n)$  para redes não ponderadas e para redes ponderadas, respectivamente.

### 2.1.2 Outras Medidas de Centralidade

A tabela a seguir apresenta outras métricas de centralidade constantes na literatura, juntamente com suas ordens de complexidade de execução.

Tabela 2 – Complexidade dos algoritmos relacionados ao cálculo de métricas de centralidade

Time Complexity	Algorithm/Centrality	Reference
$O(n^2)$	Dijkstra SSSP	(ERCIYES, 2014, p. 84)
$O(m + n \log n)$	Dijkstra SSSP (Fibonacci heap; priority queue)	(ERCIYES, 2014, p. 84)
$O(n^3)$	Floyd-Warshall APSP	(ERCIYES, 2014, p. 86)
$O(mn + n^2 \log n)$	Eccentricity	(GRANDO; LAMB, 2015)
$O(nm + n^2 \log n)$	Closeness Centrality (weighted graph)	(ERCIYES, 2014, p. 91)
$O(nm + n^2 \log n)$	Harmonic Closness Centrality	(ROCHAT, 2009, p. 10)
$O(n^3 + mn \log n)$	Walk-based Betweenness Centrality	(ROCHAT, 2009, p. 10)
$O(n^3 + mn \log n)$	Information Centrality	(ROCHAT, 2009, p. 10)

## 2.2 Aprendizado de Máquina

De acordo com Silva e Zhao (2016, p. 71), o aprendizado de máquina está relacionado ao estudo, projeto e desenvolvimento de algoritmos capazes de fazer com que computadores aprendam sem terem sido explicitamente programados. Por ser bastante genéricas, as técnicas de aprendizado de máquina podem ser utilizadas em diversas configurações e propósitos. Para tal, é necessário traduzir o problema a ser tratado para o domínio da aprendizagem de máquina que, em geral, requer um conjunto de características como entrada e produz como saída um critério de agrupamento ou classificação.

Os algoritmos de aprendizado de máquina atualmente são agrupados segundo três paradigmas: não supervisionados, supervisionados e semi-supervisionados. Os algoritmos não supervisionados dispensam qualquer tipo de conhecimento externo, sendo guiados exclusivamente pela estrutura intrínseca dos itens de dados ao longo do processo de aprendizagem. Dentre as principais funções de sua aplicação pode-se destacar: agrupamento, detecção de *outlier*, redução de dimensionalidade, e associação. Em contrapartida, os algoritmos supervisionados são aqueles que utilizam exclusivamente informações externas para induzir ou treinar suas hipóteses. São frequentemente empregados em problemas de classificação, quando os dados rotulados são discretos, e em problemas de regressão, quando tratam valores contínuos. Entre esses dois paradigmas, encontram-se os algorit-

mos semi-supervisionados, que mesclam características dos algoritmos supervisionados e dos não supervisionados, ou seja, empregam dados rotulados e não-rotulados no processo de aprendizagem.

Silva e Zhao (2016, p. 77) dividem os algoritmos supervisionados nos seguintes grupos: *decision trees*, *rule-based induction*, *artificial neural networks*, *bayesian networks*, *statistical learning theory*, *instance-based learning* e *network-based methods*.

O estudo realizado por Fernández-Delgado et al. (2014) avaliou 179 algoritmos classificadores, provenientes de 17 famílias distintas de algoritmos, implementados no WEKA, no R, em C e no Matlab, em que conclui que os classificadores com maior probabilidade de serem os mais acertivos são os do grupo RandomForest.

A Tabela 3 apresenta os algoritmos de aprendizado supervisionados implementados no WEKA e que são objetos de estudo deste trabalho de pesquisa. Os termos que identificam o grupo dos algoritmos (coluna "Grupo") referem-se à nomenclatura adotada no WEKA.

Tabela 3 – Algoritmos ML selecionados para a pesquisa

Grupo	Algoritmo	Descrição
Instance-based learning	KStar <sup>1</sup>	Proposto por Cleary e Trigg (1995), classifica as instâncias por similaridade.
Meta	Bagging <sup>2</sup>	Implementação do algoritmo proposto por Breiman (1996).
	LogitBoost <sup>3</sup>	Executa regressão logística aditiva, proposto por Friedman, Hastie e Tibshirani (1998)
	RandomCommittee <sup>4</sup>	
	RandomSubSpace <sup>5</sup>	Proposto por Ho (1998),
	J48 <sup>6</sup>	Implementação do algoritmo C4.5 proposto por Salzberg (1994)

### Decision Trees

<sup>1</sup> [weka.sourceforge.net/doc.dev/weka/classifiers/lazy/KStar.html](http://weka.sourceforge.net/doc.dev/weka/classifiers/lazy/KStar.html), acessado em: 04/02/2018

<sup>2</sup> [weka.sourceforge.net/doc.dev/weka/classifiers/meta/Bagging.html](http://weka.sourceforge.net/doc.dev/weka/classifiers/meta/Bagging.html), acessado em: 04/02/2018

<sup>3</sup> [weka.sourceforge.net/doc.dev/weka/classifiers/meta/LogitBoost.html](http://weka.sourceforge.net/doc.dev/weka/classifiers/meta/LogitBoost.html), acessado em: 04/02/2018

<sup>4</sup> [weka.sourceforge.net/doc.dev/weka/classifiers/meta/RandomCommittee.html](http://weka.sourceforge.net/doc.dev/weka/classifiers/meta/RandomCommittee.html), acessado em: 04/02/2018

<sup>5</sup> [weka.sourceforge.net/doc.dev/weka/classifiers/meta/RandomSubSpace.html](http://weka.sourceforge.net/doc.dev/weka/classifiers/meta/RandomSubSpace.html), acessado em: 04/02/2018

<sup>6</sup> [weka.sourceforge.net/doc.dev/weka/classifiers/trees/J48.html](http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/J48.html), acessado em: 04/02/2018

Tabela 3 – Algoritmos ML selecionados para a pesquisa

Grupo	Algoritmo	Descrição
	LMT <sup>7</sup>	Implementação de árvore de decisão cujas folhas possuem regressão logística, proposto por Landwehr, Hall e Frank (2005)
	RandomForest <sup>8</sup>	Implementa algoritmo proposto por Breiman (2001)
	RandomTree <sup>9</sup>	Árvore de decisão, sem poda, contemplando $K$ atributos escolhidos aleatoriamente
	REPTree <sup>10</sup>	Árvore de decisão de aprendizado rápido
Rule-based inductions	JRip <sup>11</sup>	Proposto por Cohen (1995), implementa regra de aprendizado proposicional.
	OneR <sup>12</sup>	Proposto por Holte (1993)
	PART <sup>13</sup>	Proposto por Frank e Witten (1998)
Artificial Neural Networks	Multilayer Perceptron <sup>14</sup>	Implementação de rede neural artificial <i>backpropagation</i> para classificação de instâncias
Bayesian networks	NaïveBayes <sup>15</sup>	Proposto por John e Langley (1995)

<sup>7</sup> [weka.sourceforge.net/doc.dev/weka/classifiers/trees/LMT.html](http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/LMT.html), acessado em: 04/02/2018

<sup>8</sup> [weka.sourceforge.net/doc.dev/weka/classifiers/trees/RandomForest.html](http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/RandomForest.html), acessado em: 04/02/2018

<sup>9</sup> [weka.sourceforge.net/doc.dev/weka/classifiers/trees/RandomTree.html](http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/RandomTree.html), acessado em: 04/02/2018

<sup>10</sup> [weka.sourceforge.net/doc.dev/weka/classifiers/trees/REPTree.html](http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/REPTree.html), acessado em: 04/02/2018

<sup>11</sup> [weka.sourceforge.net/doc.dev/weka/classifiers/rules/JRip.html](http://weka.sourceforge.net/doc.dev/weka/classifiers/rules/JRip.html), acessado em: 04/02/2018

<sup>12</sup> [weka.sourceforge.net/doc.dev/weka/classifiers/rules/OneR.html](http://weka.sourceforge.net/doc.dev/weka/classifiers/rules/OneR.html), acessado em: 04/02/2018

<sup>13</sup> [weka.sourceforge.net/doc.dev/weka/classifiers/rules/PART.html](http://weka.sourceforge.net/doc.dev/weka/classifiers/rules/PART.html), acessado em: 04/02/2018

<sup>14</sup> [weka.sourceforge.net/doc.dev/weka/classifiers/functions/MultilayerPerceptron.html](http://weka.sourceforge.net/doc.dev/weka/classifiers/functions/MultilayerPerceptron.html), acessado em: 04/02/2018

<sup>15</sup> [weka.sourceforge.net/doc.dev/weka/classifiers/bayes/NaiveBayes.html](http://weka.sourceforge.net/doc.dev/weka/classifiers/bayes/NaiveBayes.html), acessado em: 04/02/2018

## 2.3 Avaliação de Classificadores - curva ROC e AUC

Segundo Fawcett (2003), Fawcett (2006) e Prati, Batista e Monard (2008), gráficos ROC são muito úteis na organização e quantificação da performance de classificadores. Seu uso tem sido frequente nas pesquisas de *Data Mining* e *Machine Learning*.

Gráficos ROC foram originalmente utilizados em detecção de sinais, para se avaliar a qualidade de transmissão de um sinal em um canal com ruído. Também são muito utilizados em psicologia para se avaliar a capacidade de indivíduos distinguirem entre estímulo e não estímulo; em medicina, para analisar a qualidade de um determinado teste clínico; em economia, para a avaliação de desigualdade de renda; e em previsão do tempo, para se avaliar a qualidade das predições de eventos raros (PRATI; BATISTA; MONARD, 2008).

Para se induzir um classificador, um algoritmo de aprendizado supervisionado utiliza uma amostra de casos para os quais se conhece a classificação verdadeira. Cada caso é descrito por um conjunto de atributos. Para se distinguir casos entre as possíveis classificações, cada caso é rotulado com um atributo especial, denominado classe, cujos valores se referem à classificação verdadeira dos casos. Em problemas de classificação binária são consideradas apenas duas classes, em geral, rotuladas como positiva e negativa, respectivamente. Casos rotulados são chamados de exemplos, e a amostra utilizada pelo algoritmo de aprendizado para induzir o modelo de classificação é chamada de conjunto de exemplos de treinamento (PRATI; BATISTA; MONARD, 2008).

Uma maneira natural de apresentar as estatísticas para a avaliação de um modelo de classificação é por meio de uma tabulação cruzada entre a classe prevista pelo modelo e a classe real dos exemplos. Essa tabulação é conhecida como tabela de contingência, ou matriz de confusão (PRATI; BATISTA; MONARD, 2008). Na Figura 11 é mostrada uma matriz de contingência genérica e as medidas comuns que podem ser obtidas a partir dela. Quando um exemplo real positivo é classificado como positivo, ele é denominado verdadeiro positivo (True Positive). Quando um exemplo real negativo é classificado como positivo, ele é denominado falso positivo (False Negative). Nomenclatura similar é utilizada no caso dos exemplos classificados como negativos. Nessa tabela, **TP**, **FP**, **FN** e **TN** correspondem, respectivamente, às quantidades de verdadeiro/falso positivo/negativo. **YES** e **NO** correspondem ao número de exemplos preditos (*Classified class*) como positivos/negativos e **yes** e **no** ao número real (*Real class*) de exemplos positivos/negativos na amostra.

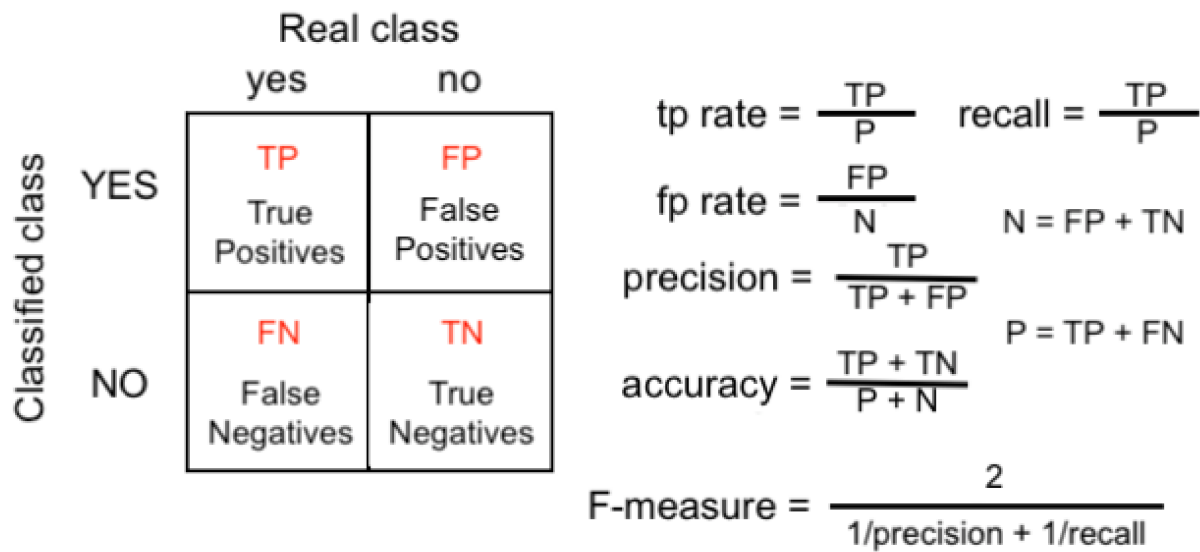


Figura 11 – Matriz de contingências e algumas métricas comuns.

Fonte: adaptado de Fawcett (2006, p. 6)

A curva ROC é um gráfico bidimensional, onde o eixo  $Y$  corresponde à taxa de verdadeiros positivos (tp rate) e o eixo  $X$  corresponde à taxa de falsos positivos (fp rate). A Figura 12 representa um gráfico de curva ROC obtido a partir de um conjunto de testes.

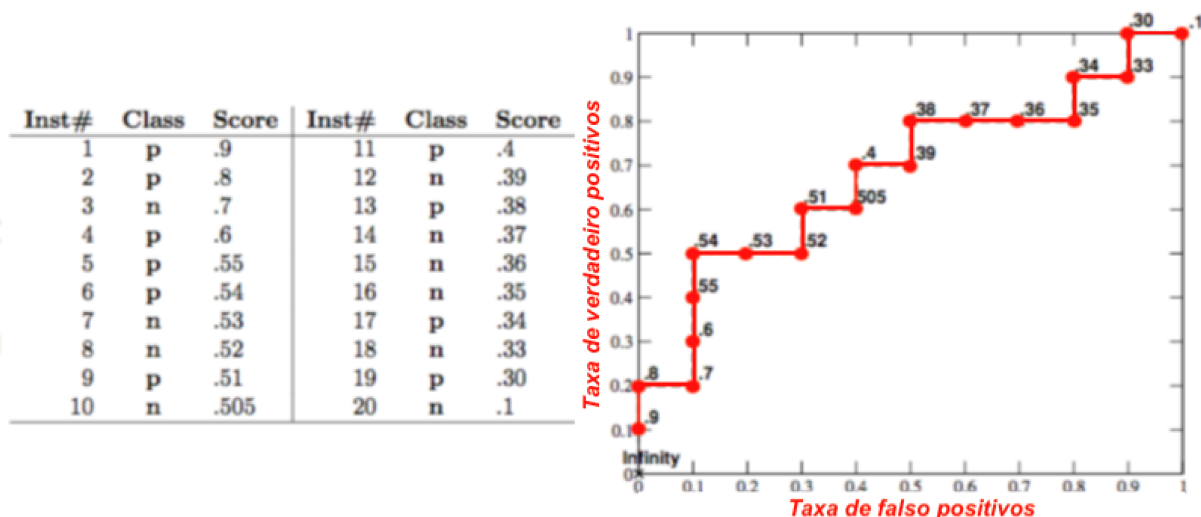


Figura 12 – Exemplo de curva ROC obtida a partir de um conjunto de testes. A tabela ao lado corresponde aos dados gerados por um classificador em que se observa a classe predita e o score atribuído a cada instância. Os pontos no gráfico estão rotulados segundo o valor threshold correspondente.

Fonte: adaptado de Fawcett (2003, p. 5)

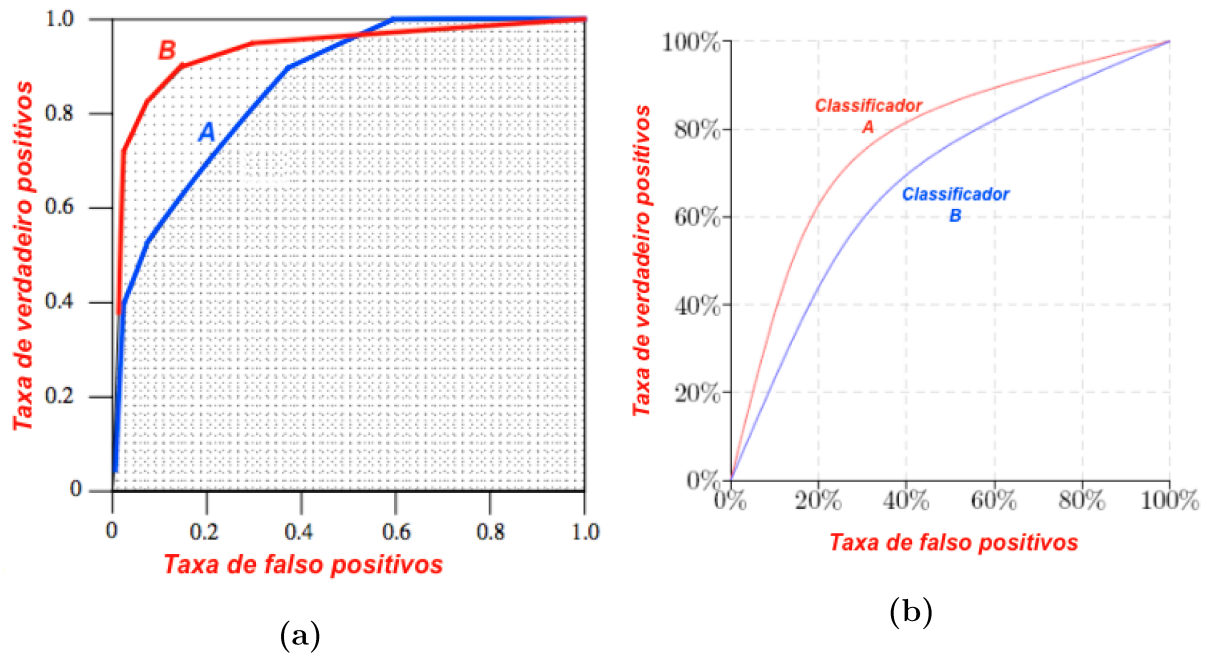


Figura 13 – Exemplos de curvas ROC, onde (a) apresenta AUC para as curvas ROC A e B, respectivamente, e (b) apresenta curvas ROC de dois classificadores.

Fontes: adaptadas de Fawcett (2006) e Prati, Batista e Monard (2008)

Uma consequência do gráfico ROC é a AUC, constituído pela área abaixo da curva ROC, e que varia entre 0 e 1. Seu valor é utilizado para avaliar a performance de classificadores. Quanto maior a AUC, mais íngreme será a curva ROC indicando haver mais instâncias TP do que FP. Considerar a análise do valor de AUC quando muito pequeno também é interessante, pois o classificador está acertando a classificação na forma de complemento de 1, ou seja, o inverso de sua classificação está correto. Na Figura 13 (a) tem-se a  $AUC_B > AUC_A$ .

## 2.4 Trabalhos Correlatos

Medidas de centralidade são métricas importantes na análise de sistemas complexos, geralmente envolvendo grandes redes. A obtenção de tais medidas, por meios convencionais, é computacionalmente dispendiosa, tanto em termos de tempo de processamento quanto em memória. No trabalho realizado por Grando e Lamb (2015), é analisada a hipótese de se obter uma boa aproximação para medidas de centralidade de alta complexidade a partir de medidas de centralidade de baixa complexidade por meio de algoritmos de aprendizado de máquina. Apesar desses algoritmos demandarem tempo considerável na fase de aprendizado, uma vez treinado, seu modelo de predição pode ser aplicado no cálculo das medidas de centralidade com tempo linear, tornando-se uma alternativa viável. Nesse trabalho, os autores analisaram oito medidas de centralidade divididas em dois grupos: centralidade de baixa complexidade (*betweenness*, *closeness*, *degree*, *eccentri-*

city) e centralidade de alta complexidade (*eigenvector*, *information*, *subgraph*, e *walk-based betweenness*). Foram utilizadas redes artificiais geradas segundo modelos geradores conhecidos para simular redes reais. Os algoritmos utilizados foram *ZeroR*, *REPTree* e *Multilayer Perceptron*, todos com parâmetros *default* a partir do WEKA. Os resultados obtidos nos experimentos realizados sugerem que a hipótese se confirma, e vai mais além, que não são necessárias muitas métricas para treinar os algoritmos e obter boas aproximações. Também apontam que a diminuição do número de atributos utilizados para determinar as métricas mais complexas reduz significativamente a performance, ou seja, reduzindo sua eficiência. Os autores ainda sugerem que, em trabalhos futuros, outros algoritmos de aprendizado sejam testados, assim como seus parâmetros sejam otimizados. Em trabalho posterior, Grando e Lamb (2016) utilizaram redes neurais artificiais para alcançar os valores das métricas *betweenness centrality* e *closeness centrality* a partir das métricas *degree centrality* e *eigenvector centrality* utilizadas como atributos de treinamento. O objetivo é criar um modelo de regressão que possa ser utilizado posteriormente para calcular *betweenness centrality* e *closeness centrality* em tempo linear. Para treinar a rede neural, os autores utilizaram doze algoritmos de treinamento *backpropagation*. Nos experimentos realizados, destacam que a métrica *closeness centrality* obteve melhor aproximação em relação à *betweenness centrality*, e que o algoritmo de *backpropagation* Levenberg-Marquardt apresentou a melhor performance, tanto em tempo de execução quanto na qualidade da aproximação. Os autores concluem esse estudo afirmando que o uso de algoritmos de aprendizado de máquina são estimadores muito bons para medidas de centralidade quando os métodos convencionais não puderem ser utilizados devido a restrições de tempo de execução. O estudo realizado por Ananthasubramanian et al. (2012) apresenta um método computacional para predição de PPIs bacterianos por meio de transferência de modelos computacionais inter-espécies. Também propõe a estimativa de características desconhecidas a partir de características conhecidas de um organismo. Nesse trabalho foi desenvolvido um classificador baseado em *Random Forest* e que foi aplicado às características de dois organismos conhecidos (*Mycobacterium tuberculosis* - **Mtb** e *Clostridium difficile* - **CD**) independentemente. Para demonstrar a utilidade da transferência de modelos computacionais inter-espécies, procedeu-se com a aplicação do modelo de aprendizado gerado para a PPI **Mtb** prosseguindo com a classificação dos pares de proteína da PPI **CD**, obtendo precisão da ordem de 88%.

Emamjomeh et al. (2014) realizaram um estudo sobre a predição de interações proteína-proteína inter-espécies por meio de um método de aprendizado *ensemble*. Os algoritmos utilizados como base desse método foram Random Forest, Naïve Bayes, Support Vector Machine e Multilayer Perceptron. Para combinar os algoritmos de base, um outro algoritmo Multilayer Perceptron foi utilizado como *meta-learner*, fornecendo o resultado da predição. Como entrada para o método, os autores utilizaram um vetor de características contendo seis descritores, dentre os quais as métricas de rede *degree*, *neighborhood connec-*



*tivity, average shortest path lenght, stress, eccentricity, closeness, betweenness centrality, radiality e clustering coefficient.* Os testes foram realizados utilizando as PPIs humana e do vírus da hepatite C, tendo como resultado uma acurácia de 0.83, especificidade de 0.94, sensibilidade de 0.79, por meio de validação cruzada.



## Metodologia

Este trabalho de pesquisa foi organizado em quatro grandes etapas, apresentadas esquematicamente pela Figura 14.

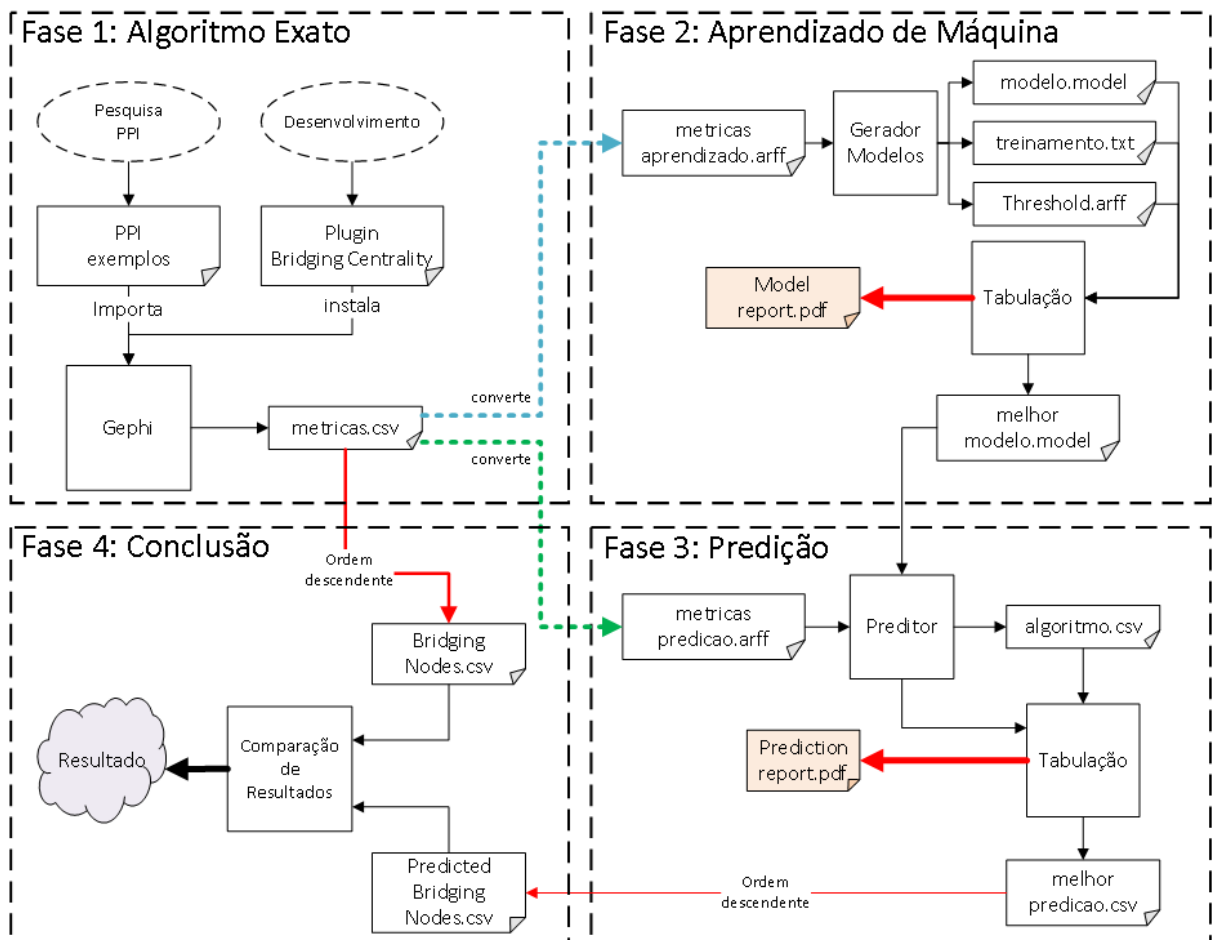


Figura 14 – Diagrama esquemático das etapas da pesquisa.

### 3.1 Fase 1: Algoritmo Exato

Foi necessário adotar um conjunto de dados correspondentes a redes PPI de organismos biológicos conhecidos. As redes PPI adotadas, listadas na Tabela 4, foram obtidas a partir do trabalho realizado por Oliveira e Santos (2016) e disponibilizadas como material suplementar, na plataforma *ResearchGate*<sup>1</sup>, sob o título “Gene Ontology *de novo* PPI”. As figuras 15 e 16, geradas pelo Gephi usando o algoritmo de *Fruchterman Reingold*<sup>2</sup>, ilustram as redes PPIs dos organismos *Corynebacterium diphtheriae* NC 002935 e *Bacillus anthrax*, respectivamente.

Tabela 4 – Redes PPI adotadas para testes da pesquisa

Rede PPI	Nós	Arestas
<i>Bacillus anthrax</i>	792	19.518
<i>Clostridium botulinum</i> A2 Kyoto	659	12.978
<i>Clostridium botulinum</i> F Langeland	659	13.042
<i>Clostridium perfringens</i>	549	9.761
<i>Clostridium tetani</i> E88	451	6.342
<i>Corynebacterium diphtheriae</i> NC 002935	379	2.585
<i>Escherichia coli</i> ED1a	1.123	37.862
<i>Mycobacterium tuberculosis</i>	673	18.430
<i>Streptococcus pneumoniae</i> Taiwan19F14	433	7.582
Média	635,30	14.233,00
Desvio-Padrão	227,20	10.422,62

<sup>1</sup> [https://www.researchgate.net/publication/313342754\\_Gene\\_Ontology\\_de\\_novo\\_PPI](https://www.researchgate.net/publication/313342754_Gene_Ontology_de_novo_PPI), acessado em: 04/02/2017

<sup>2</sup> Fruchterman, T. M. J. and Reingold, E. M. (1991), Graph drawing by force-directed placement. *Softw: Pract. Exper.*, 21: 1129–1164. doi:10.1002/spe.4380211102

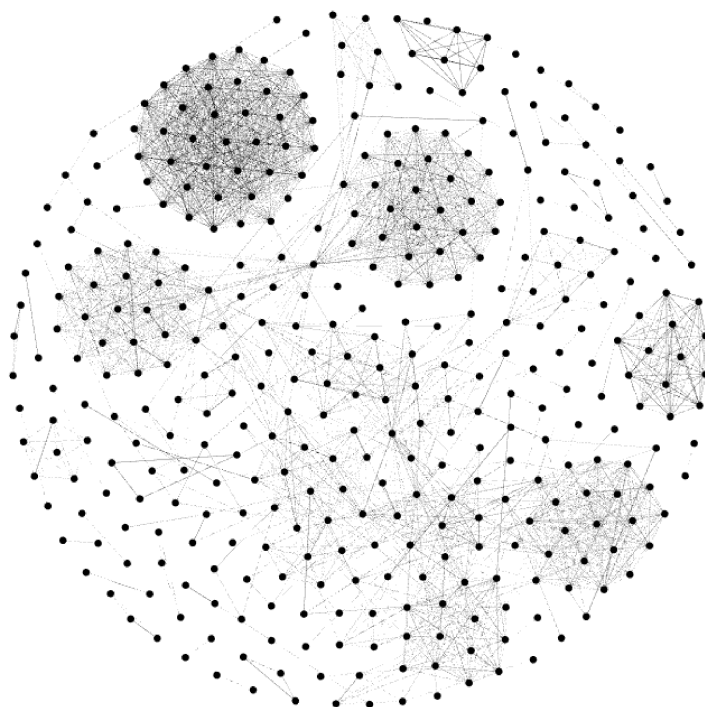


Figura 15 – Rede PPI do organismo *Corynebacterium diphtheriae* NC 002935 contendo 379 nós e 2.585 arestas, gerada pelo software Gephi.

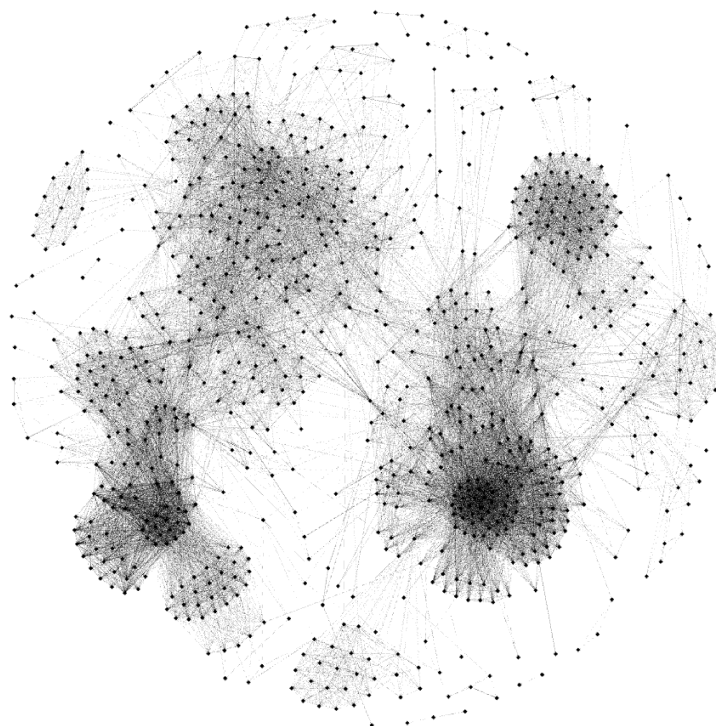


Figura 16 – Rede PPI do organismo *Bacillus anthrax* contendo 792 nós e 19.518 arestas, gerada pelo software Gephi.

Uma vez que as redes PPI foram escolhidas, realizou-se pesquisa na literatura sobre o algoritmo de cálculo da métrica *Bridging Centrality*, conforme apresentado no capítulo

2, seção 2.1.1.6. Esse algoritmo foi implementado na forma de um *plugin* para o Gephi e disponibilizado para uso.

Com o *plugin* instalado, cada rede PPI foi então importada separadamente no Gephi e, por meio das funcionalidades disponíveis, gerou-se o conjunto de métricas exatas e exportadas em arquivo no formato Comma-Separated Values (CSV). A Tabela 5 apresenta as 17 primeiras linhas do arquivo de métricas relacionado à rede PPI *Bacillus anthrax*.

Tabela 5 – Fragmento do arquivo de métricas relacionado à PPI *Bacillus anthrax*

id	D	$s_v$	$\Delta$	$BC(v)$	$C_R(v)$	class
BAA_0001	92	168.35139	2292	0.006523658	8.152973e-06	weak
BAA_0002	102	188.26359	2335	0.004061711	2.384648e-05	strong
BAA_0004	102	181.15322	2411	0.005429426	6.465369e-05	strong
BAA_0018	69	113.63337	1230	0.010860103	3.789008e-05	strong
BAA_0047	87	169.40847	2377	0.008900484	8.211381e-06	weak
BAA_0063	91	125.76373	1592	0.005519992	8.474965e-05	strong
BAA_0089	58	102.59753	1276	0.018461484	4.741778e-06	weak
BAA_0103	89	185.53377	2035	0.007786126	2.059859e-05	weak
BAA_0105	53	93.61019	1154	0.021821045	3.903574e-06	weak
BAA_0112	73	167.86190	2141	0.013418570	5.380062e-06	weak
BAA_0113	79	178.09469	2224	0.009767483	1.464775e-05	weak
BAA_0115	103	217.36322	3003	0.005975482	1.257616e-04	strong
BAA_0122	73	172.06856	2141	0.013418570	5.380062e-06	weak
BAA_0123	46	79.30727	791	0.017143929	6.234630e-05	strong
BAA_0124	46	79.35096	791	0.017143929	6.234630e-05	strong
BAA_0125	99	213.99131	2986	0.006583582	9.273165e-06	weak
BAA_0126	99	215.08094	2986	0.006583582	9.273165e-06	weak

**id** : identificação do nó

**D** : degree

$s_v$  : weighted degree

$\Delta$  : triangles

$BC(v)$  : bridging coefficient

$C_R(v)$  : bridging centrality

**class** : rótulo atribuído ao nó

As métricas adotadas nesta pesquisa foram escolhidas tendo como critério a complexidade de seus algoritmos geradores, em termos de tempo computacional. Como o cálculo da *Bridging Centrality* depende da métrica *Betweenness Centrality*, a complexidade das métricas que serão usadas pelos algoritmos ML devem ser inferiores a  $\mathcal{O}(mn + n^2 \log n)$ . Sendo assim, foram selecionadas as métricas apresentadas na Tabela 6.

Tabela 6 – Métricas selecionadas para esta pesquisa

Métrica	Complexidade	Referências
Degree Centrality	$\Theta(n^2)$	2.1.1.1
Weighted Degree	$\Theta(n^2)$	2.1.1.2
Triangles	$O(m^{1.408})$	2.1.1.3
Bridging Coefficient	$O(n(\log n)^2)$	2.1.1.5

## 3.2 Fase 2: Aprendizado de Máquina

Uma vez que as métricas exatas foram geradas para cada rede PPI na fase 1, tem início a fase 2.

Cada arquivo CSV contendo as métricas foi convertido para o formato ARFF adequadamente, de tal forma a serem utilizados como entrada pelos algoritmos ML. O critério de seleção desses algoritmos teve como influência inicial os trabalhos desenvolvidos por Ananthasubramanian et al. (2012) e Grando e Lamb (2015) em que foram selecionados RandomForest, REPTree e MultilayerPerceptron. Para atender à hipótese deste trabalho, que busca um algoritmo ML cujo resultado se aproxime do algoritmo exato, aumentou-se a quantidade de algoritmos ML visando aumentar a possibilidade de se encontrar um dentre eles que fosse promissor. O conjunto final de algoritmos é o listado na Tabela 7.

Para cada algoritmo selecionado, promoveu-se um estudo preliminar para identificar a melhor combinação de parâmetros que proporcionasse os melhores resultados tanto na fase de aprendizado quanto na fase de classificação. Os parâmetros selecionados nesse estudo foram, por fim, utilizados nos experimentos realizados, conforme descrito no capítulo 4. O anexo B contém o *script* desenvolvido para a fase de geração dos modelos de aprendizado. Já o anexo C contém o *script* responsável por executar a fase de classificação dos nós.

Tabela 7 – Algoritmos ML selecionados para a pesquisa

Algoritmo	Grupo
KStar	lazy
Bagging	meta
LogitBoost	meta
RandomCommittee	meta
RandomSubSpace	meta
J48	trees
LMT	trees
RandomForest	trees
RandomTree	trees
REPTree	trees
JRip	rules
OneR	rules
PART	rules
MultilayerPerceptron	functions
NaiveBayes	bayes

A Figura 17 ilustra o fragmento do arquivo de atributos da PPI *Bacillus anthrax* obtido a partir do arquivo de métricas, cujo fragmento é listado na Tabela 5. Os atributos *degree*, *weighted.degree*, *triangles* e *bridgingcoefficient* são numéricos. O atributo *class* é do tipo nominal e possui somente dois valores possíveis, *weak* e *strong*. O valor *weak* é atribuído aos nós da rede que possuem valores de *Bridging Centrality* menores ou iguais ao valor do terceiro quartil dessa métrica. Em caso contrário, o valor *strong* é atribuído. Este atributo será utilizado, portanto, como referência para os algoritmos ML. A Tabela 8 apresenta os valores estatísticos da métrica *Bridging Centrality* calculados para a rede PPI *Bacillus anthrax* usando o *software* R<sup>3</sup>.

<sup>3</sup> <https://www.r-project.org>, acessado em: 04/02/2017



```

@RELATION Bacillus_anthrax

@ATTRIBUTE degree NUMERIC
@ATTRIBUTE weighted.degree NUMERIC
@ATTRIBUTE triangles NUMERIC
@ATTRIBUTE bridgingcoefficient NUMERIC
@ATTRIBUTE class {weak,strong}

@DATA
92,168.351391196251,2292,0.00652365788043372,weak
102,188.263587832451,2335,0.00406171081969502,strong
102,181.153216958046,2411,0.00542942594460372,strong
69,113.6333745718,1230,0.010860103468662,strong
87,169.408469796181,2377,0.00890048380121291,weak
91,125.763729333878,1592,0.00551999198817061,strong
58,102.597532868385,1276,0.0184614838908725,weak
89,185.533767580986,2035,0.00778612628707075,weak
53,93.6101894378662,1154,0.0218210452385016,weak
73,167.861898064613,2141,0.0134185704561819,weak
79,178.094688177109,2224,0.00976748257294539,weak
103,217.363220334053,3003,0.00597548210781002,strong
73,172.068564891815,2141,0.0134185704561819,weak
46,79.3072695732117,791,0.0171439289833771,strong
46,79.3509550094604,791,0.0171439289833771,strong
99,213.991309642792,2986,0.00658358243742468,weak
99,215.080943465233,2986,0.00658358243742468,weak
73,176.769653081894,2141,0.0134185704561819,weak
129,266.200419187546,3818,0.00357423110763649,strong
130,263.47812974453,3713,0.00262524345340845,strong
73,178.336834907532,2141,0.0134185704561819,weak
99,217.827030897141,2986,0.00658358243742468,weak
104,226.201410770416,3082,0.00591433608781929,strong

```

Figura 17 – Fragmento do arquivo de métricas exatas da PPI *Bacillus anthrax*.

Tabela 8 – Valores estatísticos da métrica *Bridging Centrality* calculados a partir do arquivo de métricas da PPI *Bacillus anthrax*

Descrição	Valor
Mínimo	0.0000000000
1ºQuartil	0.0000000000
Mediana	0.0000074939
Média	0.0000339336
3ºquartil	0.0000220535
Máximo	0.0016365658

Cada arquivo de métricas é submetido ao processo de aprendizado de máquina por cada um dos algoritmos ML, produzindo como saída 3 arquivos:

- ❑ **modelo.model**: contém o modelo de aprendizado
- ❑ **treinamento.txt**: arquivo de log contendo os dados estatísticos, tanto da fase de treinamento quanto da fase de testes
- ❑ **threshold.arff**: contém os dados da curva ROC, gerados durante o treinamento

A Figura 18 ilustra a curva ROC gerada pelo algoritmo *RandomForest* executado sobre a PPI *Bacillus anthrax*, e a Figura 19 apresenta o arquivo de log do treinamento

correspondente. Pode-se notar que o valor AUC na figura (0.9628) corresponde ao valor AUC do arquivo de log do treinamento (0.963), diferindo por questões de arredondamento. A curva ROC foi adotada como métrica de performance dos algoritmos ML por considerar, em sua composição, a proporção entre as taxas de verdadeiros positivos e falsos positivos. A análise comparativa dos modelos de aprendizado é realizada, portanto, segundo valores AUC, onde o maior deles será o modelo preditivo adotado para a fase seguinte.

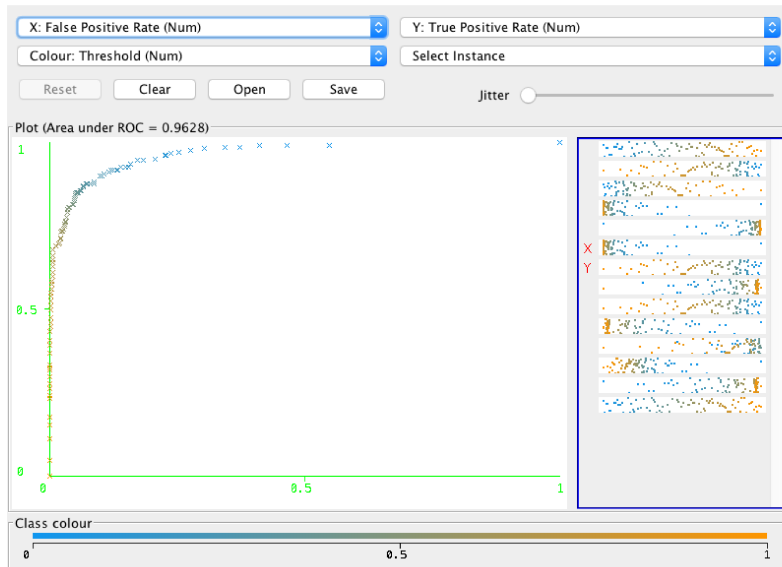


Figura 18 – Curva ROC gerada pelo algoritmo *Random Forest* na fase de teste do treinamento aplicado à PPI *Bacillus anthrax*.

```
Options: -P 100 -I 100 -num-slots 0 -K 0 -M 1 -V 0.0001 -S 20 -num-decimal-places 4

RandomForest

Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 1.0E-4 -S 20 -do-not-check-capabilities -num-decimal-places 4

Time taken to build model: 10.64 seconds
Time taken to test model on training data: 1.5 seconds

=== Error on training data ===

Correctly Classified Instances      792          100    %
Incorrectly Classified Instances      0           0    %
Kappa statistic                     1
Mean absolute error                   0.0466
Root mean squared error               0.0926
Relative absolute error               12.3825 %
Root relative squared error           21.3575 %
Total Number of Instances           792

=== Detailed Accuracy By Class ===
```

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	1,000	0,000	1,000	1,000	1,000	1,000	1,000	1,000	weak
	1,000	0,000	1,000	1,000	1,000	1,000	1,000	1,000	strong
Weighted Avg.	1,000	0,000	1,000	1,000	1,000	1,000	1,000	1,000	

```

=== Confusion Matrix ===
  a   b   <-- classified as
593   0 |   a = weak
  0 199 |   b = strong

=== Stratified cross-validation ===

Correctly Classified Instances      729          92.0455 %
Incorrectly Classified Instances      63          7.9545 %
Kappa statistic                     0.7809
Mean absolute error                   0.1304
Root mean squared error               0.2461
Relative absolute error               34.6191 %
Root relative squared error           56.7483 %
Total Number of Instances           792

=== Detailed Accuracy By Class ===
```

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,965	0,211	0,932	0,965	0,948	0,783	0,963	0,984	weak
	0,789	0,035	0,882	0,789	0,833	0,783	0,963	0,916	strong
Weighted Avg.	0,920	0,167	0,919	0,920	0,919	0,783	0,963	0,967	

```

=== Confusion Matrix ===
  a   b   <-- classified as
572  21 |   a = weak
  42 157 |   b = strong
```

Figura 19 – Arquivo de log do treinamento do algoritmo *RandomForest* aplicado à PPI *Bacillus anthrax*.

### 3.3 Fase 3: Predição

A fase de predição faz uso de 2 entradas: os arquivos de métricas de predição, no formato ARFF, e o melhor modelo de aprendizado identificado na fase 2 da Figura 14.

Os arquivos de métricas de predição ARFF, utilizados nessa fase, correspondem aos arquivos de métricas de aprendizado ARFF utilizados na fase 3, com a diferença que o atributo *class*, em todas as instâncias, possui valor “?”, indicando o que se deseja classificar, ou seja, o valor desse atributo é determinado pelo classificador, dentre as duas possibilidades disponíveis. O melhor modelo de aprendizado, e seu algoritmo, são utilizados como entrada do *script* de classificação (disponível no apêndice C) criado para classificar cada instância segundo as classes *weak* ou *strong*, resultando no arquivo “algoritmo.csv”. A Figura 20 apresenta fragmento do arquivo de métricas de predição utilizado pelo classificador sobre a rede PPI *Bacillus anthrax* e a Tabela 9 apresenta fragmento do arquivo “RandomForest.csv”, resultado da classificação aplicada à rede PPI *Bacillus anthrax*.

```
@RELATION Bacillus_anthrax

@ATTRIBUTE degree NUMERIC
@ATTRIBUTE weighted.degree NUMERIC
@ATTRIBUTE triangles NUMERIC
@ATTRIBUTE bridgingcoefficient NUMERIC
@ATTRIBUTE class {weak,strong}

@DATA

92,168.351391196251,2292,0.00652365788043372,?
102,188.263587832451,2335,0.00406171081969502,?
102,181.153216958046,2411,0.00542942594460372,?
69,113.6333745718,1230,0.010860103468662,?
87,169.408469796181,2377,0.00890048380121291,?
91,125.763729333878,1592,0.00551999198817061,?
58,102.597532868385,1276,0.0184614838908725,?
89,185.533767580986,2035,0.00778612628707075,?
53,93.6101894378662,1154,0.0218210452385016,?
73,167.861898064613,2141,0.0134185704561819,?
79,178.094688177109,2224,0.00976748257294539,?
103,217.363220334053,3003,0.00597548210781002,?
73,172.068564891815,2141,0.0134185704561819,?
46,79.3072695732117,791,0.0171439289833771,?
46,79.3509550094604,791,0.0171439289833771,?
99,213.991309642792,2986,0.00658358243742468,?
99,215.080943465233,2986,0.00658358243742468,?
73,176.769653081894,2141,0.0134185704561819,?
129,266.200419187546,3818,0.00357423110763649,?
130,263.47812974453,3713,0.00262524345340845,?
73,178.336834907532,2141,0.0134185704561819,?
99,217.827030897141,2986,0.00658358243742468,?
104,226.201410770416,3082,0.00591433608781929,?
```

Figura 20 – Fragmento do arquivo de métricas de predição correspondente à PPI *Bacillus anthrax*.

Tabela 9 – Fragmento do arquivo “RandomForest.csv”gerado pelo *script* de predição a partir do arquivo de métricas de predição ARFF correspondente à PPI *Bacillus anthrax*

inst#	actual	predicted	prediction
1	1:?	2:strong	0.94
2	1:?	2:strong	0.94
...	...	...	...
40	1:?	2:strong	0.89
41	1:?	1:weak	0.63
42	1:?	2:strong	0.89
...	...	...	...
48	1:?	2:strong	0.89
49	1:?	2:strong	0.89
52	1:?	1:weak	0.71
...	...	...	...
55	1:?	2:strong	0.92
56	1:?	2:strong	0.94
57	1:?	1:weak	0.62
58	1:?	2:strong	0.94
...	...	...	...
60	1:?	2:strong	0.94
61	1:?	1:weak	0.62
62	1:?	2:strong	0.94
...	...	...	...
82	1:?	2:strong	0.92
83	1:?	1:weak	0.63
84	1:?	2:strong	0.89

**inst#** : número da instância no arquivo de predição ARFF

**actual** : valor atual da classe atribuído à instância

**predicted** : valor predito pelo algoritmo atribuído à instância

**prediction** : valor estimado pelo algoritmo ML da probabilidade de uma instância ser classificada como *strong* ou *weak*, variando entre 0 e 1

Para avaliar a performance do classificador, é utilizada a curva ROC de onde se obtém o valor AUC. Também é utilizada a curva de acurácia, que tem por finalidade determinar o valor do corte limiar (threshold) dos nós classificados, que serão separados segundo os grupos verdadeiro positivos (TP), verdadeiro negativos (TN), falso positivos (FP) e falso negativos (FN). As informações obtidas nesse procedimento serão apresentadas em gráficos, como a Figura 21.

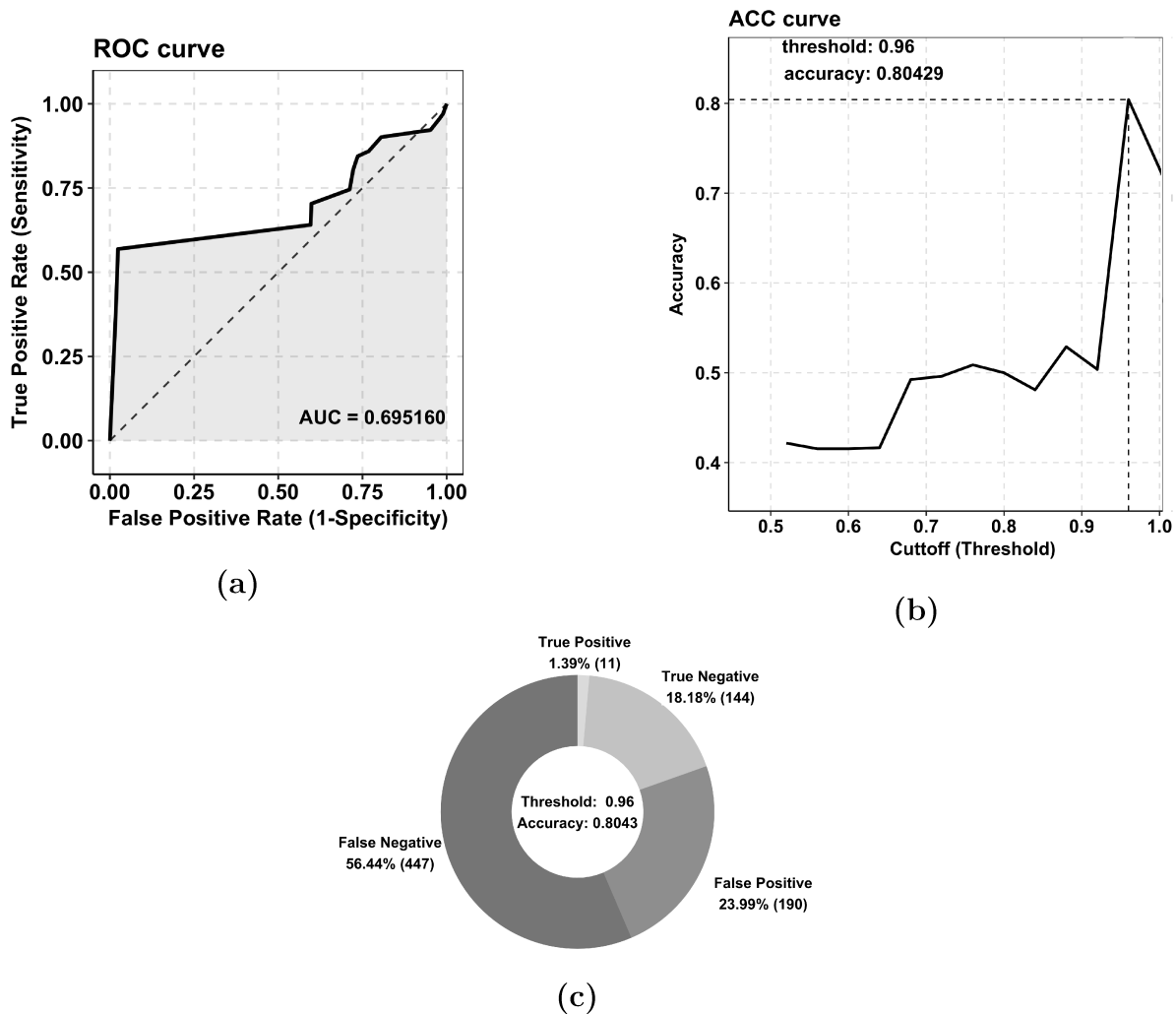


Figura 21 – Exemplo de gráficos de desempenho do modelo preditivo aplicado sobre uma PPI, onde: **(a)** mostra a curva ROC com valor da AUC, **(b)** exibe a curva de acurácia em que se destaca a linha de corte e o valor de threshold, e **(c)** exibe a proporção de verdadeiro positivos (TP), verdadeiro negativos (TN), falso positivos (FP) e falso negativos (FN) para o valor de threshold adotado

Uma vez concluído o processo de predição, passa-se para a fase de conclusão, onde os resultados gerados pelo classificador são comparados aos resultados gerados pelo algoritmo exato.

### 3.4 Fase 4: Conclusão

A fase final, de conclusão, é responsável por avaliar a performance do classificador em comparação ao resultado do algoritmo exato.

Para esta tarefa, todos os nós classificados como verdadeiro positivos (TP) pelo algoritmo preditor, para a classe **strong**, são separados e agrupados por rede PPI. Em seguida, para cada rede PPI, são quantificados os nós **strong** rotulados pelo algoritmo exato e coincidentes com os preditos. Dessa comparação resulta a taxa de performance do preditor. A Tabela 10 apresenta um exemplo dessa análise comparativa onde pode-se notar que a PPI *Bacillus anthrax* teve 7/11 dos nós reais **strong** coincidentes com os nós **strong** preditos, resultando em 63.64% de acerto.

Tabela 10 – Exemplo de comparativo Real vs Predição para o percentual de nós classificados como verdadeiros positivos (TP)

PPI	Weak Real	Strong Real	Strong Predição	Taxa Acerto
<i>Escherichia coli ED1a</i>	0	10	10	100,00
<i>Clostridium tetani E88</i>	1	27	28	96,43
<i>Clostridium perfringens</i>	1	24	25	96,00
<i>Clostridium botulinum A2 Kyoto</i>	2	17	19	89,47
<i>Mycobacterium tuberculosis</i>	7	13	20	65,00
<i>Bacillus anthrax</i>	4	7	11	63,64
<i>Streptococcus pneumoniae Taiwan19F14</i>	5	6	11	54,54
<i>Clostridium botulinum F Langeland</i>	11	12	23	52,17
Média				77,16
Desvio-Padrão				20,23





---

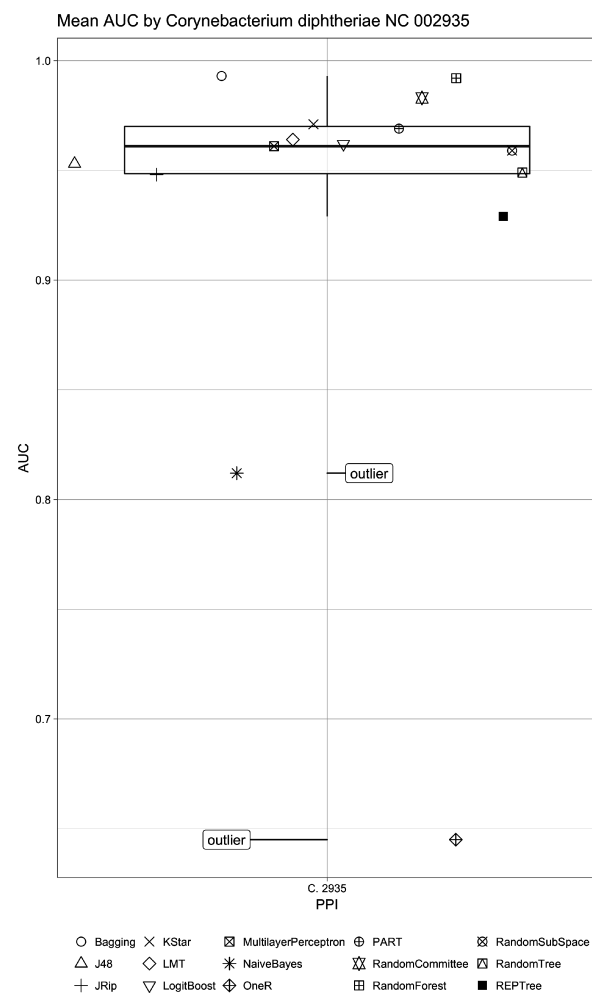
## Experimentos e Análise dos Resultados

As redes PPIs dos organismos selecionados para análises neste trabalho foram submetidas, primeiramente, ao procedimento de aprendizado de máquina (*Machine Learning - ML*) com a finalidade de identificar, entre elas, o melhor modelo preditivo possível, assim como seu algoritmo ML correspondente. Cada algoritmo ML recebeu como entrada uma PPI por vez.

A Tabela (a) da Figura 22 apresenta os valores AUC obtidos no aprendizado da PPI *Corynebacterium diphtheriae* NC 002935, onde pode-se observar que o algoritmo *Bagging* obteve o melhor resultado, com  $AUC = 0,993$  e que o algoritmo *RandomForest* obteve  $AUC = 0,992$ , uma diferença de 0,001. A razão para estes dois algoritmos apresentarem resultados tão próximos é que ambos utilizam o algoritmo *RandomTree* internamente. No caso do *Bagging*, foi-lhe passado explicitamente como parâmetro. Ainda na Tabela (a) da 22, nota-se que os algoritmos obtiveram valor médio de  $AUC = 0,9327$  com desvio-padrão de  $AUC = \pm 0,0903$  e que os algoritmos *OneR* e *NaiveBayes* obtiveram os piores resultados, com valores  $AUC = 0,812$  e  $AUC = 0,645$ , respectivamente. O gráfico (b) da Figura 22 representa o *boxplot* da Tabela (a).

Algoritmo	AUC
<b>Bagging</b>	<b>0,9930</b>
RandomForest	0,9920
RandomCommittee	0,9830
KStar	0,9710
PART	0,9690
LMT	0,9640
LogitBoost	0,9620
MultilayerPerceptron	0,9610
RandomSubSpace	0,9590
J48	0,9530
RandomTree	0,9490
JRip	0,9480
REPTree	0,9290
NaiveBayes	0,8120
OneR	0,6450
Média	0,9327
Desvio Padrão	0,0903

(a)



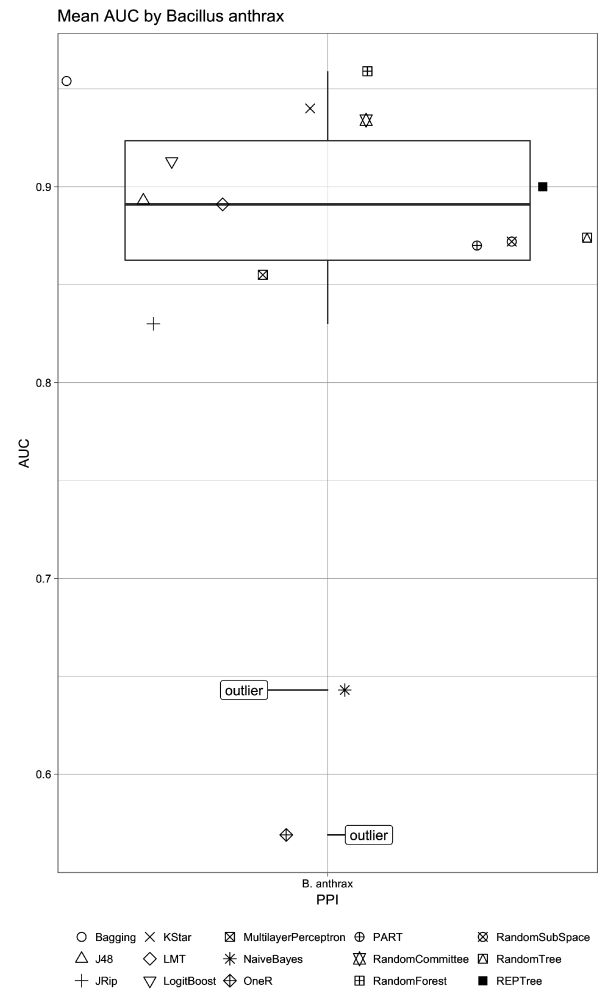
(b)

Figura 22 – Resultado da geração de modelos de classificação da PPI *Corynebacterium diphtheriae* NC 002935, onde (a) apresenta os dados tabulados, e (b) representação gráfica em *boxplot*.

A Tabela (a) da Figura 23 apresenta os resultados obtidos no aprendizado da PPI *Bacillus anthrax*, com valor médio de  $AUC = 0,8598$  e desvio-padrão de  $AUC = \pm 0,1102$ . O algoritmo *RandomForest* obteve o maior valor de  $AUC = 0,959$ , ocupando a primeira posição da tabela, seguido pelo algoritmo *Bagging* com  $AUC = 0,954$ . Novamente, observa-se que os algoritmos *NaiveBayes* e *OneR* obtiveram os piores valores  $AUC = 0,643$  e  $AUC = 0,569$ , respectivamente. O gráfico (b) da Figura 23 representa o *boxplot* da Tabela (a).

Algoritmo	AUC
<b>RandomForest</b>	<b>0,9590</b>
Bagging	0,9540
KStar	0,9400
RandomCommittee	0,9340
LogitBoost	0,9130
REPTree	0,9000
J48	0,8930
LMT	0,8910
RandomTree	0,8740
RandomSubSpace	0,8720
PART	0,8700
MultilayerPerceptron	0,8550
JRip	0,8300
NaiveBayes	0,6430
OneR	0,5690
Média	0,8598
Desvio-Padrão	0,1102

(a)



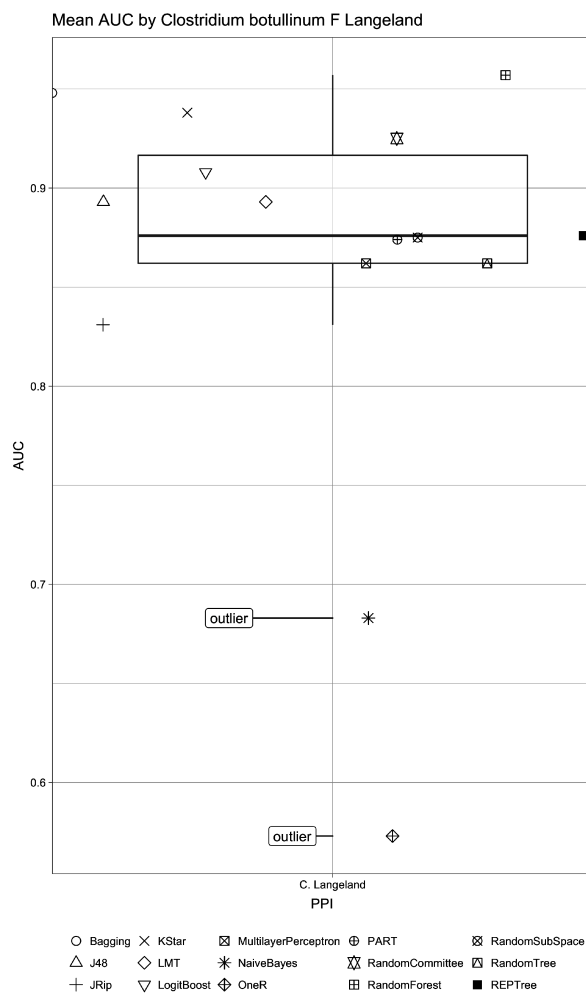
(b)

Figura 23 – Resultado da geração de modelos de classificação da PPI *Bacillus anthrax*, onde (a) apresenta os dados tabulados, e (b) representação gráfica em *boxplot*.

A Tabela (a) da Figura 24 corresponde aos dados obtidos no treinamento da PPI *Clostridium botulinum F Langeland*, onde o valor médio de AUC foi de 0,8599 com desvio-padrão de  $\pm 0,1025$ . Novamente, o algoritmo *RandomForest* obteve o maior valor de AUC = 0,957, seguido pelo algoritmo *Bagging* com AUC = 0,948. Os algoritmos *NaiveBayes* e *OneR* mantiveram os piores valores AUC = 0,683 e AUC = 0,573, respectivamente. O gráfico (b) da Figura 24 representa o *boxplot* da Tabela (a).

Algoritmo	AUC
<b>RandomForest</b>	<b>0,9570</b>
Bagging	0,9480
KStar	0,9380
RandomCommittee	0,9250
LogitBoost	0,9080
J48	0,8930
LMT	0,8930
REPTree	0,8760
RandomSubSpace	0,8750
PART	0,8740
RandomTree	0,8620
MultilayerPerceptron	0,8620
JRip	0,8310
NaiveBayes	0,6830
OneR	0,5730
Média	0,8599
Desvio-Padrão	0,1025

(a)



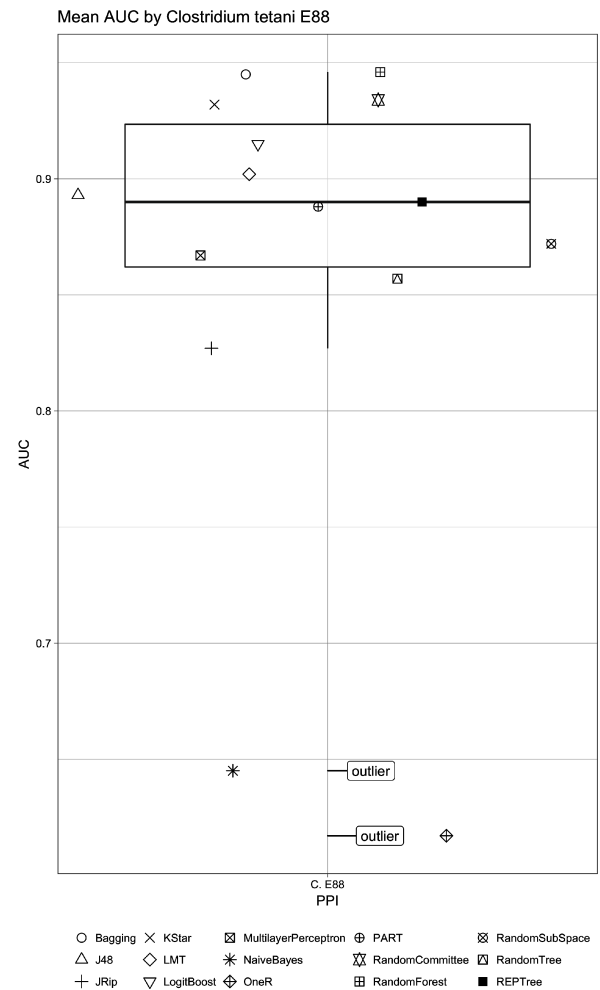
(b)

Figura 24 – Resultado da geração de modelos de classificação da PPI *Clostridium botulinum F Langeland*, onde (a) dados tabulados, e (b) representação gráfica em boxplot.

O aprendizado sobre a PPI *Clostridium tetani* E88 produziu como resultado a Tabela (a) da Figura 25 de onde se observa o valor médio de  $AUC = 0,862$  e desvio-padrão de  $AUC = \pm 0,0998$ . O algoritmo *RandomForest* permanece na primeira posição, com o maior valor de  $AUC = 0,946$ , seguido pelo algoritmo *Bagging* com  $AUC = 0,945$ , uma diferença de 0,001. Novamente, observa-se que os algoritmos *NaiveBayes* e *OneR* obtiveram os piores valores  $AUC = 0,645$  e  $AUC = 0,617$ , respectivamente. O gráfico (b) da Figura 25 representa o *boxplot* da Tabela (a).

Algoritmo	AUC
<b>RandomForest</b>	<b>0,9460</b>
Bagging	0,9450
RandomCommittee	0,9340
KStar	0,9320
LogitBoost	0,9150
LMT	0,9020
J48	0,8930
REPTree	0,8900
PART	0,8880
RandomSubSpace	0,8720
MultilayerPerceptron	0,8670
RandomTree	0,8570
JRip	0,8270
NaiveBayes	0,6450
OneR	0,6170
Média	0,8620
Desvio-Padrão	0,0998

(a)



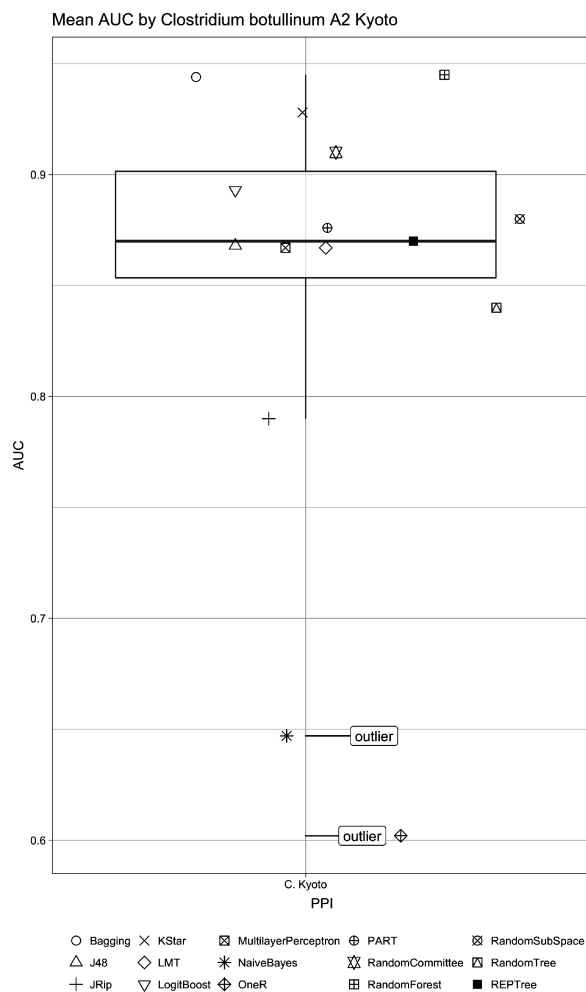
(b)

Figura 25 – Resultado da geração de modelos de classificação da PPI *Clostridium tetani* E88, onde (a) dados tabulados, e (b) representação gráfica em boxplot.

A Tabela (a) da Figura 26 concentra os resultados gerados durante o aprendizado da PPI *Clostridium botulinum A2 Kyoto*, onde foram obtidos o valor médio de  $AUC = 0,8485$  e desvio-padrão de  $AUC = \pm 0,0995$ . O algoritmo *RandomForest* ocupa a primeira posição na tabela, com valor de  $AUC = 0,945$ . A segunda posição é ocupada pelo algoritmo *Bagging* com  $AUC = 0,944$ , novamente com uma diferença de  $0,001$ . Observa-se, também, que os algoritmos *NaiveBayes* e *OneR* mantiveram as ultimas posições, com os piores valores  $AUC = 0,647$  e  $AUC = 0,602$ , respectivamente. O gráfico (b) da Figura 26 representa o *boxplot* da Tabela (a).

Algoritmo	AUC
<b>RandomForest</b>	<b>0,9450</b>
Bagging	0,9440
KStar	0,9280
RandomCommittee	0,9100
LogitBoost	0,8930
RandomSubSpace	0,8800
PART	0,8760
REPTree	0,8700
J48	0,8680
LMT	0,8670
MultilayerPerceptron	0,8670
RandomTree	0,8400
JRip	0,7900
NaiveBayes	0,6470
OneR	0,6020
Média	0,8485
Desvio-Padrão	0,0995

(a)



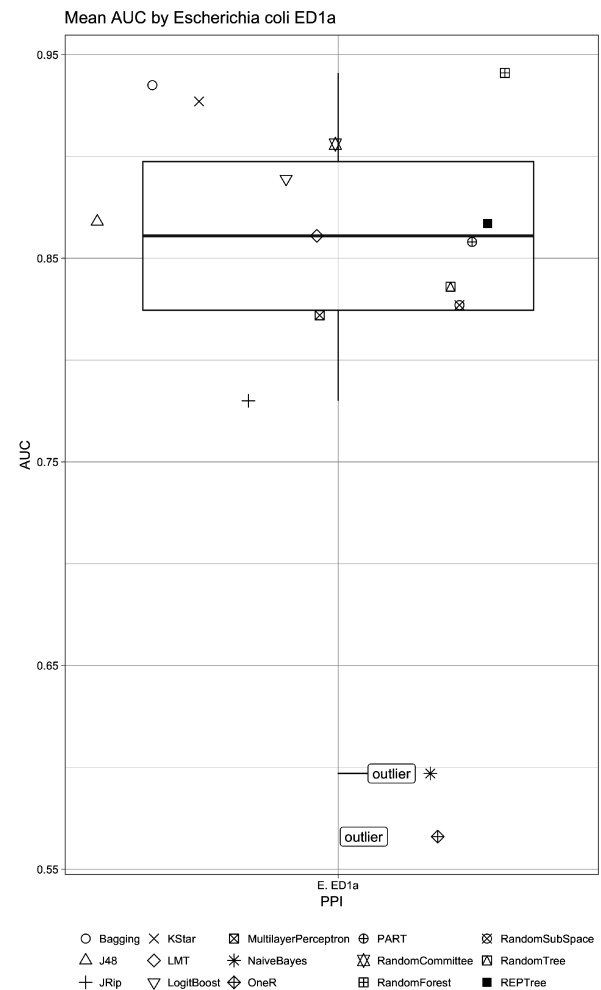
(b)

Figura 26 – Resultado da geração de modelos de classificação da PPI *Clostridium botulinum A2 Kyoto*, onde (a) dados tabulados, e (b) representação gráfica em boxplot.

A Tabela (a) da Figura 27 apresenta os resultados obtidos no aprendizado da PPI *Escherichia coli* ED1a, com valor médio de  $AUC = 0,8320$  e desvio-padrão de  $AUC = \pm 0,1112$ . O algoritmo *RandomForest* obteve o maior valor de  $AUC = 0,941$ , ocupando a primeira posição da tabela, seguido pelo algoritmo *Bagging* com  $AUC = 0,935$ , com uma diferença de 0.006. Novamente, observa-se que os algoritmos *NaiveBayes* e *OneR* obtiveram os piores valores  $AUC = 0,597$  e  $AUC = 0,566$ , respectivamente. O algoritmo KStar permanece na terceira posição na tabela, com valor  $AUC = 0,927$ . O gráfico (b) da Figura 27 representa o *boxplot* da Tabela (a).

Algoritmo	AUC
<b>RandomForest</b>	<b>0,9410</b>
Bagging	0,9350
KStar	0,9270
RandomCommittee	0,9060
LogitBoost	0,8890
J48	0,8680
REPTree	0,8670
LMT	0,8610
PART	0,8580
RandomTree	0,8360
RandomSubSpace	0,8270
MultilayerPerceptron	0,8220
JRip	0,7800
NaiveBayes	0,5970
OneR	0,5660
Média	0,8320
Desvio-Padrão	0,1112

(a)



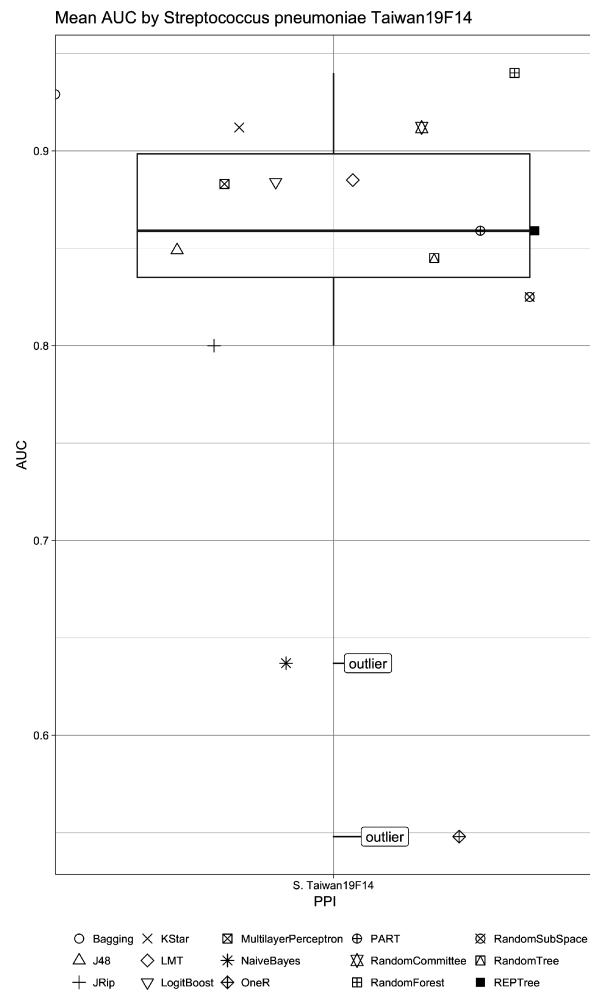
(b)

Figura 27 – Resultado da geração de modelos de classificação da PPI *Escherichia coli* ED1a, onde (a) dados tabulados, e (b) representação gráfica em boxplot.

A Tabela (a) da Figura 28 agrupa os resultados obtidos no aprendizado da PPI *Streptococcus pneumoniae Taiwan19F14*, com valor médio de  $AUC = 0,8378$  e desvio-padrão de  $AUC = \pm 0,108$ . O algoritmo *RandomForest* mantém-se com o maior valor de  $AUC = 0,940$ , ocupando a primeira posição da tabela, seguido pelo algoritmo *Bagging* com  $AUC = 0,929$ . Os algoritmos *NaiveBayes* e *OneR* obtiveram, novamente, os piores valores  $AUC = 0,637$  e  $AUC = 0,548$ , respectivamente. O algoritmo *KStar* ocupa a terceira posição na tabela, com valor  $AUC = 0,912$ . O gráfico (b) da Figura 28 representa o *boxplot* da Tabela (a).

Algoritmo	AUC
<b>RandomForest</b>	<b>0.9400</b>
Bagging	0,9290
KStar	0,9120
RandomCommittee	0,9120
LMT	0,8850
LogitBoost	0,8840
MultilayerPerceptron	0,8830
REPTree	0,8590
PART	0,8590
J48	0,8490
RandomTree	0,8450
RandomSubSpace	0,8250
JRip	0,8000
NaiveBayes	0,6370
OneR	0,5480
Média	0,8378
Desvio-Padrão	0,1080

(a)



(b)

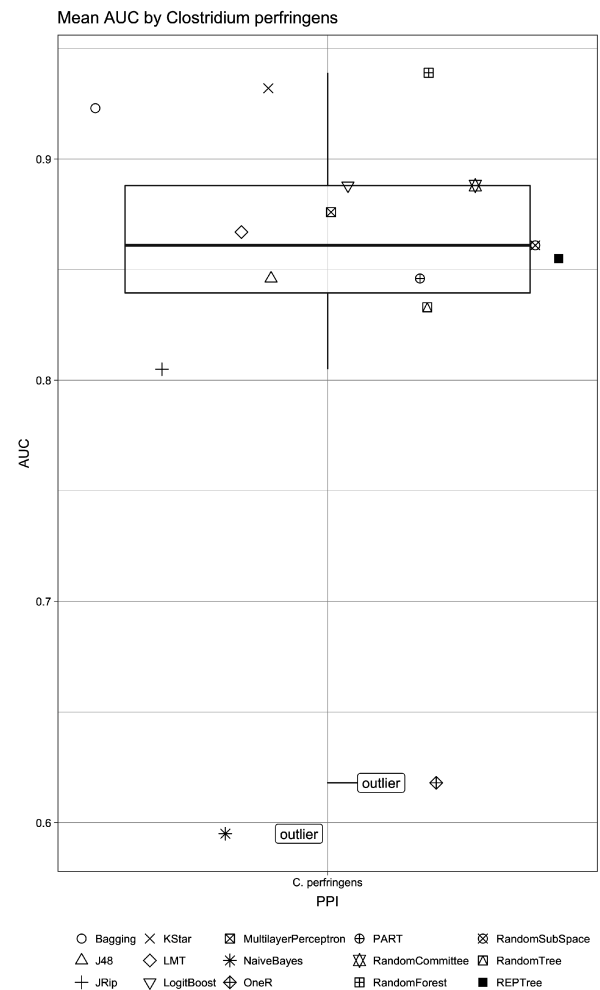
Figura 28 – Resultado da geração de modelos de classificação da PPI *Streptococcus pneumoniae Taiwan19F14*, onde (a) dados tabulados, e (b) representação gráfica em boxplot.



A Tabela (a) da Figura 29 apresenta os resultados obtidos no aprendizado da PPI *Clostridium perfringens*, com valor médio de  $AUC = 0,8381$  e desvio-padrão de  $AUC = \pm 0,1011$ . O algoritmo *RandomForest* obteve o maior valor de  $AUC = 0,939$ , ocupando a primeira posição da tabela. O algoritmo *KStar* com  $AUC = 0,932$  ocupa a segunda posição na tabela. O algoritmo *Bagging* passou para a terceira posição na tabela, com valor  $AUC = 0,923$ . O algoritmo *OneR* ocupa a décima quarta posição, com  $AUC = 0,618$ . O algoritmo *NaiveBayes* ocupa a ultima posição, com  $AUC = 0,595$ . O gráfico (b) da Figura 29 representa o *boxplot* da Tabela (a).

Algoritmo	AUC
<b>RandomForest</b>	<b>0,9390</b>
KStar	0,9320
Bagging	0,9230
LogitBoost	0,8880
RandomCommittee	0,8880
MultilayerPerceptron	0,8760
LMT	0,8670
RandomSubSpace	0,8610
REPTree	0,8550
J48	0,8460
PART	0,8460
RandomTree	0,8330
JRip	0,8050
OneR	0,6180
NaiveBayes	0,5950
Média	0,8381
Desvio-Padrão	0,1011

(a)



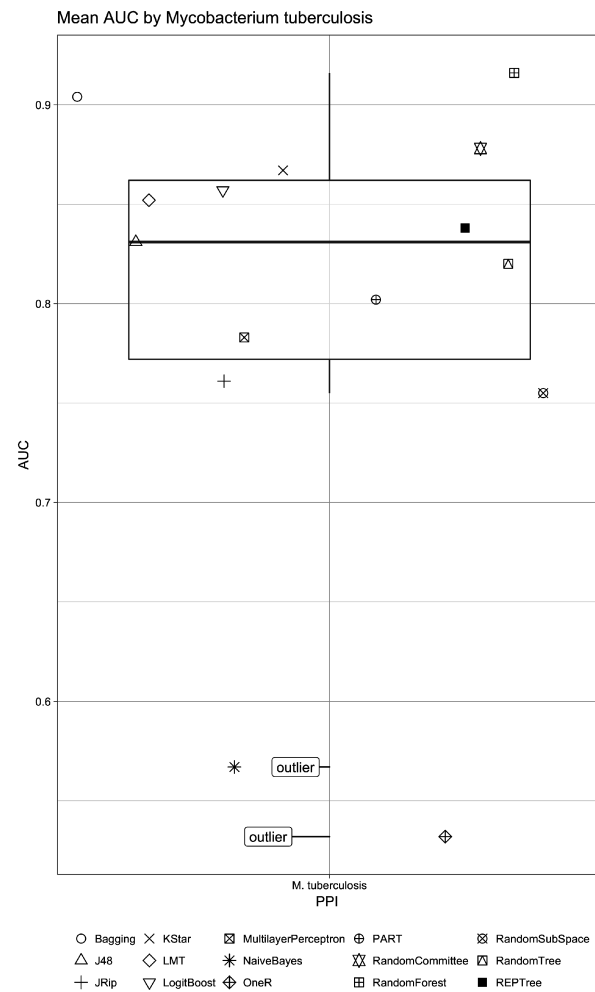
(b)

Figura 29 – Resultado da geração de modelos de classificação da PPI *Clostridium perfringens*, onde (a) dados tabulados, e (b) representação gráfica em boxplot.

A Tabela (a) da Figura 30 apresenta os resultados obtidos no aprendizado da PPI *Mycobacterium tuberculosis*, com valor médio de  $AUC = 0,7975$  e desvio-padrão de  $AUC = \pm 0,1113$ . O algoritmo *RandomForest* mantém-se com o maior valor de  $AUC = 0,916$ , ocupando a primeira posição da tabela, seguido pelo algoritmo *Bagging* com  $AUC = 0,904$ . O algoritmo *RandomCommittee* ocupa a terceira posição na tabela, com valor  $AUC = 0,878$ . Os algoritmos *NaiveBayes* e *OneR* obtiveram os piores valores  $AUC = 0,567$  e  $AUC = 0,532$ , respectivamente. O gráfico (b) da Figura 30 representa o *boxplot* da Tabela (a).

Algoritmo	AUC
<b>RandomForest</b>	<b>0,9160</b>
Bagging	0,9040
RandomCommittee	0,8780
KStar	0,8670
LogitBoost	0,8570
LMT	0,8520
REPTree	0,8380
J48	0,8310
RandomTree	0,8200
PART	0,8020
MultilayerPerceptron	0,7830
JRip	0,7610
RandomSubSpace	0,7550
NaiveBayes	0,5670
OneR	0,5320
Média	0,7975
Desvio-Padrão	0,1113

(a)



(b)

Figura 30 – Resultado da geração de modelos de classificação da PPI *Mycobacterium tuberculosis*, onde (a) dados tabulados, e (b) representação gráfica em *boxplot*.

A Tabela 11 consolida os maiores valores AUC gerados no aprendizado das PPIs, juntamente com seus algoritmos geradores. Observa-se que o algoritmo Bagging foi o que obteve o melhor resultado geral, com valor  $AUC = 0,993$ , fazendo da PPI *Corynebacterium diphtheriae NC 002935* o modelo preditivo procurado. Observa-se, ainda, que o algoritmo RandomForest foi responsável pelos maiores valores AUC para as demais PPIs (o apêndice D apresenta resultados de experimentos adicionais em que o algoritmo preditor utilizado foi o RandomForest). Em média, o valor AUC foi de 0,9484 com desvio-padrão de  $\pm 0,0208$ . No trabalho realizado por Emamjomeh et al. (2014) e Ananthasubramanian et al. (2012), o algoritmo *RandomForest* foi o que obteve melhores resultados, no entanto, neste trabalho o algoritmo *Bagging* foi o que apresentou a melhor performance, indicando que este algoritmo também pode ser usado em trabalhos futuros.

Tabela 11 – Modelos de aprendizado das redes PPI, seus respectivos algoritmos geradores e os valores de área sob a curva ROC (AUC)

PPI	Algoritmo	AUC
<b><i>Corynebacterium diphtheriae NC 002935</i></b>	<b>Bagging</b>	<b>0,993</b>
<i>Bacillus anthrax</i>	RandomForest	0,959
<i>Clostridium botulinum F Langeland</i>	RandomForest	0,957
<i>Clostridium tetani E88</i>	RandomForest	0,946
<i>Clostridium botulinum A2 Kyoto</i>	RandomForest	0,945
<i>Escherichia coli ED1a</i>	RandomForest	0,941
<i>Streptococcus pneumoniae Taiwan19F14</i>	RandomForest	0,940
<i>Clostridium perfringens</i>	RandomForest	0,939
<i>Mycobacterium tuberculosis</i>	RandomForest	0,916
Média		0,9484
Desvio-Padrão		0,0208

A Figura 31 apresenta o gráfico de *boxplot* consolidado, onde pode-se notar que a PPI *Corynebacterium diphtheriae NC 002935* obteve a maior média de AUC e que os algoritmos Bagging e RandomForest foram os melhores colocados, ao contrário dos algoritmos OneR e NaiveBayes que obtiveram os piores resultados.

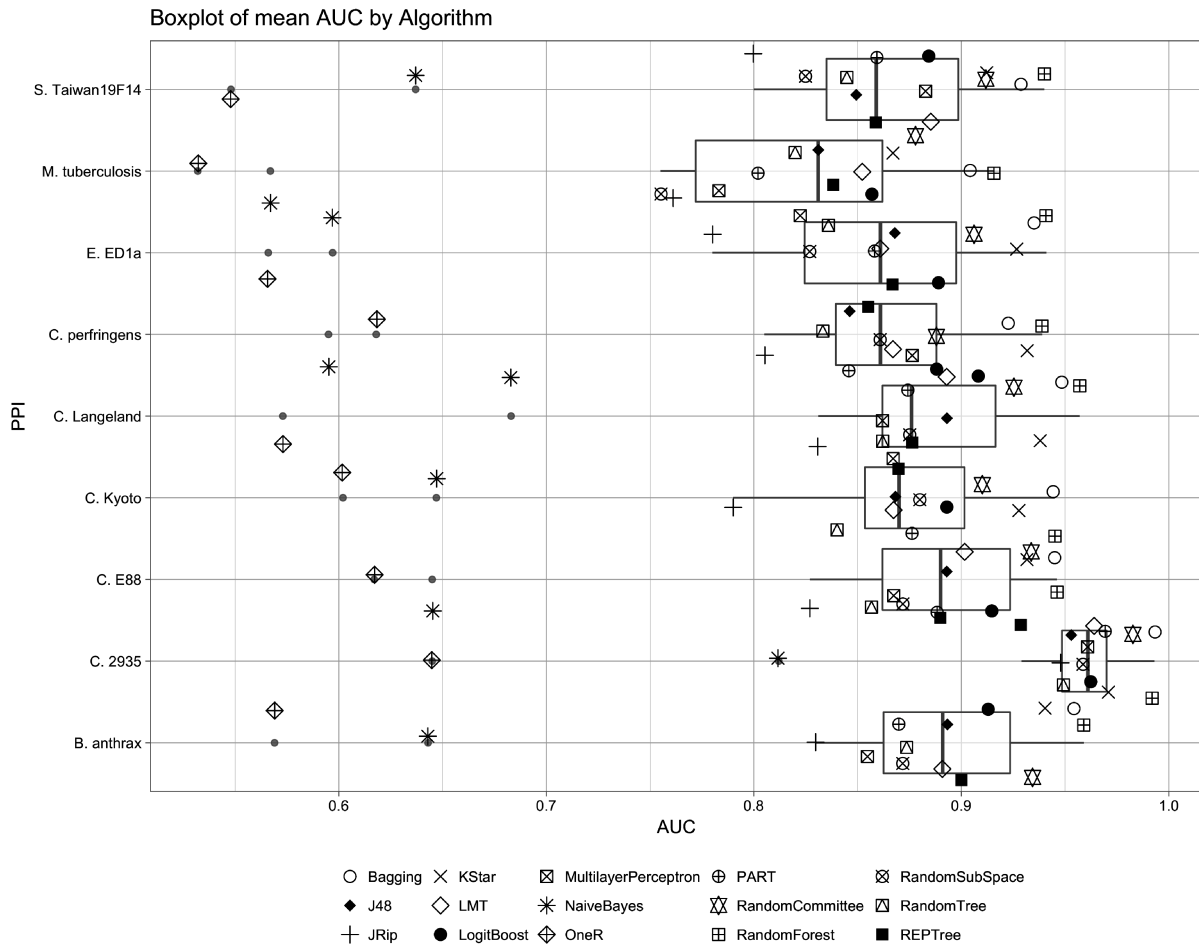


Figura 31 – Gráfico consolidado de boxplots, agrupado por PPI, exibindo a dispersão dos resultados obtidos pelos algoritmos estudados.

## 4.1 Aplicação do modelo preditivo

Uma vez identificado e selecionado o modelo preditivo, juntamente com seu algoritmo ML, as PPIs foram submetidas ao processo de classificação de seus nós, uma por vez.

A PPI *Clostridium botulinum* A2 Kyoto foi submetida ao classificador Bagging, resultando em 146 (22,15%) nós corretamente classificados, sendo 19 (2,88%) verdadeiro positivos (True Positive (TP)) e 127 (19,27%) verdadeiro negativos (True Negative (TN)), contra 513 (77,84%) nós incorretamente classificados, onde 173 (26,25%) nós foram falso positivos (False Positive (FP)) e 340 (51,59%) falso negativos (False Negative (FN)). A acurácia (*accuracy*) da predição foi de 77,85% com limiar (*threshold*) de 96,00%. O valor da AUC foi de 0,736620. A Figura 32 apresenta a curva ROC, a curva de acurácia correspondente, assim como a distribuição das quantidades preditas em termos de TP, TN, FP, FN.

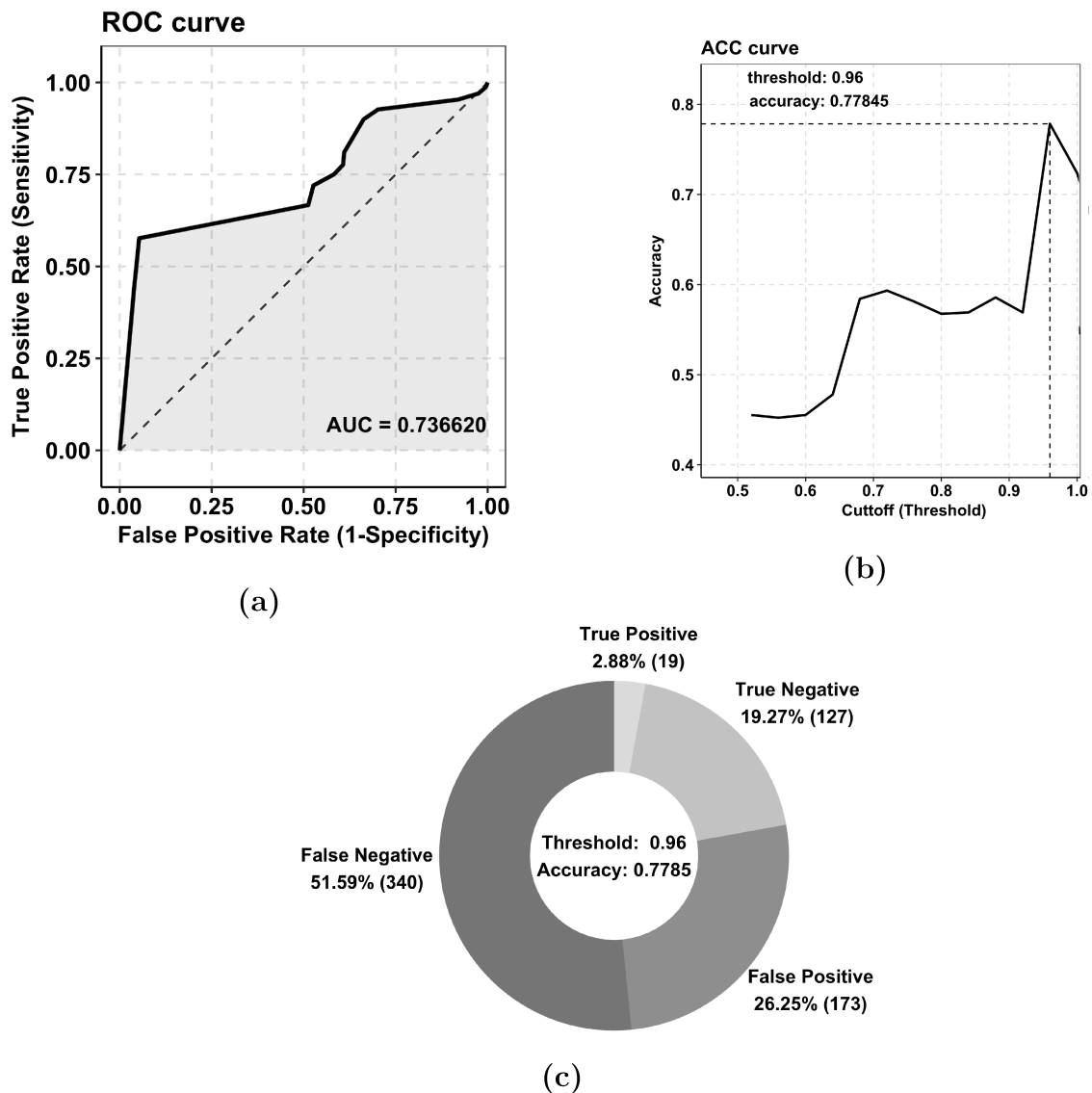


Figura 32 – Desempenho do modelo preditivo aplicado sobre a PPI *Clostridium botulinum* A2 Kyoto, sendo que (a) apresenta a curva ROC com AUC = 0,736620, (b) exibe a curva de acurácia em que destaca o valor de corte threshold = 0,96, e (c) exibe o percentual de verdadeiro positivos (TP), verdadeiro negativos (TN), falso positivos (FP) e falso negativos (FN) para o valor de threshold = 0,96

A classificação dos nós da PPI *Bacillus anthrax* pelo classificador Bagging resultou em 155 (19,57%) nós corretamente classificados, dos quais 11 (1,39%) foram verdadeiro positivos e 144 (18,18%) foram verdadeiro negativos, contra 637 (80,43%) nós incorretamente classificados, dos quais 190 (23,99%) nós foram falso positivos e 447 (56,44%) nós falso negativos. A acurácia dessa predição foi de 80,43% com limiar de 96,00%. O valor da AUC foi de 0,695160. A Figura 33 apresenta a curva ROC assim como a distribuição das quantidades preditas em termos de TP, TN, FP, FN.

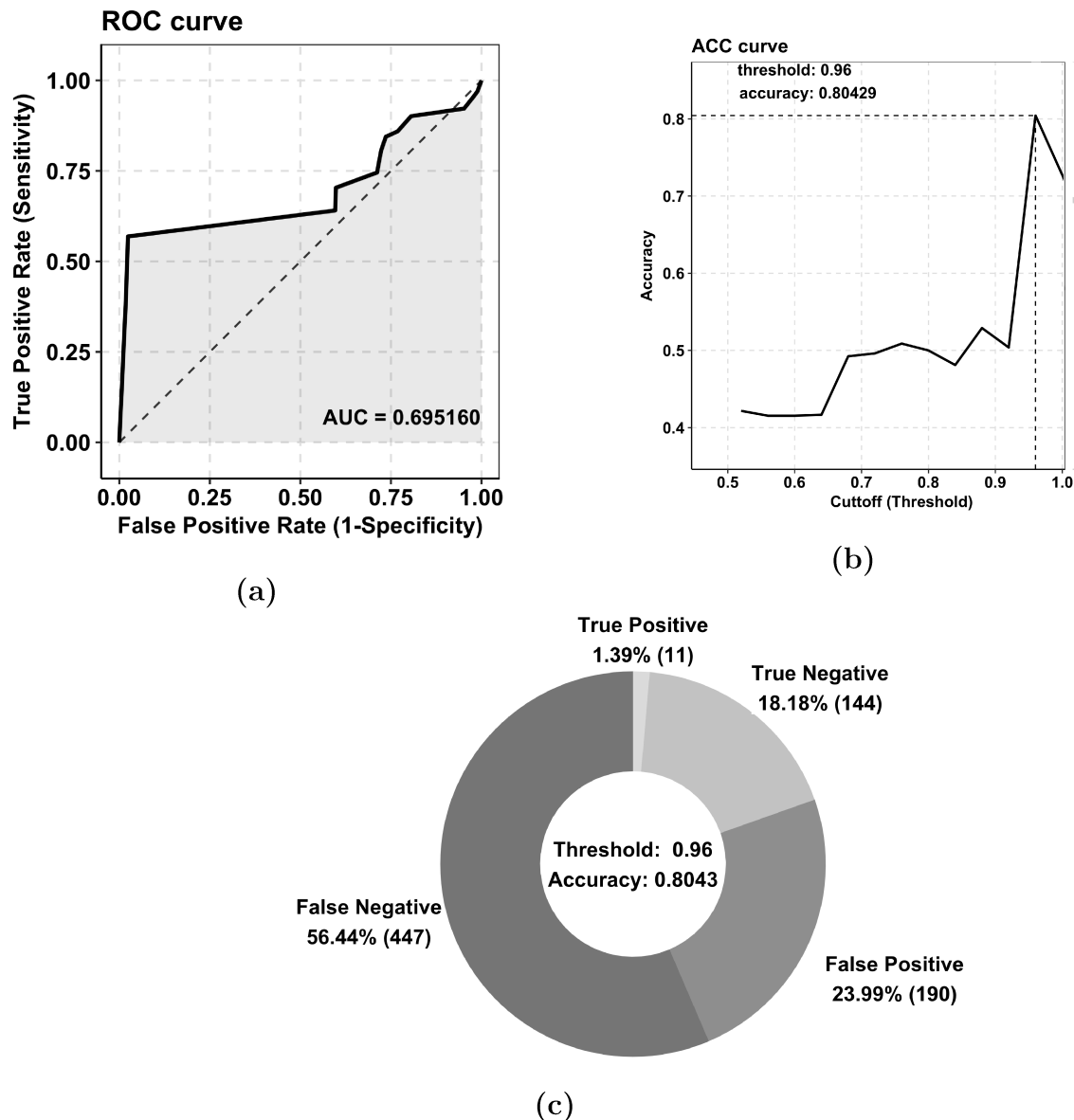


Figura 33 – Desempenho do modelo preditivo aplicado sobre a PPI *Bacillus anthrax*, sendo que (a) apresenta a curva ROC com  $AUC = 0,695160$ , (b) exibe a curva de acurácia em que destaca o valor de corte threshold = 0,96, e (c) exibe o percentual de verdadeiro positivos (TP), verdadeiro negativos (TN), falso positivos (FP) e falso negativos (FN) para o valor de threshold = 0,96

A classificação dos nós da PPI *Mycobacterium tuberculosis* pelo classificador Bagging resultou em 131 (19,46%) nós corretamente classificados, dos quais 20 (2,97%) foram verdadeiro positivos e 111 (16,49%) foram verdadeiro negativos, contra 542 (80,54%) nós incorretamente classificados, dos quais 160 (23,77%) nós foram falso positivos e 382 (56,76%) nós falso negativos. A acurácia dessa predição foi de 80,53% com limiar de 96,00%. O valor da AUC foi de 0,662467. A Figura 34 apresenta a curva ROC assim como a distribuição das quantidades preditas em termos de TP, TN, FP, FN.

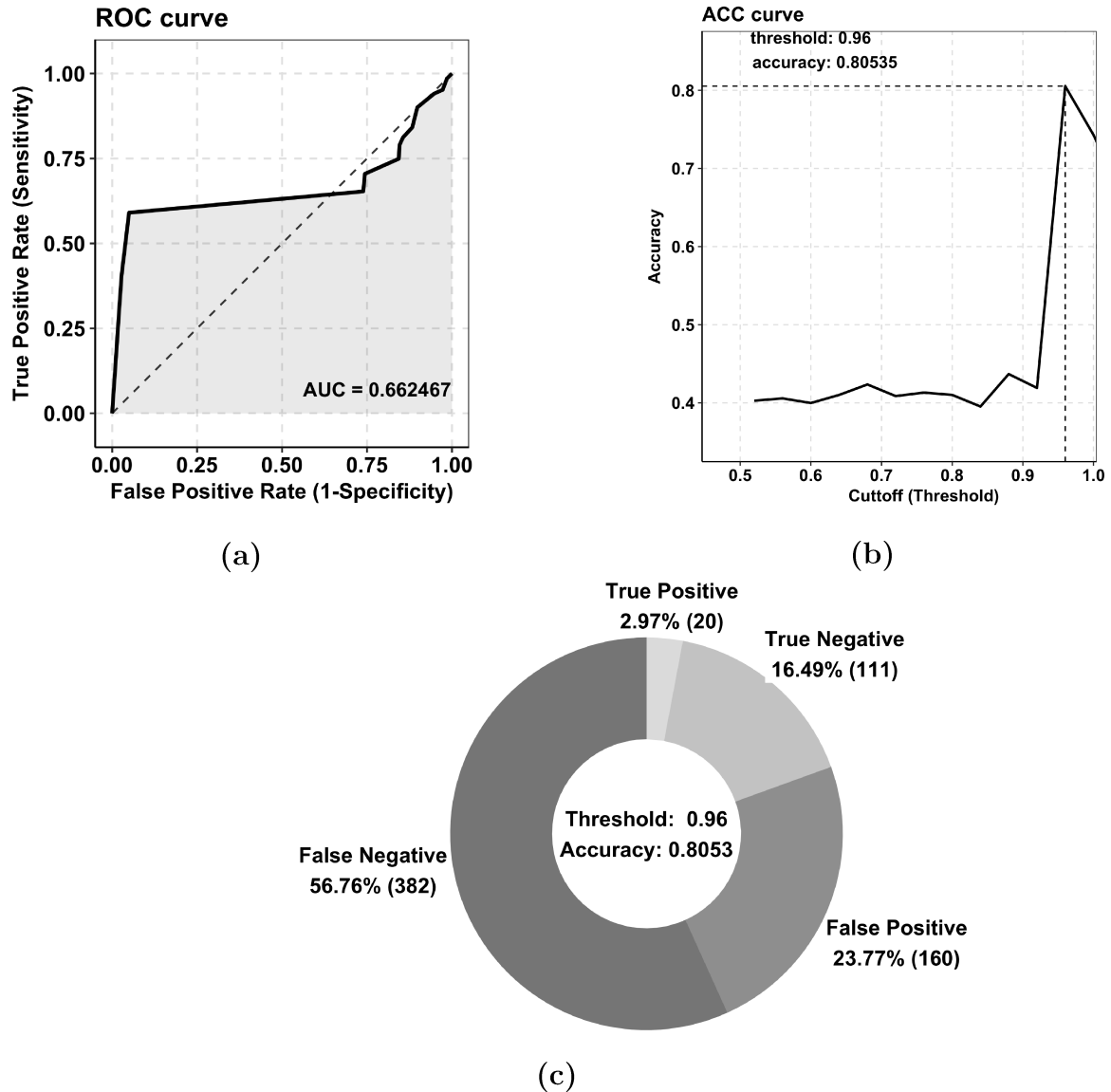


Figura 34 – Desempenho do modelo preditivo aplicado sobre a PPI *Mycobacterium tuberculosis*, sendo que (a) apresenta a curva ROC com  $AUC = 0,662467$ , (b) exibe a curva de acurácia em que destaca o valor de corte threshold = 0,96, e (c) exibe o percentual de verdadeiro positivos (TP), verdadeiro negativos (TN), falso positivos (FP) e falso negativos (FN) para o valor de threshold = 0,96

A classificação dos nós da PPI *Clostridium botulinum* *F Langeland* pelo classificador Bagging resultou em 184 (27,92%) nós corretamente classificados, dos quais 23 (3,49%) foram verdadeiro positivos e 161 (24,43%) foram verdadeiro negativos, contra 475 (72,08%) nós incorretamente classificados, dos quais 143 (21,7%) nós foram falso positivos e 332 (50,38%) nós falso negativos. A acurácia dessa predição foi de 72,08% com limiar de 96,00%. O valor da AUC foi de 0,639140. A Figura 35 apresenta a curva ROC assim como a distribuição das quantidades preditas em termos de TP, TN, FP, FN.

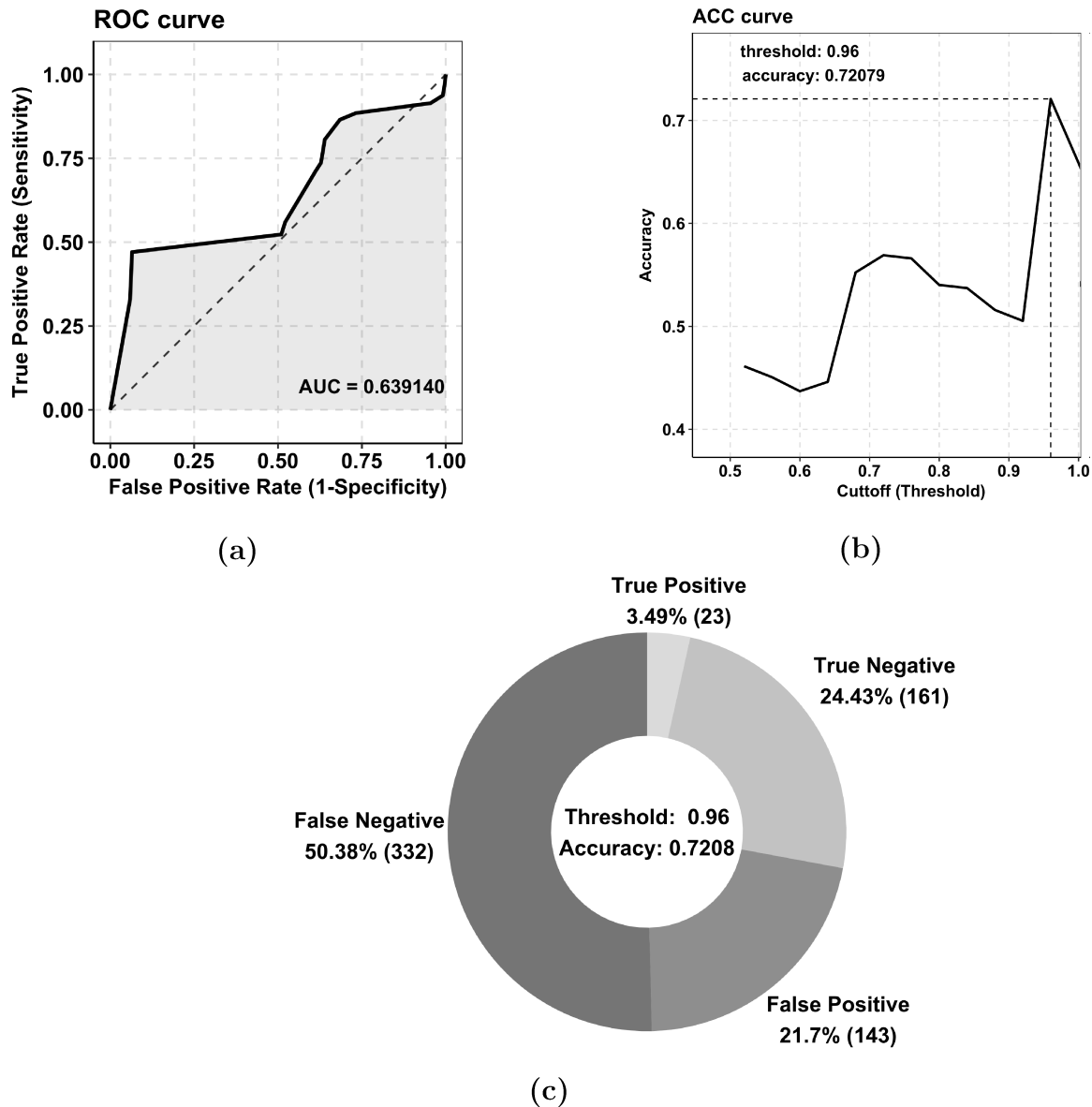


Figura 35 – Desempenho do modelo preditivo aplicado sobre a PPI *Clostridium botulinum* *F Langeland*, sendo que (a) apresenta a curva ROC com  $AUC = 0,639140$ , (b) exibe a curva de acurácia em que destaca o valor de corte  $\text{threshold} = 0,96$ , e (c) exibe o percentual de verdadeiro positivos (TP), verdadeiro negativos (TN), falso positivos (FP) e falso negativos (FN) para o valor de  $\text{threshold} = 0,96$



A classificação dos nós da PPI *Clostridium perfringens* pelo classificador Bagging resultou em 165 (30,05%) nós corretamente classificados, dos quais 25 (4,55%) foram verdadeiro positivos e 140 (25,50%) foram verdadeiro negativos, contra 384 (69,95%) nós incorretamente classificados, dos quais 173 (31,51%) nós foram falso positivos e 211 (38,43%) nós falso negativos. A acurácia dessa predição foi de 69,95% com limiar de 96,00%. O valor da AUC foi de 0,627972. A Figura 36 apresenta a curva ROC assim como a distribuição das quantidades preditas em termos de TP, TN, FP, FN.

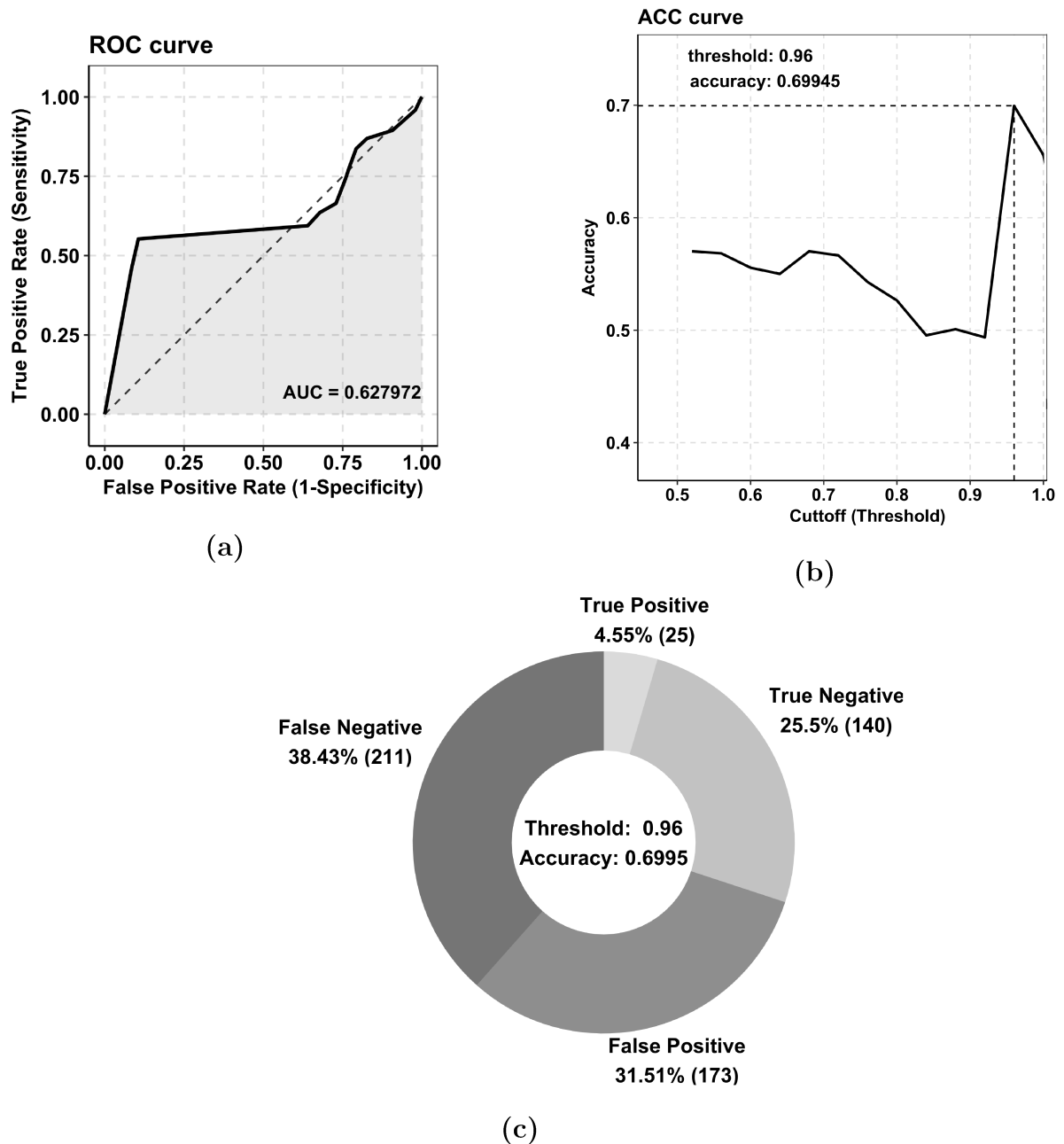


Figura 36 – Desempenho do modelo preditivo aplicado sobre a PPI *Clostridium perfringens*, sendo que (a) apresenta a curva ROC com  $AUC = 0,627972$ , (b) exibe a curva de acurácia em que destaca o valor de corte threshold = 0,96, e (c) exibe o percentual de verdadeiro positivos (TP), verdadeiro negativos (TN), falso positivos (FP) e falso negativos (FN) para o valor de threshold = 0,96

A classificação dos nós da PPI *Clostridium tetani* E88 pelo classificador Bagging resultou em 149 (33,04%) nós corretamente classificados, dos quais 28 (6,21%) foram verdadeiro positivos e 121 (26,83%) foram verdadeiro negativos, contra 302 (66,96%) nós incorretamente classificados, dos quais 125 (27,72%) nós foram falso positivos e 177 (39,25%) nós falso negativos. A acurácia dessa predição foi de 66,96% com limiar de 96,00%. O valor da AUC foi de 0,619899. A Figura 37 apresenta a curva ROC assim como a distribuição das quantidades preditas em termos de TP, TN, FP, FN.

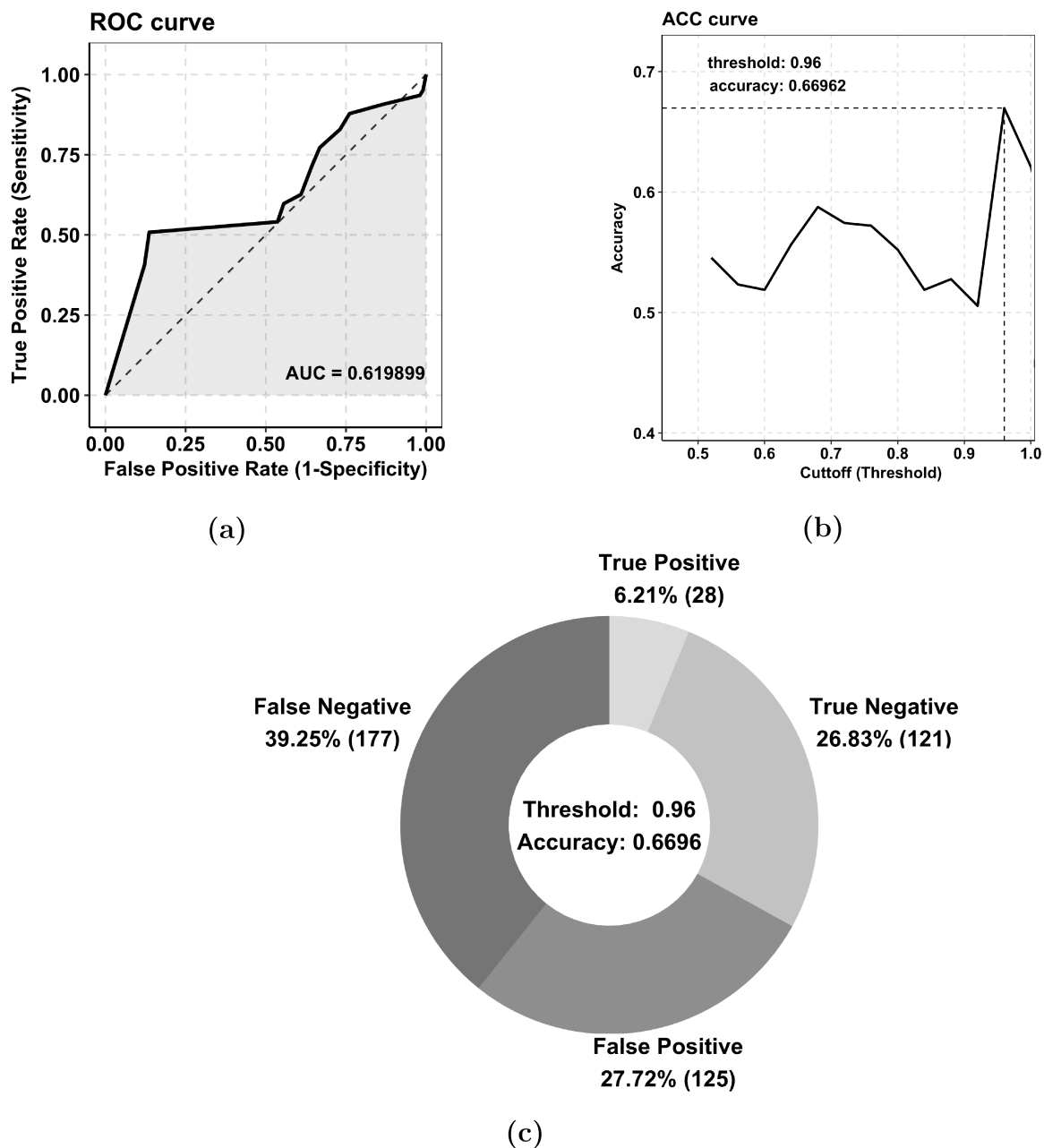


Figura 37 – Desempenho do modelo preditivo aplicado sobre a PPI *Clostridium tetani* E88, sendo que (a) apresenta a curva ROC com  $AUC = 0,619899$ , (b) exibe a curva de acurácia em que destaca o valor de corte threshold = 0,96, e (c) exibe o percentual de verdadeiro positivos (TP), verdadeiro negativos (TN), falso positivos (FP) e falso negativos (FN) para o valor de threshold = 0,96

A classificação dos nós da PPI *Streptococcus pneumoniae* Taiwan19F14 pelo classificador Bagging resultou em 140 (32,33%) nós corretamente classificados, dos quais 11 (2,54%) foram verdadeiro positivos e 129 (29,79%) foram verdadeiro negativos, contra 293 (67,67%) nós incorretamente classificados, dos quais 98 (22,63%) nós foram falso positivos e 195 (45,03%) nós falso negativos. A acurácia dessa predição foi de 67,67% com limiar de 96,00%. O valor da AUC foi de 0,618804. A Figura 38 apresenta a curva ROC assim como a distribuição das quantidades preditas em termos de TP, TN, FP, FN.

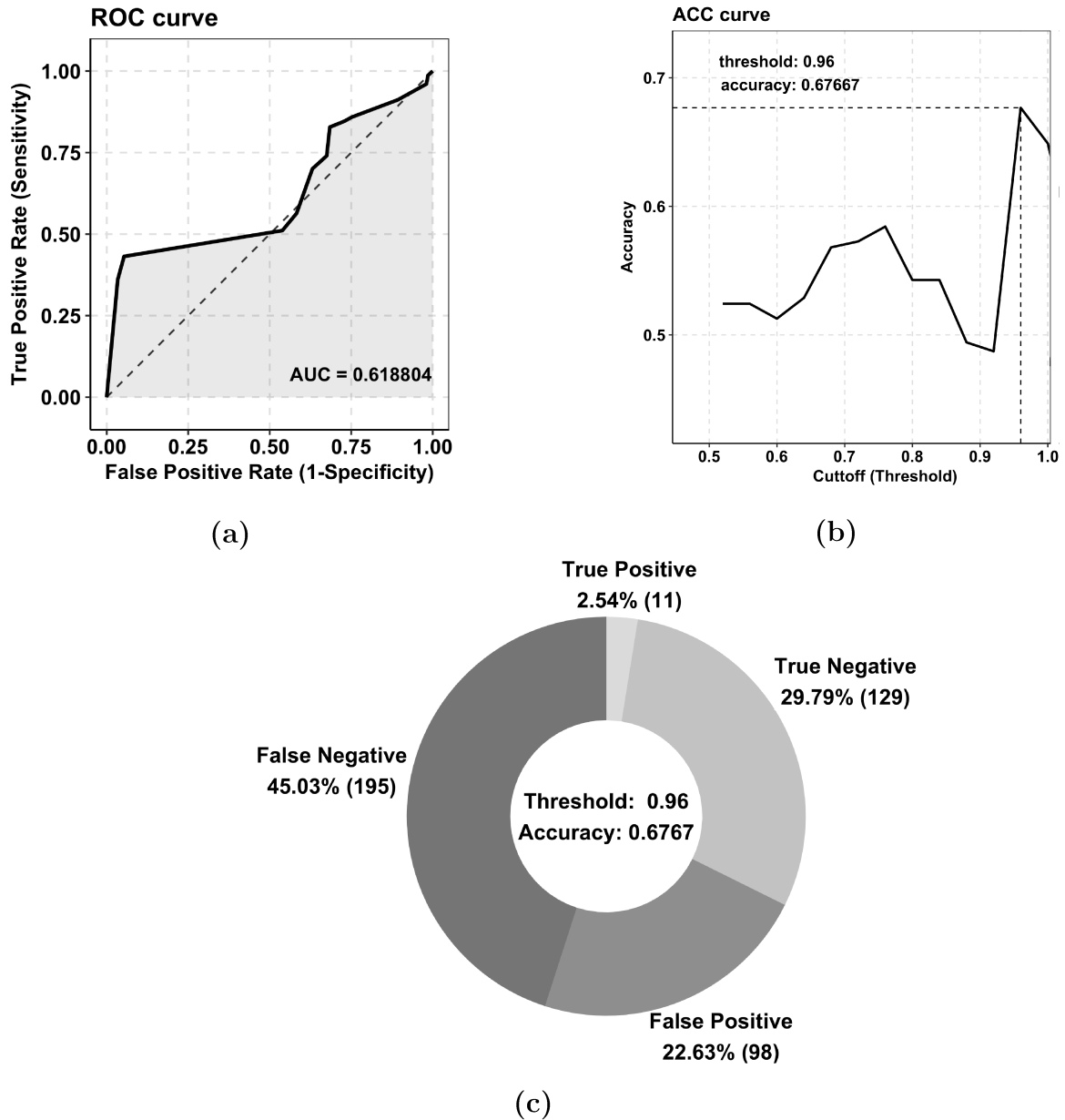


Figura 38 – Desempenho do modelo preditivo aplicado sobre a PPI *Streptococcus pneumoniae* Taiwan19F14, sendo que (a) apresenta a curva ROC com AUC = 0,618804, (b) exibe a curva de acurácia em que destaca o valor de corte threshold = 0,96, e (c) exibe o percentual de verdadeiro positivos (TP), verdadeiro negativos (TN), falso positivos (FP) e falso negativos (FN) para o valor de threshold = 0,96

A classificação dos nós da PPI *Escherichia coli* ED1a pelo classificador Bagging resultou em 229 (20,39%) nós corretamente classificados, dos quais 10 (0,89%) foram verdadeiro positivos e 219 (19,50%) foram verdadeiro negativos, contra 894 (79,61%) nós incorretamente classificados, dos quais 202 (17,99%) nós foram falso positivos e 692 (61,62%) nós falso negativos. A acurácia dessa predição foi de 79,61% com limiar de 96,00%. O valor da AUC foi de 0,606802. A Figura 39 apresenta a curva ROC assim como a distribuição das quantidades previstas em termos de TP, TN, FP, FN.

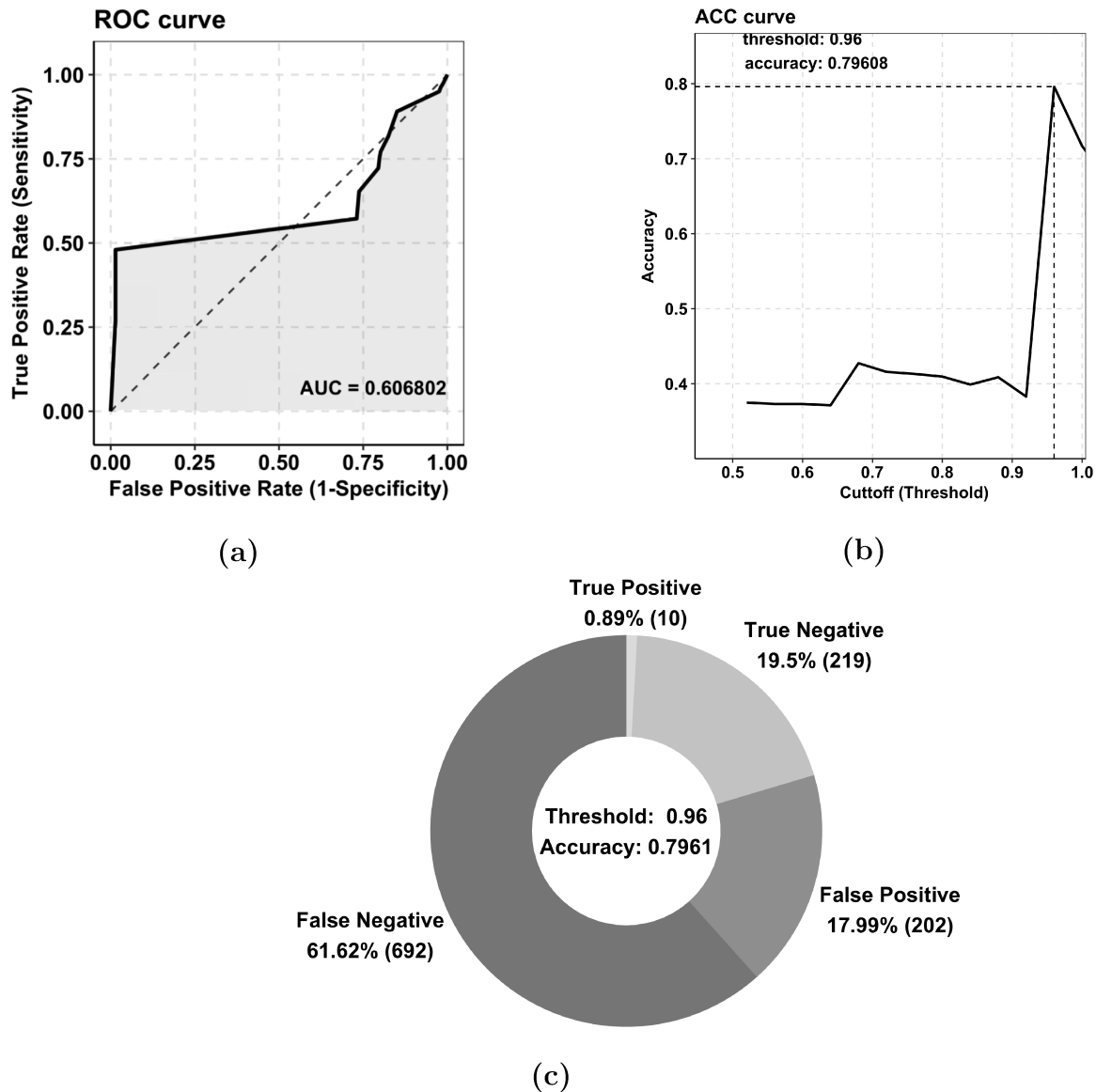


Figura 39 – Desempenho do modelo preditivo aplicado sobre a PPI *Escherichia coli* ED1a, sendo que (a) apresenta a curva ROC com  $AUC = 0,606802$ , (b) exibe a curva de acurácia em que destaca o valor de corte threshold = 0,96, e (c) exibe o percentual de verdadeiro positivos (TP), verdadeiro negativos (TN), falso positivos (FP) e falso negativos (FN) para o valor de threshold = 0,96

Em termos gerais, a acurácia média da predição dos nós de todas as PPIs foi de  $74,38\% \pm 5,84\%$ , com limiar médio de  $96,00\%$ , e AUC médio de  $65,09\% \pm 4,48\%$ , como apresenta a Tabela 12. A quantidade total de nós preditos corretamente foi 1.299 (ou  $24,33\%$  dos nós) dos quais 147 nós (ou  $2,75\%$  do total) foram classificados como verdadeiro positivos (TP) e 1.152 nós (ou  $21,58\%$  do total) como verdadeiro negativos (TN). Por outro lado, 4.040 nós (ou  $75,66\%$  do total) foram incorretamente classificados, onde 1.264 nós (ou  $23,67\%$  do total) foram dados como falso positivos (FP) e 2.776 nós (ou  $51,99\%$  do total) como falso negativos (FN). A Figura 40 apresenta a proporção da distribuição dos nós em termos de TP, TN, FP, FN.

Tabela 12 – Resultado consolidado da predição em termos de AUC, ACC. Threshold = 0,96

PPI	AUC	ACC
<i>Clostridium botulinum A2 Kyoto</i>	0,7366	0,7785
<i>Bacillus anthrax</i>	0,6952	0,8043
<i>Mycobacterium tuberculosis</i>	0,6625	0,8053
<i>Clostridium botulinum F Langeland</i>	0,6391	0,7208
<i>Clostridium perfringens</i>	0,6280	0,6995
<i>Clostridium tetani E88</i>	0,6199	0,6696
<i>Streptococcus pneumoniae Taiwan19F14</i>	0,6188	0,6767
<i>Escherichia coli ED1a</i>	0,6068	0,7961
Média	0,6509	0,7438
Desvio-Padrão	0,0448	0,0584

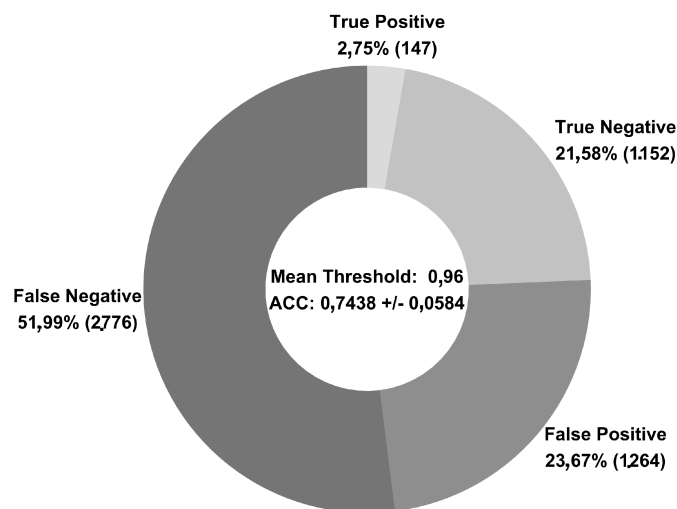


Figura 40 – Resultado consolidado proporcional da predição em termos de verdadeiro positivos (TP), verdadeiro negativos (TN), falso positivos (FP) e falso negativos (FN)

## 4.2 Comparativo Real vs Estimado

Tomando os 2,75% dos nós classificados pelo algoritmo Bagging como verdadeiro positivos (TP) para **strong** e comparando-os com os resultados obtidos pelo algoritmo exato, tem-se a Tabela 13. Nesta tabela, as colunas **Strong Real** e **Weak Real** agrupam as quantidades de nós rotulados como **weak** e **strong**, respectivamente, pelo algoritmo exato para cada rede PPI. A coluna **Strong Predição** agrupa os nós rotulados como **strong** pelo algoritmo Bagging. A coluna **Taxa Acerto** refere-se ao percentual de nós preditos coincidentes com os nós reais. Observando a tabela, a rede PPI *Bacillus anthrax* teve 11 nós preditos como **strong**, sendo que apenas 7 deles coincidiram com o algoritmo exato, ou seja, são realmente **strong**, resultando em  $7/11 = 63,64\%$  de acerto. Para a rede PPI *Escherichia coli ED1a* a taxa de acerto nessa comparação foi integral, ou seja, todos os nós preditos como **strong** coincidiram com os resultados do algoritmo exato, resultando na taxa de acerto de 100%. O pior resultado do comparativo foi obtido pela rede PPI *Clostridium botulinum F Langeland*, em que 23 nós foram preditos como **strong** sendo que apenas 12 deles são realmente **strong**, resultando em  $12/23 = 52,17\%$  de acerto na comparação. Em média, a taxa de acerto do classificador foi de  $77,16\% \pm 20,23\%$  (alternativamente, a taxa de erro médio foi de  $22,84\% \pm 20,23\%$ ).

Tabela 13 – Comparativo Real vs Predição para os 2,75% dos nós classificados como verdadeiros positivos (TP)

PPI	Weak Real	Strong Real	Strong Predição	Taxa Acerto
<i>Escherichia coli ED1a</i>	0	10	10	100,00
<i>Clostridium tetani E88</i>	1	27	28	96,43
<i>Clostridium perfringens</i>	1	24	25	96,00
<i>Clostridium botulinum A2 Kyoto</i>	2	17	19	89,47
<i>Mycobacterium tuberculosis</i>	7	13	20	65,00
<i>Bacillus anthrax</i>	4	7	11	63,64
<i>Streptococcus pneumoniae Taiwan19F14</i>	5	6	11	54,54
<i>Clostridium botulinum F Langeland</i>	11	12	23	52,17
Média				77,16
Desvio-Padrão				20,23

A taxa de rendimento média apontada pela Tabela 13 sugere que o resultado preditivo gerado pelo processo de aprendizado de máquina em estudo nesta pesquisa se aproxima do resultado obtido pelo algoritmo exato. Porém, a taxa de erro médio de  $22,84\% \pm 20,23\%$  ainda é grande a ponto de inviabilizar sua aplicação na identificação de *bridging nodes*, uma vez que a exatidão na classificação é fator de interesse.

---

## Conclusões

### 5.1 Síntese dos Resultados

As redes de interação proteína-proteína, com frequência, comportam números significativos de nós (proteínas) assim como de arestas (interações) entre eles, sendo comum chegarem à ordem dos milhares. Para identificar os nós promissores em redes PPI de grande porte, a adoção da técnica de *bridging centrality* torna-se um problema computacional a ser vencido. Nesse sentido, uma alternativa atraente foi utilizar algoritmos de aprendizado de máquina aplicados às características estruturais dessas redes, com objetivo de predizer quais dos nós estariam mais propensos a serem classificados como *bridging nodes*.

Seguindo essa linha de investigação, este trabalho de pesquisa analisou a aplicação do cálculo de *bridging centrality* sobre redes PPI tomadas como objeto de pesquisa. Um *plugin* para o *software* Gephi foi desenvolvido, cuja função foi calcular o valor exato da *bridging centrality* dos nós das PPIs.

Neste trabalho, as redes PPIs selecionadas para estudo foram submetidas ao processo de aprendizado de máquina de 15 algoritmos disponíveis no *software* WEKA. Foram utilizadas cinco métricas como características estruturais da rede, onde os algoritmos de cálculo de seus valores possuem tempos computacionais menores que  $O(nm + n^2 \log n)$ . Os resultados obtidos nesse processo de aprendizado identificou como modelo preditivo a PPI *Corynebacterium diphtheriae* NC 002935, e o algoritmo Bagging como classificador. Em média, o valor AUC obtido nessa fase foi de  $0.9484 \pm 0.0208$ . Foi observado também que o algoritmo RandomForest é um algoritmo muito promissor, além do Bagging, uma vez que também apresentou elevados valores AUC. Experimentos adicionais foram realizados usando RandomForest como classificador e os resultados obtidos encontram-se no Apêndice D.

Na etapa de classificação, as PPIs em estudo foram submetidas ao algoritmo Bagging tendo como modelo preditivo a PPI *Corynebacterium diphtheriae* NC 002935. Como resultado geral da classificação, 24,33% de todos os nós foram corretamente classifi-

cados, contra 75,66% de nós incorretamente classificados. O valor da acurácia foi de 74,38%  $\pm$  5.84% e o valor de corte limiar (threshold) correspondente foi de 96%. Separou-se, então, os nós classificados corretamente como verdadeiro positivos (TP) para a classe **strong** pelo algoritmo ML. Estes nós foram comparados com os valores atribuídos pelo algoritmo exato. O resultado obtido dessa comparação foi 77,16%  $\pm$  20,23% de eficiência, ou seja, uma taxa de erro média de 22.84%  $\pm$  20.23%, sugerindo que o resultado preditivo gerado pelo processo de aprendizado de máquina em estudo nesta pesquisa se aproxima do resultado obtido pelo algoritmo exato. No entanto, a considerável taxa de erro médio ainda é grande a ponto inviabilizar sua aplicação na identificação de *bridging nodes*, pois a exatidão na classificação dos nós é fator de interesse para a identificação de proteínas visando a produção de fármacos. Dessa forma, nossos resultados corroboram o conhecimento da literatura sobre uso de ML em redes complexas, ou seja, algoritmos de ML aplicados a medidas de centralidade complexas como a *bridging centrality* não são eficazes.

## 5.2 Dificuldades Encontradas

Durante a execução deste projeto de pesquisa, houve relativa dificuldade em se encontrar fontes bibliográficas confiáveis contendo análise de complexidade dos algoritmos para o cálculo das métricas em redes complexas, como *eigenvector centrality*, *hub*, *authority*, *page rank*. Tendo em vista que estas métricas estão relacionadas a operações matriciais, decidiu-se excluí-las da pesquisa. Em estudos posteriores, podem ser objeto de novas avaliações.

Outra dificuldade relacionada a pesquisa bibliográfica, refere-se à análise de complexidade dos algoritmos de aprendizado de máquina. Pouco se encontrou a respeito. Uma investigação mais aprofundada também pode ser sugestão a ser considerada em novos trabalhos.

Uma dificuldade, de ordem prática, foi a seleção dos atributos dos algoritmos ML estudados. Para contorná-la, foi necessário executar diversos testes combinatórios para identificar os parâmetros que geram os melhores resultados para cada algoritmo ML. O Apêndice A apresenta os resultados obtidos nesses testes. O Apêndice B apresenta o *script* gerador dos modelos preditivos, e o Apêndice C apresenta o *script* classificador desenvolvido nessa pesquisa.

## 5.3 Principais Contribuições

Uma contribuição deste trabalho de pesquisa foi a implementação e disponibilização do *plugin*, para o *software* GEPHI, versão 0.9.1 ou superior, para o cálculo da *bridging*



*centrality*, e pode ser obtido pelo site [www.sourceforge.net](http://www.sourceforge.net), sob o nome de BridgingCentralityPlugin<sup>1</sup>.

Outra contribuição relevante é a disponibilização das ordens de complexidade, em termos de tempos de execução, dos algoritmos de cálculo das métricas estudadas, organizadas na Tabela 2 e na Tabela 6, dada a dificuldade encontrada como disposto acima.

Este trabalho de pesquisa também gera contribuição ao apresentar estudo comparativo entre os resultados de um algoritmo exato com os resultados produzidos por algoritmos de aprendizado de máquina, com aplicações em bioinformática.

## 5.4 Trabalhos Futuros

Como sugestões para trabalhos futuros, lista-se:

1. implementar a métrica *bridging centrality* para outros *softwares* de análise de redes complexas, tais como *Pajek*, *Cytoscape*, *SocNetV*;
2. repetir o experimento, ampliando o conjunto de métricas, e sem levar em consideração a complexidade de seus algoritmos geradores. O objetivo, nesse caso, é identificar a combinação de métricas que gere a maior aproximação possível dos resultados produzidos pelo algoritmo exato;

---

<sup>1</sup> Endereço do projeto: <https://sourceforge.net/projects/bridgingcentralityplugin/>



---

## Referências

AKHTAR, N. Social network analysis tools. In: IEEE. **Communication Systems and Network Technologies (CSNT), 2014 Fourth International Conference on**. 2014. p. 388–392. Disponível em: <<https://doi.org/10.1109/CSNT.2014.83>>.

ANANTHASUBRAMANIAN, S. et al. Mycobacterium tuberculosis and clostridium difficile interactomes: demonstration of rapid development of computational system for bacterial interactome prediction. **Microbial informatics and experimentation**, BioMed Central, v. 2, n. 1, p. 1, 2012. Disponível em: <<https://doi.org/10.1186/2042-5783-2-4>>.

BRANDES, U. A faster algorithm for betweenness centrality. **The Journal of Mathematical Sociology**, v. 25, n. 2, p. 163–177, 2001. Disponível em: <<https://doi.org/10.1080/0022250X.2001.9990249>>.

BRANDES, U.; ERLEBACH, T. **Network analysis: methodological foundations**. Springer Science & Business Media, 2005. v. 3418. Disponível em: <<https://doi.org/10.1007/b106453>>.

BRAUN, P. Interactome mapping for analysis of complex phenotypes: insights from benchmarking binary interaction assays. **Proteomics**, Wiley Online Library, v. 12, n. 10, p. 1499–1518, 2012. Disponível em: <<https://doi.org/10.1002/pmic.201100598>>.

BREIMAN, L. Bagging predictors. **Machine Learning**, v. 24, n. 2, p. 123–140, 1996. Disponível em: <<https://doi.org/10.1023/A:1018054314350>>.

\_\_\_\_\_. Random forests. **Machine Learning**, v. 45, n. 1, p. 5–32, Oct 2001. ISSN 1573-0565. Disponível em: <<https://doi.org/10.1023/A:1010933404324>>.

BREITLING, R. et al. Rank products: a simple, yet powerful, new method to detect differentially regulated genes in replicated microarray experiments. **FEBS letters**, Wiley Online Library, v. 573, n. 1-3, p. 83–92, 2004. Disponível em: <<https://doi.org/10.1016/j.febslet.2004.07.055>>.

CHAPELA, V. et al. Mathematical foundations: Complex networks and graphs (a review). In: \_\_\_\_\_. **Intentional Risk Management through Complex Networks Analysis**. Cham: Springer International Publishing, 2015. cap. 2, p. 9–36. ISBN 978-3-319-26423-3. Disponível em: <[https://doi.org/10.1007/978-3-319-26423-3\\_2](https://doi.org/10.1007/978-3-319-26423-3_2)>.

- CLEARY, J. G.; TRIGG, L. E. K\*. An instance-based learner using an entropic distance measure. In: **12th International Conference on Machine Learning**. [s.n.], 1995. p. 108–114. Disponível em: <<https://doi.org/10.1016/B978-1-55860-377-6.50022-0>>.
- COHEN, W. W. Fast effective rule induction. In: PRIEDITIS, A.; RUSSELL, S. (Ed.). **Machine Learning Proceedings 1995**. San Francisco (CA): Morgan Kaufmann, 1995. p. 115–123. ISBN 978-1-55860-377-6. Disponível em: <<https://doi.org/10.1016/B978-1-55860-377-6.50023-2>>.
- CONSORTIUM, G. O. et al. The gene ontology (go) database and informatics resource. **Nucleic acids research**, Oxford Univ Press, v. 32, n. suppl 1, p. D258–D261, 2004. Disponível em: <<https://doi.org/10.1093/nar/gkh036>>.
- COPPERSMITH, D.; WINOGRAD, S. Matrix multiplication via arithmetic progressions. In: **Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing**. New York, NY, USA: ACM, 1987. (STOC '87), p. 1–6. ISBN 0-89791-221-7. Disponível em: <<https://doi.org/10.1145/28395.28396>>.
- COSTA, L. d. F. et al. Characterization of complex networks: A survey of measurements. **Advances in Physics**, Taylor & Francis, v. 56, n. 1, p. 167–242, 2007. Disponível em: <<https://doi.org/10.1080/00018730601170527>>.
- DEHMER, M.; BASAK, S. C. **Statistical and machine learning approaches for network analysis**. John Wiley & Sons, 2012. v. 707. Disponível em: <<https://doi.org/10.1002/9781118346990>>.
- EMAMJOMEH, A. et al. Predicting protein-protein interactions between human and hepatitis c virus via an ensemble learning method. **Mol. BioSyst.**, The Royal Society of Chemistry, v. 10, p. 3147–3154, 2014. Disponível em: <<http://dx.doi.org/10.1039/C4MB00410H>>.
- ERCIYES, K. **Complex Networks: An Algorithmic Perspective**. 1. ed. CRC Press, 2014. ISBN 978-1-4665-7167-9. Disponível em: <<https://doi.org/10.13140/2.1.1608.1281>>.
- FAWCETT, T. **ROC graphs: Notes and practical considerations for data mining researchers**. Palo Alto, CA, USA., 2003. Disponível em: <<http://binf.gmu.edu/mmasso/ROC101.pdf>>.
- \_\_\_\_\_. An introduction to roc analysis. **Pattern recognition letters**, Elsevier, v. 27, n. 8, p. 861–874, 2006. Disponível em: <<https://doi.org/10.1016/j.patrec.2005.10.010>>.
- FERNÁNDEZ-DELGADO, M. et al. Do we need hundreds of classifiers to solve real world classification problems? **Journal of Machine Learning Research**, v. 15, p. 3133–3181, 2014. Disponível em: <<http://jmlr.org/papers/v15/delgado14a.html>>.
- FRANK, E.; WITTEN, I. H. Generating accurate rule sets without global optimization. In: SHAVLIK, J. (Ed.). **Fifteenth International Conference on Machine Learning**. Morgan Kaufmann, 1998. p. 144–151. Disponível em: <<https://hdl.handle.net/10289/1047>>.
- FRIEDMAN, J.; HASTIE, T.; TIBSHIRANI, R. **Additive Logistic Regression: a Statistical View of Boosting**. Stanford University, 1998. Disponível em: <<https://doi.org/10.1214/aos/1016218223>>.

GRANDO, F.; LAMB, L. C. Estimating complex networks centrality via neural networks and machine learning. In: **2015 International Joint Conference on Neural Networks (IJCNN)**. [s.n.], 2015. p. 1–8. Disponível em: <<https://doi.org/10.1109/IJCNN.2015.7280334>>.

\_\_\_\_\_. On approximating networks centrality measures via neural learning algorithms. In: IEEE. **Neural Networks (IJCNN), 2016 International Joint Conference on**. 2016. p. 551–557. Disponível em: <<https://doi.org/10.1109/IJCNN.2016.7727248>>.

HANSON, A. D. et al. "unknown" proteins and "orphan" enzymes: the missing half of the engineering parts list—and how to find it. **Biochemical Journal**, Portland Press Limited, v. 425, n. 1, p. 1–11, 2010. Disponível em: <<https://doi.org/10.1042/BJ20091328>>.

HO, T. K. The random subspace method for constructing decision forests. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 20, n. 8, p. 832–844, 1998. ISSN 0162-8828. Disponível em: <<https://doi.org/10.1109/34.709601>>.

HOLTE, R. Very simple classification rules perform well on most commonly used datasets. **Machine Learning**, v. 11, p. 63–91, 1993. Disponível em: <<https://doi.org/10.1023/A:1022631118932>>.

HWANG, W. et al. Bridging centrality: identifying bridging nodes in scale-free networks. In: **Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining**. [s.n.], 2006. p. 20–23. Disponível em: <<https://www.cse.buffalo.edu/tech-reports/2006-05.pdf>>.

\_\_\_\_\_. Bridging centrality: Graph mining from element level to group level. In: **Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: ACM, 2008. (KDD '08), p. 336–344. ISBN 978-1-60558-193-4. Disponível em: <<https://doi.org/10.1145/1401890.1401934>>.

JOHN, G. H.; LANGLEY, P. Estimating continuous distributions in bayesian classifiers. In: **Eleventh Conference on Uncertainty in Artificial Intelligence**. San Mateo: Morgan Kaufmann, 1995. p. 338–345. Disponível em: <<https://arxiv.org/abs/1302.4964v1>>.

KOSCHÜTZKI, D. Network centralities. In: \_\_\_\_\_. **Analysis of Biological Networks**. John Wiley & Sons, Inc., 2007. p. 65–84. ISBN 9780470253489. Disponível em: <<https://doi.org/10.1002/9780470253489.ch4>>.

LANDWEHR, N.; HALL, M.; FRANK, E. Logistic model trees. **Machine Learning**, v. 95, n. 1-2, p. 161–205, 2005. Disponível em: <<https://doi.org/10.1007/s10994-005-0466-3>>.

LARSEN, P.; HAMADA, Y.; GILBERT, J. Modeling microbial communities: Current, developing, and future technologies for predicting microbial community interaction. **Journal of biotechnology**, Elsevier, v. 160, n. 1, p. 17–24, 2012. Disponível em: <<https://doi.org/10.1016/j.jbiotec.2012.03.009>>.

LIU, Z.-P. et al. Inferring a protein interaction map of mycobacterium tuberculosis based on sequences and interologs. **BMC bioinformatics**, BioMed Central Ltd, v. 13, n. Suppl 7, p. S6, 2012. Disponível em: <<https://doi.org/10.1186/1471-2105-13-S7-S6>>.

MERING, C. V. et al. String 7—recent developments in the integration and prediction of protein interactions. **Nucleic acids research**, Oxford Univ Press, v. 35, n. suppl 1, p. D358–D362, 2007. Disponível em: <<https://doi.org/10.1093/nar/gkl825>>.

OLIVEIRA, G.; SANTOS, A. Using the gene ontology tool to produce de novo protein-protein interaction networks with is\_a relationship. **Genetics and molecular research: GMR**, v. 15, n. 4, 2016. Disponível em: <<https://doi.org/10.4238/gmr15049273>>.

OPSAHL, T.; AGNEESSENS, F.; SKVORETZ, J. Node centrality in weighted networks: Generalizing degree and shortest paths. **Social Networks**, v. 32, n. 3, p. 245 – 251, 2010. ISSN 0378-8733. Disponível em: <<https://doi.org/10.1016/j.socnet.2010.03.006>>.

PRATI, R.; BATISTA, G.; MONARD, M. Curvas roc para avaliação de classificadores. **Revista IEEE América Latina**, v. 6, n. 2, p. 215–222, 2008. Disponível em: <[http://conteudo.icmc.usp.br/pessoas/gbatista/files/ieee\\_la2008.pdf](http://conteudo.icmc.usp.br/pessoas/gbatista/files/ieee_la2008.pdf)>.

ROCHAT, Y. Closeness Centrality Extended to Unconnected Graphs: the Harmonic Centrality Index. In: **ASNA**. [s.n.], 2009. Disponível em: <<http://infoscience.epfl.ch/record/200525>>.

SALZBERG, S. L. C4.5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993. **Machine Learning**, v. 16, n. 3, p. 235–240, Sep 1994. ISSN 1573-0565. Disponível em: <<https://doi.org/10.1007/BF00993309>>.

SCHANK, T.; WAGNER, D. Finding, counting and listing all triangles in large graphs, an experimental study. In: SPRINGER. **WEA**. Springer, Berlin, Heidelberg, 2005. p. 606–609. Disponível em: <[https://doi.org/10.1007/11427186\\_54](https://doi.org/10.1007/11427186_54)>.

SCHWARTZ, A. S. et al. Cost-effective strategies for completing the interactome. **Nature methods**, Nature Publishing Group, v. 6, n. 1, p. 55, 2008. Disponível em: <<https://doi.org/10.1038/nmeth.1283>>.

SILVA, T. C.; ZHAO, L. **Machine learning in complex networks**. Springer, 2016. Disponível em: <<https://doi.org/10.1007/978-3-319-17290-3>>.

SNEL, B. et al. String: a web-server to retrieve and display the repeatedly occurring neighbourhood of a gene. **Nucleic acids research**, Oxford University Press, v. 28, n. 18, p. 3442–3444, 2000. Disponível em: <<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC110752/>>.

STELZL, U. et al. A human protein-protein interaction network: a resource for annotating the proteome. **Cell**, Elsevier, v. 122, n. 6, p. 957–968, 2005. Disponível em: <<https://doi.org/10.1016/j.cell.2005.08.029>>.

WANG, C. G. X. L. X. **Fundamentals of complex networks : models, structures and dynamics**. 1. ed. Wiley, 2015. Disponível em: <<https://doi.org/10.1002/9781118718124>>.

ZHANG, A. **Protein interaction networks: computational analysis**. Cambridge University Press, 2009. Disponível em: <<https://doi.org/10.1017/CBO9780511626593>>.

## Apêndices





## Otimização de parâmetros

A Tabela 14 apresenta os tempos, em milisegundos, obtidos na execução dos algoritmos de cálculo das métricas utilizadas nesta pesquisa.

Tabela 14 – Tempo de execução dos algoritmos geradores das métricas, em milisegundos

Rede	$C_D(i)$	$C_D^w(i)$	$C_B(v)$	$BC(v)$	$C_R(v)$	$\Delta$
<i>Bacillus anthrax</i>	13	32	229	12	241	162
<i>Clostridium botulinum A2 Kyoto</i>	4	11	6	9	15	45
<i>Clostridium botulinum F Langeland</i>	6	17	306	7	313	33
<i>Clostridium perfringens</i>	6	11	574	9	583	23
<i>Clostridium tetani E88</i>	4	9	932	10	942	14
<i>Corynebacterium diphtheriae NC 002935</i>	4	7	243	4	247	8
<i>Escherichia coli ED1a</i>	8	31	139	20	159	236
<i>Mycobacterium tuberculosis</i>	3	23	140	9	149	71
<i>Streptococcus pneumoniae Taiwan19F14</i>	4	11	712	15	727	16

$C_D(i)$  : Degree Centrality

$C_D^w(i)$  : Weighted Degree Centrality

$C_B(v)$  : Betweenness Centrality

$BC(v)$  : Bridging coefficient

$C_R(v)$  : Bridging Centrality

$\Delta$  : triangles

A Tabela 15 apresenta a duração, em segundos, da execução dos testes de otimização de parâmetros aplicada aos algoritmos estudados nesta pesquisa.

Tabela 15 – Tempo de execução dos scripts de otimização de parâmetros dos algoritmos ML

Algoritmo	Início	Fim	Duração [s]
Bagging	2017-07-13 15:05:51	2017-07-14 05:51:11	53.120
J48	2017-07-12 18:44:32	2017-07-12 18:58:17	825
JRip	2017-07-12 18:44:31	2017-07-12 19:29:47	2.716
KStar	2017-07-12 18:44:31	2017-07-12 19:05:47	1.276
LMT	2017-07-12 18:44:29	2017-07-12 21:00:26	8.157
OneR	2017-07-12 18:44:27	2017-07-12 18:49:03	276
PART	2017-07-12 18:44:43	2017-07-12 19:28:49	2.646
RandomCommittee	2017-07-14 10:33:43	2017-07-14 10:43:52	609
RandomForest	2017-07-12 18:44:28	2017-07-12 19:25:43	2.475
RandomSubSpace	2017-07-13 14:55:56	2017-07-14 00:04:39	32.923
RandomTree	2017-07-12 18:44:25	2017-07-12 19:57:17	4.372
REPTree	2017-07-13 13:01:29	2017-07-13 14:08:20	4.011

As seções a seguir apresentam os valores testados para cada parâmetro relevante, de cada algoritmo estudado.

## A.1 Bagging

Tabela 16 – Parâmetros testados para o algoritmo Bagging

Parâmetro	Valores	Descrição
-P	100, 90, 80, 70, 60, 50, 40, 30, 20, 10	-P Size of each bag, as a percentage of the training set size. (default 100)
-I	10, 25, 50, 75, 100, 250, 500	Number of iterations. (current value 10)

-W	RandomTree, RandomForest, REPTree, DecisionStump, ZeroR, KStar	Classifiers
----	---	-------------

## A.2 J48

Tabela 17 – Parâmetros testados para o algoritmo J48

Parâmetro	Valores	Descrição
-C	0, 0.05, 0.10, 0.25, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.90, 0.99	
-M	2, 3, 5, 6, 7	
-O	true, false	collapse tree
-A	true, false	Use Laplace smoothing
-N	3 10 30 50 100 200	
-R	true, false	

## A.3 JRip

Tabela 18 – Parâmetros testados para o algoritmo JRip

Parâmetro	Valores	Descrição
-F	3, 10, 30, 50, 100, 200	number of folds
-N	1.0, 1.5, 2.0, 2.5, 3.0, 3.5	Set the minimal weights of instances within a split. (default 2.0)
-O	2, 3, 4, 5, 10, 20	Set the number of runs of optimizations. (Default: 2)

-P	true, false	Whether NOT use pruning (default: use pruning)
-E	true, false	Whether NOT check the error rate $\geq 0.5$ in stopping criteria (default: check)

## A.4 KStar

Tabela 19 – Parâmetros testados para o algoritmo KStar

Parâmetro	Valores	Descrição
-B	5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 100, 200	Manual blend setting (default 20%)
-M	a, d, m, n	Specify the missing value treatment mode (default a) Valid options are: a(verage), d(elete), m(axdiff), n(ormal)

## A.5 LMT

Tabela 20 – Parâmetros testados para o algoritmo LMT

Parâmetro	Valores	Descrição
-I	1, 2, 5, 10, 20, 30, 40, 50, 100, 200	Set fixed number of iterations for LogitBoost (instead of using cross-validation)
-M	5, 10, 15, 20, 30, 40, 50, 100	Set minimum number of instances at which a node can be split (default 15)
-W	0.0, 1.0, 1.5, 2.0, 2.5, 5.0, 5.5, 10.0, 10.5	Set beta for weight trimming for LogitBoost. Set to 0 (default) for no weight trimming.
-A	true, false	The AIC is used to choose the best iteration.

-C	true, false	Use cross-validation for boosting at all nodes (i.e., disable heuristic)
----	-------------	--

## A.6 MultiLayerPerceptron

Tabela 21 – Parâmetros testados para o algoritmo MultiLayerPerceptron

Parâmetro	Valores	Descrição
-L	0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1	Learning Rate
-M	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9	momentum
-N	500, 1.000, 1.500, 3.000, 5.000, 10.000	epochs
-H	"a", "a,3", "a,4", "a,5", "a,6", "a,7", "a,4,3", "a,5,3", "a,6,3", "a,7,3", "a,5,4,3", "a,6,5,4,3", "a,7,6,5,4,3", "a,8,7,6,5,4,3", "a,8,7,8,7,6,5,4,3"	RNA layers

## A.7 OneR

Tabela 22 – Parâmetros testados para o algoritmo OneR

Parâmetro	Valores	Descrição
-B	$\{1, 2, \dots, 100\}$	

## A.8 PART

Tabela 23 – Parâmetros testados para o algoritmo PART

Parâmetro	Valores	Descrição
-M	$\{1, 2, 3, \dots, 100\}$	Set minimum number of objects per leaf (default 2)
-C	$\{0.05, 0.10, \dots, 0.50\}$	

## A.9 RandomCommittee

Tabela 24 – Parâmetros testados para o algoritmo RandomCommittee

Parâmetro	Valores	Descrição
-I	10, 25, 50, 75, 100, 250, 500	Number of iterations. (current value 10)
-W	RandomTree, RandomForest, REPTree	Classifiers

## A.10 RandomForest

Tabela 25 – Parâmetros testados para o algoritmo RandomForest

Parâmetro	Valores	Descrição
-P	100, 75, 50, 25, 10, 5	Size of each bag, as a percentage of the training set size. (default 100)
-I	100, 50, 25, 10	Number of iterations. (current value 100)
-M	1, 2, 3, 4	Set minimum number of instances per leaf. (default 1)

-V	0.0001, 0.001, 0.005, 0.010, 0.050, 0.100, 0.500, 0.750	Set minimum numeric class variance proportion of train variance for split (default: $10^{-3}$ ).
----	--	--

## A.11 RandomSubSpace

Tabela 26 – Parâmetros testados para o algoritmo RandomSubSpace

Parâmetro	Valores	Descrição
-P	0.1, 0.25, 0.5, 0.75, 1	Size of each subspace: < 1: percentage of the number of attributes; >=1: absolute number of attributes
-I	10, 25, 50, 75, 100, 250, 500	Number of iterations. (current value 10)
-W	KStar, RandomTree, RandomForest, REPTree, J48, LMT, OneR, PART	Classifiers

## A.12 RandomTree

Tabela 27 – Parâmetros testados para o algoritmo RandomTree

Parâmetro	Valores	Descrição
-x	10, 30, 50, 100, 200	folds
-K	0, 1, 2, 3, 4, 5, 6, 7	attr_2_invest
-M	1, 2, 3	per_leaf
-V	$10^{-1}$ , $10^{-2}$ , $10^{-3}$ , $10^{-4}$ , $10^{-5}$ , $10^{-6}$	variance
-N	0, 2, 3	backfitting

## A.13 REPTree

Tabela 28 – Parâmetros testados para o algoritmo REPTree

Parâmetro	Valores	Descrição
-M	$\{1, 2, \dots, 7\}$	Set minimum number of instances per leaf (default 2).
-V	$\{2, 3, \dots, 50\}$	Number of folds for reduced error pruning (default 3)
-N	0.001, 0.0001, 0.005, 0.010, 0.050, 0.100, 0.500	Set minimum numeric class variance proportion of train variance for split (default 1e-3).



## Script Gerador de Modelos Preditivos

```

1  #!/bin/bash

3  help () {
    echo "Usage: $0 <algoritmo><nro modelo><nro exec><nro threshold type>
      seed mode>"
5  echo ""
    echo "  algoritmo"
7  echo "      1 : lazy - KStar"
    echo "      3 : meta - Bagging"
9  echo "      6 : meta - LogitBoost"
    echo "      7 : meta - RandomCommittee"
11 echo "      9 : meta - RandomSubSpace"
    echo "     10 : trees - J48"
13 echo "     11 : trees - LMT"
    echo "     12 : trees - RandomForest"
15 echo "     13 : trees - RandomTree"
    echo "     14 : trees - REPTree"
17 echo "     15 : rules - JRip"
    echo "     16 : rules - OneR"
19 echo "     17 : rules - PART"
    echo "     18 : functions - MultilayerPerceptron"
21 echo "     19 : bayes - NaiveBayes"
    echo "  nro modelo"
23 echo "      1 : Bacillus_anthrax"
    echo "      4 : Clostridium_botulinum_A2_Kyoto"
25 echo "      5 : Clostridium_botulinum_F_Langeland"
    echo "      7 : Clostridium_perfringens"
27 echo "      8 : Clostridium_tetani_E88"
    echo "      9 : Corynebacterium_diphtheriae_NC_002935"
29 echo "     10 : Escherichia_coli_ED1a"
    echo "     12 : Mycobacterium_tuberculosis"
31 echo "     13 : Streptococcus_pneumoniae_Taiwan19F14"
    echo "  nro exec"
33 echo "      quantidade de execucoes (default: 1)"

```

```
    echo " nro threshold type"
35  echo "          formato do arquivo de threshold (estatistica)"
    echo "          1 : arff (default)"
37  echo "          2 : csv"
    echo " seed mode"
39  echo "          modo de tratamento aleatorio para os algoritmos"
    echo "          1 : seed = 1 (default)"
41  echo "          2 : seed aleatorio"
    exit 1
43 }

45 if [ $# -le 1 ]
then
47     help
fi
49
case "$1" in
51     1 )
        group="lazy "
53     algorithm="KStar "
        ;;
55     3 )
        group="meta "
57     algorithm="Bagging "
        ;;
59     6 )
        group="meta "
61     algorithm="LogitBoost "
        ;;
63     7 )
        group="meta "
65     algorithm="RandomCommittee "
        ;;
67     9 )
        group="meta "
69     algorithm="RandomSubSpace "
        ;;
71     10 )
        group="trees "
73     algorithm="J48 "
        ;;
75     11 )
        group="trees "
77     algorithm="LMT"
        ;;
79     12 )
        group="trees "
```

---

```

81     algorithm=" RandomForest "
      ;;
83 13 )
      group=" trees "
85     algorithm=" RandomTree "
      ;;
87 14 )
      group=" trees "
89     algorithm=" REPTree "
      ;;
91 15 )
      group=" rules "
93     algorithm=" JRip "
      ;;
95 16 )
      group=" rules "
97     algorithm=" OneR "
      ;;
99 17 )
      group=" rules "
101     algorithm=" PART "
      ;;
103 18 )
      group=" functions "
105     algorithm=" MultilayerPerceptron "
      ;;
107 19 )
      group=" bayes "
109     algorithm=" NaiveBayes "
      ;;
111 *) echo "Opcao invalida "
      help
113     exit 2
esac
115
116 case "$2" in
117 1 )
      ppi_modelo=" Bacillus_anthrax "
119     ;;
      4 )
121     ppi_modelo=" Clostridium_botullinum_A2_Kyoto "
      ;;
123 5 )
      ppi_modelo=" Clostridium_botullinum_F_Langeland "
125     ;;
      7 )
127     ppi_modelo=" Clostridium_perfringens "

```

```
;;
129 8 )
    ppi_modelo="Clostridium_tetani_E88"
131 ;;
    9 )
133 ppi_modelo="Corynebacterium_diphtheriae_NC_002935"
    ;;
135 10 )
    ppi_modelo="Escherichia_coli_ED1a"
137 ;;
    12 )
139 ppi_modelo="Mycobacterium_tuberculosis"
    ;;
141 13 )
    ppi_modelo="Streptococcus_pneumoniae_Taiwan19F14"
143 ;;
    *) echo "Opcao invalida"
145 help
    exit 3
147 esac

149 if [[ "$3" -eq "" ]]; then
    max=1
151 else
    max=$3
153 fi

155 case "$4" in
    2 )
157     threshold_format="csv"
        ;;
159     *) threshold_format="arff"
161 esac

161 case "$5" in
163     2 )
        seed_mode=2
165     ;;
        *) seed_mode=1
167 esac

169 algoritmo="weka.classifiers.${group}.${algorithm}"
    resultado_treinamento="models/${algorithm}/treinamento.txt"
171 logfile="models/${algorithm}/modelo.model.log"
    arquivo_teste="${ppi_modelo}.arff"
173
    cd ${ppi_modelo}
```

```

175 | directory_name="models/${algorithm}"
177 | if [ ! -d $directory_name ]
    | then
179 |     mkdir -p $directory_name
    | fi
181 |
183 | IFS=$'\012'
185 | echo ""
187 | echo "Gerando modelo de aprendizado "
189 | echo "    >          PPI: ${ppi_modelo}"
191 | echo "    > algoritmo: ${algoritmo}"
193 | echo "    > execucoes: ${max}"
195 | echo ""
197 | start_date=$(date +%s)
199 | inicio=$(date '+%Y-%m-%d %H:%M:%S')
201 | echo "    >>>>> Inicio: ${inicio}"
203 | echo "    >>>>> Inicio: ${inicio}" > ${logfile}
205 |
207 | printf -v max_formatado "%3d" ${max}
209 |
211 | for (( i = 1; i <= ${max}; i++ )); do
213 |
215 |     if [[ ${seed_mode} -eq "1" ]]; then
        |         seed=1
        |     else
217 |         seed='python -c'import random; print repr(random.randrange(1,100))''
        |     fi
219 |
221 |     printf -v resultado_treinamento "models/%s/modelo_%03d_treinamento.txt" $
        |         {algorithm} ${i}
223 |     printf -v output_do_modelo "models/%s/modelo_%03d.model" ${algorithm} ${i
        |         }
225 |     printf -v output_thr "models/%s/modelo_%03d_threshold.%s" ${algorithm} ${
        |         i} ${threshold_format} # threshold file
227 |
229 |     classifications=""
231 |
233 |     printf "    >>>>> %3d de %3d - random seed: %3d - %s - %s - %s\012" ${i} $
        |         {max} ${seed} ${output_do_modelo} ${resultado_treinamento} ${output_thr}
235 |
237 |     case "$1" in
239 |         1 )
241 |             java -cp $WEKA_HOME/weka.jar weka.classifiers.lazy.KStar ${
        |                 classifications} -threshold-file ${output_thr} -threshold-label "strong"

```

```

-s ${seed} -t ${arquivo_teste} -d ${output_do_modelo} -B 5 -M a -num-
decimal-places 4 > ${resultado_treinamento}
;;
2 )
217
219 java -cp $WEKA_HOME/weka.jar weka.classifiers.meta.
AttributeSelectedClassifier ${classifications} -threshold-file ${
output_thr} -threshold-label "strong" -s ${seed} -t ${arquivo_teste} -d
${output_do_modelo} -E "weka.attributeSelection.CfsSubsetEval -P 1 -E 1"
-S "weka.attributeSelection.BestFirst -D 1 -N 5" -W weka.classifiers.
trees.J48 -- -Q ${seed} -C 0.25 -M 2 > ${resultado_treinamento}
;;
221 3 )
java -cp $WEKA_HOME/weka.jar weka.classifiers.meta.Bagging ${
classifications} -threshold-file ${output_thr} -threshold-label "strong"
-t ${arquivo_teste} -d ${output_do_modelo} -P 100 -S ${seed} -num-slots
0 -I 25 -num-decimal-places 4 -W weka.classifiers.trees.RandomTree -- -
K 0 -M 1 -V 1e-3 -S ${seed} -depth 0 -N 0 -U -B -num-decimal-places 4 -
batch-size 100 > ${resultado_treinamento}
223 ;;
4 )
225 java -cp "$WEKA_HOME/weka.jar:$WEKA_HOME/libs/*" weka.classifiers.
meta.ClassificationViaRegression ${classifications} -threshold-file ${
output_thr} -threshold-label "strong" -s ${seed} -t ${arquivo_teste} -d
${output_do_modelo} -W weka.classifiers.trees.M5P -- -M 4.0 > ${
resultado_treinamento}
;;
227 5 )
java -cp $WEKA_HOME/weka.jar weka.classifiers.meta.
IterativeClassifierOptimizer ${classifications} -threshold-file ${
output_thr} -threshold-label "strong" -W weka.classifiers.meta.
LogitBoost -t ${arquivo_teste} -d ${output_do_modelo} -L 50 -P 1 -E 1 -I
1 -F 10 -R 1 -metric RMSE -S ${seed} -- -P 100 -L -1.7976931348623157
E308 -H 1.0 -Z 3.0 -O 1 -E 1 -S ${seed} -I 10 -W weka.classifiers.trees.
DecisionStump > ${resultado_treinamento}
229 ;;
6 )
231 java -cp $WEKA_HOME/weka.jar weka.classifiers.meta.LogitBoost ${
classifications} -threshold-file ${output_thr} -threshold-label "strong"
-t ${arquivo_teste} -d ${output_do_modelo} -P 100 -L
-1.7976931348623157E308 -H .5 -Z 1.0 -O 1 -E 1 -S ${seed} -I 10 -num-
decimal-places 4 -W weka.classifiers.trees.RandomTree -- -K 0 -M 1 -V 1e
-3 -S ${seed} -depth 0 -N 0 -U -B -num-decimal-places 4 -batch-size 100
> ${resultado_treinamento}
;;
233 7 )
java -cp $WEKA_HOME/weka.jar weka.classifiers.meta.RandomCommittee ${
classifications} -threshold-file ${output_thr} -threshold-label "strong"

```

```

-S ${seed} -num-slots 0 -I 10 -num-decimal-places 4 -t ${arquivo_teste}
-d ${output_do_modelo} -W weka.classifiers.trees.RandomTree -- -K 0 -M
1 -V 1e-3 -S ${seed} -depth 0 -N 0 -U -B -num-decimal-places 4 -batch-
size 100 > ${resultado_treinamento}
235     ;;
      8 )
237     java -cp $WEKA_HOME/weka.jar weka.classifiers.meta.
RandomizableFilteredClassifier ${classifications} -threshold-file ${
output_thr} -threshold-label "strong" -t ${arquivo_teste} -d ${
output_do_modelo} -S ${seed} -F "weka.filters.unsupervised.attribute.
RandomProjection -N 10 -R 42 -D Sparse1" -W weka.classifiers.lazy.IBk --
-K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.
EuclideanDistance -R first-last\" " > ${resultado_treinamento}
      ;;
239     9 )
        java -cp $WEKA_HOME/weka.jar weka.classifiers.meta.RandomSubSpace ${
classifications} -threshold-file ${output_thr} -threshold-label "strong"
-t ${arquivo_teste} -d ${output_do_modelo} -P 0.1 -S ${seed} -num-slots
0 -I 10 -num-decimal-places 4 -W weka.classifiers.trees.RandomTree -- -
K 0 -M 1 -V 1e-3 -S ${seed} -depth 0 -N 0 -U -B -num-decimal-places 4 -
batch-size 100 > ${resultado_treinamento}
241     ;;
      10 )
243     java -cp $WEKA_HOME/weka.jar weka.classifiers.trees.J48 ${
classifications} -threshold-file ${output_thr} -threshold-label "strong"
-Q ${seed} -s ${seed} -M 2 -O -A -num-decimal-places 4 -t ${
arquivo_teste} -d ${output_do_modelo} > ${resultado_treinamento}
      ;;
245     11 )
        java -cp $WEKA_HOME/weka.jar weka.classifiers.trees.LMT ${
classifications} -threshold-file ${output_thr} -threshold-label "strong"
-s ${seed} -I -1 -M 30 -W 0.0 -A -C -num-decimal-places 4 -t ${
arquivo_teste} -d ${output_do_modelo} > ${resultado_treinamento}
247     ;;
      12 )
249     java -cp $WEKA_HOME/weka.jar weka.classifiers.trees.RandomForest ${
classifications} -threshold-file ${output_thr} -threshold-label "strong"
-P 100 -I 100 -num-slots 0 -K 0 -M 1 -V 0.0001 -S ${seed} -s ${seed} -
num-decimal-places 4 -t ${arquivo_teste} -d ${output_do_modelo} > ${
resultado_treinamento}
      ;;
251     13 )
        java -cp $WEKA_HOME/weka.jar weka.classifiers.trees.RandomTree ${
classifications} -threshold-file ${output_thr} -threshold-label "strong"
-K 0 -M 1 -V 1e-3 -S ${seed} -s ${seed} -depth 0 -N 0 -U -B -num-
decimal-places 3 -batch-size 100 -t ${arquivo_teste} -d ${
output_do_modelo} > ${resultado_treinamento}

```

```

253     ;;
14 )
255     java -cp $WEKA_HOME/weka.jar weka.classifiers.trees.REPTree ${
classifications} -threshold-file ${output_thr} -threshold-label "strong"
-M 1 -V 0.001 -N 7 -S ${seed} -s ${seed} -L -1 -I 0.0 -num-decimal-
places 4 -t ${arquivo_teste} -d ${output_do_modelo} > ${
resultado_treinamento}
    ;;
257 15 )
    java -cp $WEKA_HOME/weka.jar weka.classifiers.rules.JRip ${
classifications} -threshold-file ${output_thr} -threshold-label "strong"
-S ${seed} -s ${seed} -F 3 -N 1.0 -O 2 -P -E -num-decimal-places 4 -t $
{arquivo_teste} -d ${output_do_modelo} > ${resultado_treinamento}
259     ;;
16 )
261     java -cp $WEKA_HOME/weka.jar weka.classifiers.rules.OneR ${
classifications} -threshold-file ${output_thr} -threshold-label "strong"
-s ${seed} -B 1 -num-decimal-places 4 -t ${arquivo_teste} -d ${
output_do_modelo} > ${resultado_treinamento}
    ;;
263 17 )
    java -cp $WEKA_HOME/weka.jar weka.classifiers.rules.PART ${
classifications} -threshold-file ${output_thr} -threshold-label "strong"
-Q ${seed} -s ${seed} -M 3 -C .20 -num-decimal-places 4 -t ${
arquivo_teste} -d ${output_do_modelo} > ${resultado_treinamento}
265     ;;
18 )
267     java -cp $WEKA_HOME/weka.jar weka.classifiers.functions.
MultilayerPerceptron ${classifications} -threshold-file ${output_thr} -
threshold-label "strong" -S ${seed} -s ${seed} -L 0.3 -M 0.2 -N 1000 -V
0 -E 20 -H a -t ${arquivo_teste} -d ${output_do_modelo} > ${
resultado_treinamento}
    ;;
269 19 )
    java -cp $WEKA_HOME/weka.jar weka.classifiers.bayes.NaiveBayes ${
classifications} -threshold-file ${output_thr} -threshold-label "strong"
-s ${seed} -t ${arquivo_teste} -d ${output_do_modelo} > ${
resultado_treinamento}
271     ;;
    esac
273 done

275 # -----
    # Tratando melhor resultado
277 # -----

279 echo "Identificando modelo de treinamento com menor taxa de erro ..."

```



```

281 get_files () {
    files='ls models/${algorithm}/modelo_*_treinamento.txt '
283 }

285 get_files

287 output_resultados="models/${algorithm}/resultados.csv"

289 for f in ${files}
do
291     teste='sed '/Incorrectly Classified Instances/!d' ${f} | head -1 | awk -F
        ' ' '{print $5}' '
        cross='sed '/Incorrectly Classified Instances/!d' ${f} | tail -1 | awk -F
        ' ' '{print $5}' '
293     # obtendo dados de AUC para labels "strong", "weak" e "avg.", da sessao
        de cross validation
        auc_strong='cat ${f} | sed -n '/Weighted Avg./{g;1!p;};h' | sed 's/^[[:
        blank:]]*//;s/[[:blank:]]*$// ' | sed 's/[[:blank:]]\{2,\}/ /g' | cut -d'
        ' -f7 | tail -1'
295     auc_weak='cat ${f} | sed '1N;$!N;/.*\n.*\n.*Weighted Avg.*/P;D' | sed 's
        /^[[:blank:]]*//;s/[[:blank:]]*$// ' | sed 's/[[:blank:]]\{2,\}/ /g' |
        cut -d' ' -f7 | tail -1'
        auc_avg='cat ${f} | sed '/Weighted Avg./!d' | sed 's/Weighted Avg./ /g' |
        sed 's/[[:blank:]]\{2,\}/ /g' | sed 's/^[[:blank:]]*//;s/[[:blank:]]*$
        //' | cut -d' ' -f 7 | tail -1'
297     echo "${f};${teste};${cross};${auc_strong};${auc_weak};${auc_avg}" >> ${
        output_resultados}
done
299
    cat ${output_resultados} | sed '/^$/d' | sort -t ";" -k2 -n | head -1 | cut
        -d';' -f1,2 > "models/${algorithm}/menor_teste.txt"
301 cat ${output_resultados} | sed '/^$/d' | sort -t ";" -k3 -n | head -1 | cut
        -d';' -f1,3 > "models/${algorithm}/menor_cross.txt"
    cat ${output_resultados} | sed '/^$/d' | sort -t ";" -k3 -n | head -1 | cut
        -d';' -f1 > "models/${algorithm}/menor_modelo.txt"
303
    cat ${output_resultados} | sed '/^$/d' | sort -t ";" -k4 -n -r | head -1 |
        cut -d';' -f1,4 > "models/${algorithm}/melhor_auc_strong.txt"
305 cat ${output_resultados} | sed '/^$/d' | sort -t ";" -k5 -n -r | head -1 |
        cut -d';' -f1,5 > "models/${algorithm}/melhor_auc_weak.txt"
    cat ${output_resultados} | sed '/^$/d' | sort -t ";" -k6 -n -r | head -1 |
        cut -d';' -f1,6 > "models/${algorithm}/melhor_auc_avg.txt"
307
# -----
309 # salvando modelo de aprendizado baseado na melhor AUC do label "strong"
# -----

```

```

311 nro='cat models/${algorithm}/melhor_auc_strong.txt | sed '/^$/d' | head -1
    | cut -d";" -f1 | sed 's/\.txt//g' | cut -d"_" -f2 '
cp "models/${algorithm}/modelo_${nro}_threshold.${threshold_format}" "
  models/${algorithm}/modelo_threshold.${threshold_format}"
313 cp "models/${algorithm}/modelo_${nro}.model" "models/${algorithm}/modelo.
    model"
cp "models/${algorithm}/modelo_${nro}_treinamento.txt" "models/${algorithm}
  }/modelo_treinamento.txt"
315
# -----
317 # compacta os arquivos gerados
# -----
319 printf -v max_formatado "%03d" ${max}

321 data_atual=$(date '+%Y-%m-%d-%H-%M-%S')
archive_dir="models/${algorithm}/models_exec_${max_formatado}_dt_${
  data_atual}"
323 if [ ! -d ${archive_dir} ]
then
325     mkdir -p ${archive_dir}
fi
327
mv models/${algorithm}/modelo_*_threshold.${threshold_format} ${archive_dir}
  }
329 mv models/${algorithm}/modelo_*.model ${archive_dir}
mv models/${algorithm}/modelo_*_treinamento.txt ${archive_dir}
331 tar -czf ${archive_dir}.tgz ${archive_dir}/*
rm -rf ${archive_dir}
333
# -----
335 # salvando arquivos para relatorios futuros
# -----
337 report_dir="models/${algorithm}/info_files"
if [ ! -d ${report_dir} ]
339 then
    mkdir -p ${report_dir}
341 fi
mv models/${algorithm}/melhor_*.txt ${report_dir}
343 mv models/${algorithm}/menor_*.txt ${report_dir}
mv models/${algorithm}/resultados.csv ${report_dir}
345
tx_erro='cat ${report_dir}/menor_cross.txt | sed '/^$/d' | head -1 | cut -d
  ',' -f2 '
347 auc='cat ${report_dir}/melhor_auc_strong.txt | sed '/^$/d' | head -1 | cut
  -d',' -f2 '
echo "    >>>>> Modelo: ${ppi_modelo} - Erro Cross Validation: ${tx_erro} %
  - AUC (strong): ${auc}"

```

```
349 |  
    # -----  
351 |  
    end_date=$(date +%s)  
353 | final=$(date '+%Y-%m-%d %H:%M:%S')  
  
355 | echo "    >>>>> Final: ${final} ($(( ($end_date - $start_date) / (1) )) s)"  
    echo "    >>>>> Final: ${final} ($(( ($end_date - $start_date) / (1) )) s)"  
    >> ${logfile}  
357 | echo "    >>>>> Modelo gerado com sucesso"  
  
359 | cd ..
```



## Script Classificador

```

1  #!/bin/bash
   help () {
3   echo "Usage: $0 <algoritmo><nro modelo><nro exec><nro threshold type><
      seed mode>"
   echo " "
5   echo "  algoritmo"
   echo "      1 : lazy - KStar"
7   echo "      3 : meta - Bagging"
   echo "      6 : meta - LogitBoost"
9   echo "      7 : meta - RandomCommittee"
   echo "      9 : meta - RandomSubSpace"
11  echo "     10 : trees - J48"
   echo "     11 : trees - LMT"
13  echo "     12 : trees - RandomForest"
   echo "     13 : trees - RandomTree"
15  echo "     14 : trees - REPTree"
   echo "     15 : rules - JRip"
17  echo "     16 : rules - OneR"
   echo "     17 : rules - PART"
19  echo "     18 : functions - MultilayerPerceptron"
   echo "     19 : bayes - NaiveBayes"
21  echo "  nro modelo"
   echo "      1 : Bacillus_anthrax"
23  echo "      4 : Clostridium_botullinum_A2_Kyoto"
   echo "      5 : Clostridium_botullinum_F_Langeland"
25  echo "      7 : Clostridium_perfringens"
   echo "      8 : Clostridium_tetani_E88"
27  echo "      9 : Corynebacterium_diphtheriae_NC_002935"
   echo "     10 : Escherichia_coli_ED1a"
29  echo "     12 : Mycobacterium_tuberculosis"
   echo "     13 : Streptococcus_pneumoniae_Taiwan19F14"
31  echo "  nro exec"
   echo "      quantidade de execucoes (default: 1)"
33  echo " "

```

```
35  echo " nro threshold type"
36  echo "      formato do arquivo de threshold (estatística)"
37  echo "      1 : arff (default)"
38  echo "      2 : csv"
39  echo " seed mode"
40  echo "      modo de tratamento aleatório para os algoritmos"
41  echo "      1 : seed = 1 (default)"
42  echo "      2 : seed aleatório"
43  exit 1
44 }
45 if [[ $# -le 1 ]]; then
46     help
47 fi
48
49 case "$1" in
50     1 )
51         y="lazy"
52         x="KStar"
53         ;;
54     3 )
55         y="meta"
56         x="Bagging"
57         ;;
58     6 )
59         y="meta"
60         x="LogitBoost"
61         ;;
62     7 )
63         y="meta"
64         x="RandomCommittee"
65         ;;
66     9 )
67         y="meta"
68         x="RandomSubSpace"
69         ;;
70     10 )
71         y="trees"
72         x="J48"
73         ;;
74     11 )
75         y="trees"
76         x="LMT"
77         ;;
78     12 )
79         y="trees"
80         x="RandomForest"
```

---

```

81     ;;
13  )
83     y=" trees "
      x=" RandomTree "
85     ;;
14  )
87     y=" trees "
      x=" REPTree "
89     ;;
15  )
91     y=" rules "
      x=" JRip "
93     ;;
16  )
95     y=" rules "
      x=" OneR "
97     ;;
17  )
99     y=" rules "
      x=" PART "
101    ;;
18  )
103    y=" functions "
      x=" MultilayerPerceptron "
105    ;;
19  )
107    group=" bayes "
      algorithm=" NaiveBayes "
109    ;;
    *) echo "Opcao invalida "
111    help
    exit 2
113 esac

115 case "$2" in
1   )
117     ppi_modelo=" Bacillus_anthrax "
      ;;
119   4 )
      ppi_modelo=" Clostridium_botullinum_A2_Kyoto "
121     ;;
123   5 )
      ppi_modelo=" Clostridium_botullinum_F_Langeland "
      ;;
125   7 )
      ppi_modelo=" Clostridium_perfringens "
127     ;;

```

```

8 )
129 ppi_modelo="Clostridium_tetani_E88"
    ;;
131 9 )
    ppi_modelo="Corynebacterium_diphtheriae_NC_002935"
133 ;;
10 )
135 ppi_modelo="Escherichia_coli_ED1a"
    ;;
137 12 )
    ppi_modelo="Mycobacterium_tuberculosis"
139 ;;
13 )
141 ppi_modelo="Streptococcus_pneumoniae_Taiwan19F14"
    ;;
143 *) echo "Opcao invalida"
    help
145 exit 3
esac

147 if [[ "$3" -eq "" ]]; then
149     max=1
    else
151     max=$3
    fi
153
    case "$4" in
155     2 )
        threshold_format="csv"
157         ;;
        *) threshold_format="arff"
159     esac

161 case "$5" in
    2 )
163     seed_mode=2
        ;;
165     *) seed_mode=1
    esac
167
    algoritmo="weka.classifiers.${y}.${x}"
169 dir_predictions="predictions"

171 IFS=$'\012'

173 get_files () {
    files="ls */*.arff"

```



```

175 }
177 get_files
179 echo ""
180 echo "Processando arquivos usando algoritmo: ${algoritmo}"
181 echo "                Total de execucoes : ${max}"
182 echo ""
183
184 for f in ${files}
185 do
186
187     if [ "${f%/*}" != "${ppi_modelo}" ]; then
188         modelo="../${ppi_modelo}/models/${x}/modelo.model"
189     else
190         modelo="models/${x}/modelo.model"
191     fi
192
193     cd ${f%/*}
194
195     if [ ! -d ${dir_predictions} ]
196     then
197         mkdir -p ${dir_predictions}
198     fi
199
200     printf -v output_log "%s/%s.log" ${dir_predictions} ${x}
201     logfile="${output_log}"
202     echo "" > ${logfile}
203
204     for (( i = 1; i <= ${max}; i++ )); do
205
206         if [[ ${seed_mode} -eq "1" ]]; then
207             seed=1
208         else
209             seed='python -c'import random; print repr(random.randrange(1,100))'
210         fi
211
212         printf -v output_csv "%s/%s_%03d_prediction.csv" ${dir_predictions} ${x}
213         printf -v output_txt "%s/%s_%03d_prediction.txt" ${dir_predictions} ${x}
214         printf -v output_thr "%s/%s_%03d_threshold.%s" ${dir_predictions} ${x}
215         ${i} ${threshold_format}          # threshold file
216
217         printf -v msg ">>>> %3d de %3d - random seed: %3d - %s - %s - %s - %s - %s\n"
218         ${i} ${max} ${seed} ${output_csv} ${output_txt} ${output_log} ${
219         output_thr} ${f#*/}

```

```

217     saida_csv="weka.classifiers.evaluation.output.prediction.CSV -suppress
- file ${output_csv}"
219     saida_txt="weka.classifiers.evaluation.output.prediction.PlainText -
suppress -file ${output_txt}"

221     echo ""
222     echo ${msg}
223     echo ${msg} >> ${logfile}
msg="java -cp $WEKA_HOME/weka.jar ${algoritmo} -s ${seed} -T ${f#*/} -l
${modelo} -classifications ${saida_csv} -threshold-file ${output_thr} -
threshold-label \"strong\" "
225     echo ${msg} >> ${logfile}
start_date=$(date +%s)
227     java -cp $WEKA_HOME/weka.jar ${algoritmo} -s ${seed} -T ${f#*/} -l ${
modelo} -classifications ${saida_csv} -threshold-file ${output_thr} -
threshold-label "strong"
end_date=$(date +%s)

229     msg=">>>> arquivo gerado com sucesso: ${output_csv}"
231     echo ${msg}
232     echo ${msg} >> ${logfile}

233     msg=">>>> Tempo decorrido: $(( ($end_date - $start_date) / (1) )) s"
235     echo ${msg}
236     echo ${msg} >> ${logfile}
237 done

239 # -----
# gerando relatorio de predicoes
241 # -----
242 echo " >>>>> Identificando prediction com menor taxa de erro ..."
243
get_predictions () {
245     predict_files='ls ${dir_predictions}/${x}*_prediction.csv '
}

247 get_predictions

249 output_resultados="${dir_predictions}/${x}_resultado.csv"

251 echo "" > ${output_resultados}

253
for f in ${predict_files}
255 do
    erros='grep -c " , + , " ${f}'
257     acertados='grep -c " , , " ${f}'

```

```

    echo "${f};"${erros};"${acertos}" >> ${output_resultados}
259 done

261 cat ${output_resultados} | sed '/^$/d' | sort -t ";" -k2 -n | head -1 > "
    ${dir_predictions}/${x}_resultado_menor_erro.txt"

263 valor_erros=$(cat ${dir_predictions}/${x}_resultado_menor_erro.txt | cut
    -d';' -f2)
    valor_acertos=$(cat ${dir_predictions}/${x}_resultado_menor_erro.txt |
    cut -d';' -f3)
265 nome_arquivo=$(cat ${dir_predictions}/${x}_resultado_menor_erro.txt | cut
    -d';' -f1)
    printf "    >>>>> Modelo: %s - %s - Acertos: %3d - Erros: %3d \012" ${
        modelo} ${nome_arquivo} ${valor_acertos} ${valor_erros}
267 cp ${nome_arquivo} ${dir_predictions}/${x}.csv
    cp ${nome_arquivo%_*}_threshold.${threshold_format} ${dir_predictions}/${
        x}_threshold.${threshold_format}

269
    ## -----
271 ## compacta os arquivos gerados
    ## -----

273 printf -v max_formatado "%03d" ${max}
    data_atual=$(date '+%Y-%m-%d-%H-%M-%S')
275 archive_dir="${dir_predictions}/${x}_predictions_exec_${max_formatado}
    _dt_${data_atual}"
    if [ ! -d ${archive_dir} ]
277 then
        mkdir -p ${archive_dir}
279 fi

281 mv ${dir_predictions}/${x}*_threshold.${threshold_format} ${archive_dir}
    mv ${dir_predictions}/${x}*_prediction.csv ${archive_dir}
283 mv ${dir_predictions}/${x}.log ${archive_dir}
    mv ${dir_predictions}/${x}_resultado_menor_erro.txt ${archive_dir}
285 mv ${dir_predictions}/${x}_resultado.csv ${archive_dir}

287 tar -czf ${archive_dir}.tgz ${archive_dir}/*
    rm -rf ${archive_dir}
289 # -----
    cd ..
291 done

```



## Experimentos Adicionais

Foram realizados experimentos de predição usando como modelo de aprendizado as PPIs apontadas em cada seção a seguir. O preditor utilizado foi o algoritmo **Random-Forest**. Para cada PPI, o modelo de aprendizado foi obtido pela linha de comando da listagem D.1 e a aplicação do preditor foi realizada pela linha de comando da listagem D.2. Em ambas as listagens, **<REDE\_PPI>** foi substituído pelo nome da PPI do experimento.

```
1 java -cp $WEKA_HOME/weka.jar weka.classifiers.trees.RandomForest \
   -threshold-file RandomForest_modelo_threshold.csv \
3   -threshold-label "strong" -P 100 -I 100 -num-slots 0 -K 0 -M 1 \
   -V 0.0001 -S 44 -s 44 -num-decimal-places 4 \
5   -t <REDE_PPI>.arff -d <REDE_PPI>.model > RandomForest_treinamento.txt
```

Listing D.1 – Linha de comando para geração de modelo

```
1 java -cp $WEKA_HOME/weka.jar weka.classifiers.trees.RandomForest
   -s 1 -T <REDE_PPI>.arff -l <REDE_PPI>.model \
3   -classifications weka.classifiers.evaluation.output.prediction.CSV \
   -suppress -file RandomForest_prediction.csv \
5   -threshold-file RandomForest_prediction_threshold.csv \
   -threshold-label "strong"
```

Listing D.2 – Linha de comando para predição

### D.1 PPI: *Bacillus anthrax*

Os dados da tabela 29 correspondem ao comparativo entre os *bridging nodes* identificados pelo algoritmo exato com os classificados pelo algoritmo classificador **RandomForest**, obtendo taxa de acerto de **62,46%** e desvio-padrão **16,00%**, tendo sido adotada a PPI **Bacillus anthrax** como modelo preditivo.

Tabela 29 – Resultado da comparação Strong Real vs Strong Predito usando a PPI *Bacillus anthrax* como modelo preditivo para o classificador RandomForest

PPI	Real		Predicted	%
	Weak	Strong	Strong	Correct
<i>Corynebacterium diphtheriae</i> NC 002935	1,00	29,00	30,00	96,67
<i>Escherichia coli</i> ED1a	40,00	107,00	147,00	72,79
<i>Clostridium botulinum</i> A2 Kyoto	43,00	80,00	123,00	65,04
<i>Mycobacterium tuberculosis</i>	61,00	83,00	144,00	57,64
<i>Clostridium botulinum</i> F Langeland	74,00	97,00	171,00	56,73
<i>Clostridium tetani</i> E88	69,00	75,00	144,00	52,08
<i>Streptococcus pneumoniae</i> Taiwan19F14	71,00	72,00	143,00	50,35
<i>Clostridium perfringens</i>	96,00	90,00	186,00	48,39
Média				62,46
Desvio-Padrão				16,00

## D.2 PPI: *Clostridium botulinum* A2 Kyoto

Os dados da tabela 30 correspondem ao comparativo entre os *bridging nodes* identificados pelo algoritmo exato com os classificados pelo algoritmo classificador **RandomForest**, obtendo taxa de acerto de **65,80%** e desvio-padrão **9,75%**, tendo sido adotada a PPI ***Clostridium botulinum* A2 Kyoto** como modelo preditivo.

Tabela 30 – Resultado da comparação Strong Real vs Strong Predito usando a PPI *Clostridium botulinum* A2 Kyoto como modelo preditivo para o classificador RandomForest

PPI	Real		Predicted	%
	Weak	Strong	Strong	Correct
<i>Clostridium botulinum</i> F Langeland	27,00	126,00	153,00	82,35
<i>Corynebacterium diphtheriae</i> NC 002935	19,00	63,00	82,00	76,83
<i>Clostridium perfringens</i>	36,00	91,00	127,00	71,65
<i>Clostridium tetani</i> E88	51,00	80,00	131,00	61,07
<i>Bacillus anthrax</i>	67,00	102,00	169,00	60,36
<i>Streptococcus pneumoniae</i> Taiwan19F14	49,00	72,00	121,00	59,50
<i>Mycobacterium tuberculosis</i>	70,00	96,00	166,00	57,83
<i>Escherichia coli</i> ED1a	98,00	129,00	227,00	56,83
Média				65,80
Desvio-Padrão				9,75

## D.3 PPI: *Clostridium botulinum* F Langeland

Os dados da tabela 31 correspondem ao comparativo entre os *bridging nodes* identificados pelo algoritmo exato com os classificados pelo algoritmo classificador **RandomForest**, obtendo taxa de acerto de **68,10%** e desvio-padrão **10,17%**, tendo sido adotada a PPI ***Clostridium botulinum* F Langeland** como modelo preditivo.

Tabela 31 – Resultado da comparação Strong Real vs Strong Predito usando a PPI *Clostridium botulinum* F Langeland como modelo preditivo para o classificador RandomForest

PPI	Real		Predicted	%
	Weak	Strong	Strong	Correct
<i>Clostridium botulinum</i> A2 Kyoto	18,00	100,00	118,00	84,75
<i>Corynebacterium diphtheriae</i> NC 002935	14,00	52,00	66,00	78,79
<i>Clostridium perfringens</i>	31,00	88,00	119,00	73,95
<i>Streptococcus pneumoniae</i> Taiwan19F14	31,00	66,00	97,00	68,04
<i>Bacillus anthrax</i>	53,00	91,00	144,00	63,19
<i>Mycobacterium tuberculosis</i>	50,00	78,00	128,00	60,94
<i>Escherichia coli</i> ED1a	125,00	176,00	301,00	58,47
<i>Clostridium tetani</i> E88	52,00	68,00	120,00	56,67
Média				68,10
Desvio-Padrão				10,17

## D.4 PPI: *Clostridium perfringens*

Os dados da tabela 32 correspondem ao comparativo entre os *bridging nodes* identificados pelo algoritmo exato com os classificados pelo algoritmo classificador **RandomForest**, obtendo taxa de acerto de **61,61%** e desvio-padrão **6,91%**, tendo sido adotada a PPI ***Clostridium perfringens*** como modelo preditivo.

Tabela 32 – Resultado da comparação Strong Real vs Strong Predito usando a PPI *Clostridium perfringens* como modelo preditivo para o classificador RandomForest

PPI	Real		Predicted	%
	Weak	Strong	Strong	Correct
<i>Corynebacterium diphtheriae</i> NC 002935	25,00	61,00	86,00	70,93
<i>Clostridium botulinum</i> A2 Kyoto	44,00	92,00	136,00	67,65
<i>Streptococcus pneumoniae</i> Taiwan19F14	32,00	57,00	89,00	64,04
<i>Clostridium botulinum</i> F Langeland	58,00	95,00	153,00	62,09
<i>Mycobacterium tuberculosis</i>	55,00	90,00	145,00	62,07
<i>Bacillus anthrax</i>	67,00	100,00	167,00	59,88
<i>Clostridium tetani</i> E88	49,00	69,00	118,00	58,47
<i>Escherichia coli</i> ED1a	140,00	128,00	268,00	47,76
Média				61,61
Desvio-Padrão				6,91

## D.5 PPI: *Clostridium tetani* E88

Os dados da tabela 33 correspondem ao comparativo entre os *bridging nodes* identificados pelo algoritmo exato com os classificados pelo algoritmo classificador **RandomForest**, obtendo taxa de acerto de **70,59%** e desvio-padrão **13,06%**, tendo sido adotada a PPI ***Clostridium tetani* E88** como modelo preditivo.

Tabela 33 – Resultado da comparação Strong Real vs Strong Predito usando a PPI *Clostridium tetani* E88 como modelo preditivo para o classificador RandomForest

PPI	Real		Predicted	%
	Weak	Strong	Strong	Correct
<i>Escherichia coli</i> ED1a	1,00	18,00	19,00	94,74
<i>Bacillus anthrax</i>	2,00	9,00	11,00	81,82
<i>Corynebacterium diphtheriae</i> NC 002935	15,00	52,00	67,00	77,61
<i>Streptococcus pneumoniae</i> Taiwan19F14	24,00	50,00	74,00	67,57
<i>Clostridium perfringens</i>	44,00	84,00	128,00	65,62
<i>Clostridium botulinum</i> A2 Kyoto	41,00	64,00	105,00	60,95
<i>Clostridium botulinum</i> F Langeland	44,00	62,00	106,00	58,49
<i>Mycobacterium tuberculosis</i>	8,00	11,00	19,00	57,89
Média				70,59
Desvio-Padrão				13,06



## D.6 PPI: *Corynebacterium diphtheriae* NC 002935

Os dados da tabela 34 correspondem ao comparativo entre os *bridging nodes* identificados pelo algoritmo exato com os classificados pelo algoritmo classificador **RandomForest**, obtendo taxa de acerto de **76,25%** e desvio-padrão **23,52%**, tendo sido adotada a PPI *Corynebacterium diphtheriae* NC 002935 como modelo preditivo.

Tabela 34 – Resultado da comparação Strong Real vs Strong Predito usando a PPI *Corynebacterium diphtheriae* NC 002935 como modelo preditivo para o classificador RandomForest

PPI	Real		Predicted Strong	% Correct
	Weak	Strong		
<i>Clostridium botulinum</i> A2 Kyoto	0,00	2,00	2,00	100,00
<i>Clostridium perfringens</i>	0,00	19,00	19,00	100,00
<i>Escherichia coli</i> ED1a	0,00	10,00	10,00	100,00
<i>Bacillus anthrax</i>	1,00	6,00	7,00	85,71
<i>Clostridium botulinum</i> F Langeland	4,00	10,00	14,00	71,43
<i>Mycobacterium tuberculosis</i>	2,00	3,00	5,00	60,00
<i>Clostridium tetani</i> E88	86,00	86,00	172,00	50,00
<i>Streptococcus pneumoniae</i> Taiwan19F14	4,00	3,00	7,00	42,86
Média				76,25
Desvio-Padrão				23,52

## D.7 PPI: *Escherichia coli* ED1a

Os dados da tabela 35 correspondem ao comparativo entre os *bridging nodes* identificados pelo algoritmo exato com os classificados pelo algoritmo classificador **RandomForest**, obtendo taxa de acerto de **64,76%** e desvio-padrão **12,29%**, tendo sido adotada a PPI *Escherichia coli* ED1a como modelo preditivo.

Tabela 35 – Resultado da comparação Strong Real vs Strong Predito usando a PPI *Escherichia coli* ED1a como modelo preditivo para o classificador RandomForest

PPI	Real		Predicted	%
	Weak	Strong	Strong	Correct
<i>Corynebacterium diphtheriae</i> NC 002935	8,00	51,00	59,00	86,44
<i>Bacillus anthrax</i>	34,00	116,00	150,00	77,33
<i>Clostridium perfringens</i>	32,00	76,00	108,00	70,37
<i>Mycobacterium tuberculosis</i>	55,00	94,00	149,00	63,09
<i>Clostridium botulinum</i> F Langeland	69,00	102,00	171,00	59,65
<i>Clostridium botulinum</i> A2 Kyoto	90,00	111,00	201,00	55,22
<i>Clostridium tetani</i> E88	63,00	72,00	135,00	53,33
<i>Streptococcus pneumoniae</i> Taiwan19F14	72,00	80,00	152,00	52,63
Média				64,76
Desvio-Padrão				12,29

## D.8 PPI: *Mycobacterium tuberculosis*

Os dados da tabela 36 correspondem ao comparativo entre os *bridging nodes* identificados pelo algoritmo exato com os classificados pelo algoritmo classificador **RandomForest**, obtendo taxa de acerto de **57,34%** e desvio-padrão **8,22%**, tendo sido adotada a PPI ***Mycobacterium tuberculosis*** como modelo preditivo.

Tabela 36 – Resultado da comparação Strong Real vs Strong Predito usando a PPI *Mycobacterium tuberculosis* como modelo preditivo para o classificador RandomForest

PPI	Real		Predicted	%
	Weak	Strong	Strong	Correct
<i>Streptococcus pneumoniae</i> Taiwan19F14	31,00	70,0	101,00	69,31
<i>Escherichia coli</i> ED1a	69,00	131,0	200,00	65,50
<i>Bacillus anthrax</i>	64,00	119,0	183,00	65,03
<i>Corynebacterium diphtheriae</i> NC 002935	49,00	65,0	114,00	57,02
<i>Clostridium perfringens</i>	85,00	93,0	178,00	52,25
<i>Clostridium botulinum</i> F Langeland	93,00	99,0	192,00	51,56
<i>Clostridium tetani</i> E88	75,00	77,0	152,00	50,66
<i>Clostridium botulinum</i> A2 Kyoto	100,00	90,0	190,00	47,37
Média				57,34
Desvio-Padrão				8,22

## D.9 PPI: *Streptococcus pneumoniae* Taiwan19F14

Os dados da tabela 37 correspondem ao comparativo entre os *bridging nodes* identificados pelo algoritmo exato com os classificados pelo algoritmo classificador **RandomForest**, obtendo taxa de acerto de **61,10%** e desvio-padrão **16,83%**, tendo sido adotada a PPI ***Streptococcus pneumoniae* Taiwan19F14** como modelo preditivo.

Tabela 37 – Resultado da comparação Strong Real vs Strong Predito usando a PPI *Streptococcus pneumoniae* Taiwan19F14 como modelo preditivo para o classificador RandomForest

PPI	Real		Predicted	%
	Weak	Strong	Strong	Correct
<i>Corynebacterium diphtheriae</i> NC 002935	1,00	12,00	13,00	92,31
<i>Mycobacterium tuberculosis</i>	5,00	17,00	22,00	77,27
<i>Clostridium tetani</i> E88	20,00	31,00	51,00	60,78
<i>Clostridium perfringens</i>	51,00	78,00	129,00	60,47
<i>Bacillus anthrax</i>	81,00	94,00	175,00	53,71
<i>Clostridium botulinum</i> A2 Kyoto	71,00	82,00	153,00	53,59
<i>Clostridium botulinum</i> F Langeland	65,00	75,00	140,00	53,57
<i>Escherichia coli</i> ED1a	107,00	63,00	170,00	37,06
Média				61,10
Desvio-Padrão				16,83