

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Amanda Silva Moreira

**Classificação de Proteínas Expostas na
Superfície com Random Forest**

Uberlândia, Brasil

2022

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Amanda Silva Moreira

**Classificação de Proteínas Expostas na Superfície com
Random Forest**

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Anderson Rodrigues dos Santos

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2022

Amanda Silva Moreira

Classificação de Proteínas Expostas na Superfície com Random Forest

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Ciência da Computação.

Trabalho aprovado. Uberlândia, Brasil, 19 de agosto de 2022:

Anderson Rodrigues dos Santos
Orientador

Alexsandro Santos Soares
UFU

Paulo Henrique Ribeiro Gabriel
UFU

Uberlândia, Brasil
2022

Dedico este trabalho aos meus pais, que sempre me apoiaram e a minha avó (in memoriam) que, em cada dificuldade, esteve sempre rezando por mim.

Agradecimentos

Não poderia começar esses agradecimentos de outra maneira, pois tenho grande gratidão pelos meus pais, não só pela força e apoio que sempre me deram nos momentos difíceis, mas por sempre acreditarem nos meus sonhos e por me ajudarem a conquistá-los. Sem seu apoio, eu não teria conseguido chegar até aqui. Eles foram a minha força, meu modelo e, principalmente, meu porto seguro nessa jornada. Agradeço, também, aos meus amigos, que sempre estiveram ao meu lado, por serem os melhores amigos que alguém pode ter. Vocês tornaram tudo mais leve, pois eu sabia que poderia sempre contar com vocês.

Agradeço, também, a todos os professores dessa instituição de ensino, que contribuíram para a minha formação acadêmica e, de alguma forma, para a realização deste trabalho. Agradeço ao meu orientador, que me guiou no desenvolvimento deste trabalho de Conclusão de Curso, sem o qual não seria possível. Obrigada, Anderson, pela dedicação e tempo despendido em meu auxílio na realização da pesquisa e, principalmente, por, diante das minhas dificuldades, não desistir de me orientar. O mundo precisa de mais professores como você.

*"Sonhos determinam o que você quer.
Ação determina o que você conquista." - Aldo Novak*

Resumo

Uma das principais causas de doenças e mortes no mundo são infecções bacterianas. A título de exemplo, podemos citar a tuberculose, causada pela bactéria *Mycobacterium tuberculosis*, que ainda mata um milhão e meio de pessoas no mundo, por ano, de acordo com dados recentes da Organização Mundial da Saúde. A vacinação é a melhor estratégia no combate à estas infecções. Entretanto, o desenvolvimento de vacinas para patógenos tem obstáculos, como por exemplo, identificar as proteínas alvo. Somente para *Mycobacterium tuberculosis* há mais de quatro mil proteínas candidatas a alvos para construção de uma vacina. Uma alternativa é o uso de informações genômicas na busca de proteínas que são boas candidatas. O objetivo deste estudo é testar algoritmos populares em Aprendizado de Máquina, implementados no software WEKA, para classificar proteínas expostas na superfície, a partir do padrão hidrofóbico presente na sequência de aminoácidos de 40 genomas, causadores de doenças graves em humanos. As florestas aleatórias tiveram 72,83% de acurácia, tendo tão bons resultados quanto os algoritmos Support Vector Machine e Multilayer Perceptron que alcançaram 70 e 65% de acurácia, respectivamente. Os resultados mostram o bom desempenho dos algoritmos usados para classificar as proteínas expostas na superfície, principalmente considerando a dificuldade em identificá-las.

Palavras-chave: Aprendizado de Máquina, Random Forest, proteínas bacterianas expostas, PSE, WEKA, vacinas.

Abstract

One of the leading causes of disease and death worldwide is bacterial infections. For example, we can mention tuberculosis, caused by the bacterium *Mycobacterium tuberculosis*, which still kills one and a half million people worldwide yearly, according to recent data from the World Health Organization. Vaccination is the best strategy to combat these infections. However, developing vaccines for pathogens have obstacles, such as identifying target proteins. For *Mycobacterium tuberculosis* alone, there are more than 4,000 proteins that are candidates for targets for building a vaccine. An alternative is the use of genomic information in the search for proteins that are good candidates. This work aimed to test popular Machine Learning algorithms, implemented in WEKA software, to classify surface exposed proteins, using the hydrophobic pattern in the amino acid sequence of 40 genomes causing severe human diseases. The random forests had a 72.83% accuracy, performing as well as the Support Vector Machine and Multilayer Perceptron algorithms, reaching of 70 and 65% accuracy, respectively. The results show the satisfactory performance of the algorithms used to classify proteins exposed on the surface, especially considering the difficulty in identifying them.

Keywords: Machine Learning, Random Forest, exposed bacterial proteins, PSE, WEKA, vaccines.

Lista de ilustrações

Figura 1 – Corte da parte hidrofóbica da cadeia proteica	15
Figura 2 – Resultados: Classificação PSE	16
Figura 3 – Representação bactéria - Bacilo	17
Figura 4 – Representação da Membrana Plasmática	18
Figura 5 – Representação do local subcelular das proteínas	21
Figura 6 – Entropia X Probabilidade	22
Figura 7 – Representação de um perceptron	23
Figura 8 – Representação de uma Rede MLP com duas camadas ocultas	24
Figura 9 – Representação da separação dos elementos pelo SMO em um espaço de busca multidimensional	25
Figura 10 – Tabela química dos aminoácidos	28
Figura 11 – Índice de propensão de aminoácidos - AAindex	29

Lista de tabelas

Tabela 1 – Quantidade de proteínas total e por local subcelular predito de 10 patógenos humanos (COSTA J.V. ; SANTOS, 2018).	20
Tabela 2 – Índices de propensão de aminoácidos utilizados para gerar descritores de proteínas na busca por classificar proteínas PSE	30
Tabela 3 – Experimentos PSE com RF	32
Tabela 4 – Resultados: Experimentação PSE com MLP	33
Tabela 5 – Resultados: Experimentação PSE com SMO	33

Lista de abreviaturas e siglas

TB	Tuberculose humana
OMS	Organização Mundial da Saúde
TB MDR	Tuberculose multirresistente
TB XDR	Tuberculose com resistência extensiva a drogas
BCG	Bacilo Calmette-Guérin
PSE	Proteína expostas na superfície
HMMs	Hidden Markov Models
RNA	Redes Neurais Artificiais
RF	Random Forest
WEKA	Waikato Environment for Knowledge Analysis
MED	Mature Epitope Density
BLAST	Basic Local Alignment Search Tool
IA	Inteligência Artificial
ML	Machine Learning
VC	Validação Cruzada
SMO	Otimização sequencial mínima
SVM	Máquina de suporte vetorial
MPLM	Perceptron Multicamadas
QP	Programação quadrática

Sumário

1	INTRODUÇÃO	13
1.1	Objetivos	14
1.1.1	Objetivo geral	14
1.1.2	Objetivos específicos	14
1.2	Metodologia	14
1.3	Resultados	16
1.4	Organização do Trabalho	16
2	FUNDAMENTAÇÃO TEÓRICA	17
2.1	Bactérias	17
2.1.1	Membrana Plasmática	18
2.2	Proteínas	18
2.2.1	Classificação das proteínas	19
2.3	Bioinformática	21
2.4	Machine Learning	21
2.4.1	Árvore de Decisão	22
2.4.2	Random Forest - Floresta aleatória	22
2.4.3	MLP	23
2.4.4	SMO	25
3	METODOLOGIA	27
4	RESULTADOS E DISCUSSÕES	32
4.1	Classificação de proteínas PSE com RF	32
4.2	Classificação de proteínas PSE com outros Algoritmos	32
4.2.1	MLP	32
4.2.2	SMO	33
4.3	Análise dos Resultados	34
5	CONSIDERAÇÕES FINAIS	35
	REFERÊNCIAS	36
	ANEXOS	39
	ANEXO A – CÓDIGO FONTE DO PROGRAMA FEATURES.LISP	40

ANEXO B – FORMATAÇÃO DAS PROTEÍNAS NO FORMATO ARFF	49
ANEXO C – ÍNDICE DE PROPENSÃO DE AMINOÁCIDOS DIS- PONÍVEIS NO REPOSITÓRIO AAINDEX.	51

1 Introdução

Infeções bacterianas ainda são uma das principais causas de doenças no mundo. Por exemplo, a tuberculose humana (TB) que mata aproximadamente um milhão e meio de pessoas por ano no mundo. Recentemente, a Organização Mundial da Saúde (OMS) (OWH, 2021) divulgou que cerca de 1,5 milhão de pessoas morreram de tuberculose em 2020, incluindo 214 mil entre pessoas que vivem com HIV. O risco de uma pessoa com HIV/Aids desenvolver TB ativa é 26 vezes maior em comparação com pessoas sem HIV/Aids (OWH, 2015).

Em relação ao tratamento da TB, no mundo, em 2014, o percentual de sucesso foi de 83% entre os casos novos e recidivas, 52% para os casos multirresistentes (TB MDR) e 28% para os casos com resistência extensiva a drogas (TB XDR). Os resultados para os indivíduos com HIV o percentual de sucesso de tratamento foi de cerca de 75% (SAÚDE, 2019).

Diante da meta de reduzir em 50% a taxa de incidência e mortalidade, a estratégia da OMS tem ações de controle da doença, valorizando a inovação e a incorporação de novas tecnologias, fortalecendo a necessidade do compromisso político, incluindo ações de proteção social aos pacientes e recomendando o acesso universal à saúde (SAÚDE, 2019).

A vacina seria a melhor estratégia no combate à TB, mas não existe uma vacina eficaz contra a doença. Desde 1920, a vacina BCG (Bacilo Calmette-Guérin) é utilizada como medida preventiva complementar no controle da tuberculose, prevenindo especialmente as formas graves da doença, como TB miliar e meníngea na criança. É uma das mais utilizadas no mundo e sua incorporação nos programas de imunização teve impacto na redução da mortalidade infantil por TB em países endêmicos. A BCG não protege indivíduos já infectados e nem evita o adoecimento por reativação endógena, que é a manifestação de TB recorrente causada pela cepa da infecção que não foi totalmente curada, ou reinfeção exógena, que é causada pela infecção de uma cepa diferente do primeiro episódio. Além disso, a partir dos cinco anos de idade, nenhuma pessoa deve ser vacinada com BCG (SAÚDE, 2019).

Uma alternativa é o uso de informações genômicas na busca de proteínas, que são boas candidatas para produção de uma vacina. No entanto, para a identificação dessas proteínas, que são as proteínas alvo, é necessário a obtenção de informações como a sua função biológica e o seu processo de interação com outras moléculas em um sistema biológico (CHOU; SHEN, 2007).

1.1 Objetivos

1.1.1 Objetivo geral

O objetivo deste estudo é testar algoritmos populares em Aprendizado de Máquina (do Inglês Machine Learning ou ML), implementados no software WEKA, para classificar proteínas expostas na superfície (PSE). Neste trabalho há um resumo literário sobre PSE, as primeiras interfaces de contato com as células de um hospedeiro infectado e, por isto, facilmente identificadas pelo sistema imunológico. Este resumo esclarece a importância da classificação de proteínas PSE e como estudos como esse podem ajudar no combate a doenças causadas por infecções bacterianas.

1.1.2 Objetivos específicos

O trabalho fundamenta-se na ideia de utilizar algoritmos de aprendizado de máquina para a classificação das proteínas PSE. Por consequência, os objetivos específicos são:

- Utilizar um conjunto com duas mil e quinhentas proteínas oriundas de bactérias causadoras de doenças em humanos como amostral de proteínas PSE para o treinamento e teste dos algoritmos de classificação;
- Treinar o modelo de classificação das proteínas nas implementações de ML do WEKA;
- Classificar as proteínas entre PSE e não PSE não previamente utilizadas no treinamento do Aprendizado de Máquina;
- Comparar os resultados obtidos via as implementações de algoritmos de ML do WEKA.

1.2 Metodologia

Para a solução do problema proposto já foram usadas Cadeias ocultas de Markov (Hidden Markov Models(HMMs)) (KROGH et al., 2001) e Redes Neurais Artificiais (RNA) (SU et al., 2012) para a predição de proteínas presentes na membrana. Vamos utilizar *Random Forest*(RF) para classificar das proteínas PSE, usando o padrão hidrofóbico presente na sequência de aminoácidos e 40 genomas causadores de doenças graves.

Foi utilizado um programa em Lisp, do laboratório de pesquisa Comp2Bio, chamado *Features* (SANTOS, 2013) que está disponível no anexo A. Este programa gera um histograma de frequência de aminoácidos que, juntamente com dados de propensão de

hidrofobicidade obtidos do banco de dados DBGET (FUJIBUCHI et al., 1998) e características de cada aminoácido, como polaridade e massa, gera um arquivo .arff que foi testado no WEKA (HALL et al., 2009), com *Random Forest*, buscando a probabilidade das proteínas testadas ser do tipo PSE.

A RF usada teve como entrada as proteínas de um genoma, e utilizando uma janela deslizante verificou-se a probabilidade de hidrofobia na cadeia proteica. Em proteínas PSE, a hidrofobia pode estar presente ao longo da cadeia proteica e em qualquer lugar.

Além disso, foi utilizado um pipeline de predição de proteínas alvo, que as classifica numericamente pela estatística de Densidade de Epítomos Maduros ou MED (Mature Epitope Density), que é calculada a cada sequência de aminoácidos, informando os melhores alvos a produção de vacinas e diagnósticos dos patógenos (SANTOS et al., 2013).

Quando à probabilidade do conjunto de aminoácidos ser hidrofóbico, começa a diminuir ao longo da proteína. O modelo treinado corta essa parte hidrofóbica e o cálculo do MED é aplicado, calculando a probabilidade da proteína ser um bom sinalizador para o sistema biológico detectá-la. Esse corte da parte hidrofóbica pode ser visto na figura 1, onde temos um gráfico de probabilidade por quantidade de proteínas.

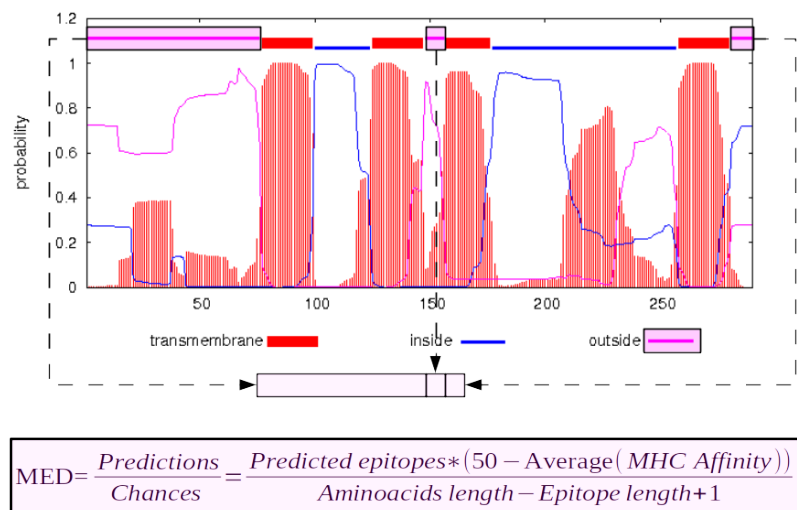


Figura 1 – Corte da parte hidrofóbica da cadeia proteica

Fonte: (SANTOS et al., 2013).

A equação representada na figura 2 divide o número de epítomos lineares previstos de cada sequência de aminoácidos pelo número, por exemplo, de possíveis janelas de sobreposição de peptídeos.

Ao final, para comparar o desempenho da RF, na classificação de proteínas PSE, utilizaremos os algoritmos MLP (multilayer perceptron) e SMO (sequential minimal op-

timization) para o treinamento do mesmo modelo.

1.3 Resultados

Foram realizados os testes conforme proposto com a RF, MLP e SMO e comparamos os resultados.

Experimento	Qtde Árvores	Qtde Features	Tx. acerto RF	C. Ocultas	Neuronios em cada C. Oculta	η (learning Rate)	Epocas	Tx de acerto MLP	Dados	C	Kernel Expoente	Tx de acerto SMO
Experimento 1	50	3	72,86%	1	12	0,3	500	65,67%	normalizado	1	1	63,81%
Experimento 2	50	6	72,48%	2	12	0,3	500	65,33%	padronizado	1	1	65,43%
Experimento 3	50	12	71,62%	1	12	0,3	1000	67,62%	normalizado	5	1	65,19%
Experimento 4	100	3	72,29%	2	12	0,3	1000	65,24%	normalizado	10	1	65,43%
Experimento 5	100	6	72,95%	1	12	0,5	500	64,81%	normalizado	20	1	65,24%
Experimento 6	100	12	73,14%	2	12	0,5	500	63,67%	normalizado	30	1	65,76%
Experimento 7	200	3	72,81%	1	12	0,5	1000	65,67%	normalizado	1	2	68,33%
Experimento 8	200	6	73,19%	2	12	0,5	1000	66,05%	normalizado	30	2	69,76%
Experimento 9	200	12	73,00%	1	24	0,3	500	66,57%	normalizado	1	3	70,62%
Experimento 10	500	12	72,71%	2	24	0,3	500	66,10%	normalizado	1	4	70,24%

Figura 2 – Resultados: Classificação PSE

Fonte: tabela elaborada pela autora (2022).

Analisando os resultados podemos inferir que todos tiveram acurácias próximas. O tempo de execução, em segundos, dos algoritmos RF e MLP se aproxima e, por isso, não foi um ponto usado para comparação, mas o SMO foi o algoritmo menos eficiente nesse quesito.

Com as execuções desses algoritmos, pode-se concluir que a utilização das RF é válida para solução o problema proposto, uma vez que, em média, as RF tiveram uma boa acurácia. É importante ressaltar que 73% de acurácia é considerado aceitável na classificação de proteínas PSE, pois, essas são um meio termo entre as de membrana e secretadas, compartilhando domínios hidrofóbicos e aumentando a dificuldade em identificá-las.

1.4 Organização do Trabalho

Esse trabalho segue a seguinte estrutura. No capítulo 2, é apresentado uma fundamentação teórica, necessária para melhor entendimento quanto ao assunto. No capítulo 3, é especificada a metodologia utilizada. No capítulo 4 são exibidos os experimentos e analisados os resultados encontrados. E, finalmente, no capítulo 5, são feitas conclusões e sugestões de trabalhos futuros.

2 Fundamentação Teórica

Neste capítulo são apresentados os conceitos básicos necessários para compreensão deste estudo, bem como os trabalhos relacionados.

2.1 Bactérias

Bactérias são procariontes, unicelulares, as quais podem viver em colônias, são autótrofos ou heterótrofos, aeróbicas ou anaeróbicas. Têm grande importância para a saúde pública, porque elas podem causar doenças, mas também atuam na produção de alimentos, na estética, na microbiologia e na biotecnologia, na associação com outros seres vivos e na fertilização do solo.

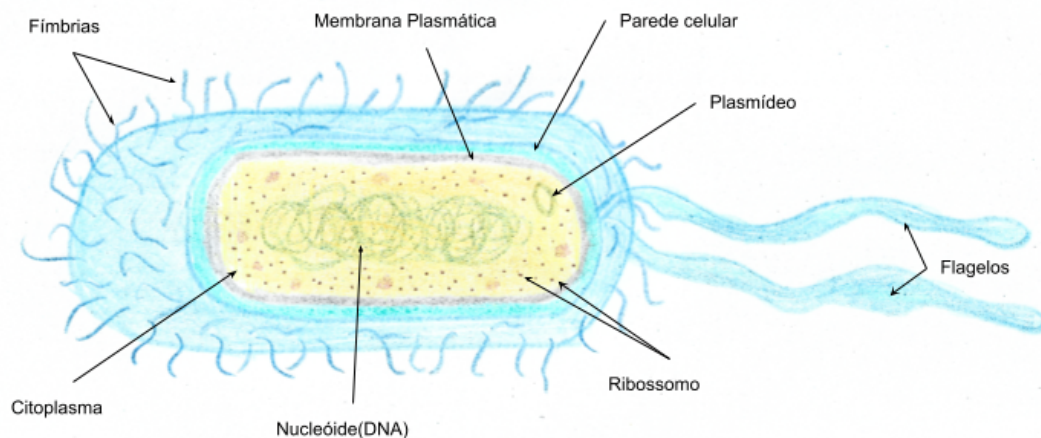


Figura 3 – Representação bactéria - Bacilo

Fonte: elaborado pela autora (2022).

Sua estrutura é formada pelas fimbrias (PILI), que participam da troca de material genético; parede celular, que é formada por peptidoglicanos; flagelos; citoplasma; DNA circular - nucleotídeo; o ribossomo, que produz as proteínas da bactéria; a membrana plasmática e o plasmídeo, que é um fragmento de DNA circular. Algumas bactérias possuem cápsula, que dificulta a defesa do organismo a matar essa célula bacteriana (VERLI, 2014).

São classificadas em heterotróficas e autotróficas. As heterotróficas são parasitas (patogênicas) que causam doenças ou saprofágicas, que são as bactérias que se alimentam do meio. As autotróficas são fotossintetizantes ou quimiosintetizantes, ou seja, são bactérias que produzem, a partir do meio, o próprio alimento.

A reprodução das bactérias é de forma assexuada, ou seja, gerando clones das células. Ou ocorre na forma de troca de material genético, que é a forma sexuada da reprodução.

2.1.1 Membrana Plasmática

A membrana plasmática está presente em todas as células, é lipoproteica, formada de lipídeos e proteínas, é semipermeável, possuindo permeabilidade seletiva. Sua função na célula é revestir, proteger e selecionar.

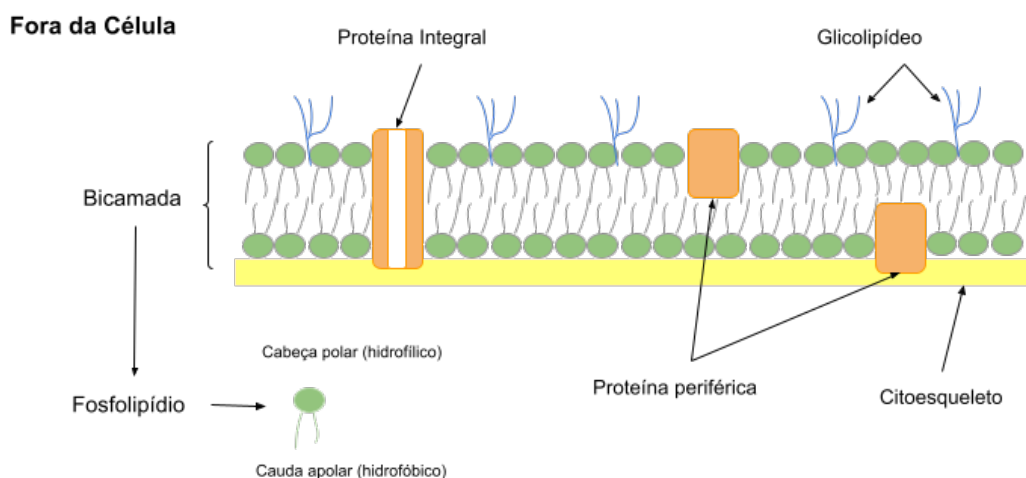


Figura 4 – Representação da Membrana Plasmática

Fonte: elaborado pela autora (2022).

É formada basicamente por fosfolípidios em uma estrutura de bicamada, sendo a cabeça polar hidrofílica e a cauda apolar hidrofóbica. Fazendo com que a parte externa da membrana seja hidrofílica e, tanto para fora da célula quanto para dentro da célula, a parte interna da membrana é hidrofóbica.

Existem proteínas presentes na membrana, sendo elas, integralmente presente, que tem a função de transporte celular, ou parcialmente presente na membrana, proteínas periféricas, que tem função enzimática, ou seja, atuam na velocidade das reações químicas da bactéria (WHITELEGGE, 2006).

2.2 Proteínas

As proteínas são compostos orgânicos, basicamente um polímero formado por monômeros, sendo estes aminoácidos, ou seja, moléculas ligadas por ligação covalente. Existem 20 aminoácidos que formam as proteínas, diferenciadas pela sequência deles, ou seja, de acordo com a sequência da união de aminoácidos uma proteína diferente é for-

mada. As proteínas participam de praticamente todos os aspectos do processo biológico dentro da célula (MULEY; ACHARYA, 2012).

Possuem função estrutural, energética, de defesa (anticorpos), enzimáticas e hormonais. A ligação entre os aminoácidos para formar uma proteína é uma ligação peptídica, realizada por síntese por desidratação, isto é, ocorre entre um carbono e um nitrogênio (VERLI, 2014).

A forma da proteína é diretamente ligada à sua função, se a proteína modifica sua forma, ou estrutura, ela modifica sua função, essa modificação pode ser feita por mutação genética ou por desnaturação. A desnaturação que ocorre quando o meio é alterado de tal forma que também altera estrutura tridimensional da proteína e afeta sua função biológica (MULEY; ACHARYA, 2012).

A estrutura da proteína pode ser primária, isto é, uma fila de aminoácidos, secundária, que por atração química entre os aminoácidos ela se enrola, sendo no formato Alfa hélice ou folha Beta. Ainda existem as estruturas terciárias ou quaternárias que são formadas por domínios similares de grupos de alfa hélice e folhas beta (VERLI, 2014).

2.2.1 Classificação das proteínas

As proteínas produzidas em uma célula podem ser classificadas em quatro tipos, diferenciadas quanto ao seu local na célula e função, são esses: citoplasmáticas, membranares, expostas na superfície e secretadas.

- Citoplasmáticas: ficam no interior da célula, onde são produzidas e onde atuam. São hidrofílicas, ou seja, solúveis à água presente no interior da célula da bactéria (CORDWELL, 2006).
- Membranares: são localizadas na membrana celular. Podem ser chamadas de proteínas transmembrana e são responsáveis pela comunicação da célula bacteriana e o hospedeiro (CORDWELL, 2006).
- Expostas na superfície: apesar de estar localizada também na membrana celular, possuem contato com o exterior da célula e tem contato direto com o hospedeiro infectado (CORDWELL, 2006).
- Secretadas: presentes diretamente no organismo do hospedeiro, são as proteínas que foram secretadas pela bactéria com sua informação genética, podendo agir como indutoras da proliferação celular (CORDWELL, 2006)

-	Organismo	Total de proteínas	Citoplasmáticas	Membranares	Expostas na Superfície	Secretadas
1	Bacillus anthracis str. A0248	5292	3518	1063	502	209
2	Clostridium botulinum A2	3877	2772	606	375	124
3	Enterococcus faecium NRRL B-2354	2639	1875	460	221	83
4	Escherichia coli ED1a	4911	3385	707	401	418
5	Klebsiella pneumoniae	5265	3653	829	375	408
6	Mycobacterium tuberculosis str. Beijing	4110	3145	421	326	218
7	Salmonella enterica str. DT104	4597	3109	473	601	414
8	Staphylococcus aureus 04-02981	2648	1826	480	225	117
9	Streptococcus agalactiae NEM316	2094	1450	339	220	85
10	Vibrio cholerae IEC224	3664	2415	519	395	335

Tabela 1 – Quantidade de proteínas total e por local subcelular predito de 10 patógenos humanos (COSTA J.V. ; SANTOS, 2018).

A quantidade de proteínas presente em cada local subcelular dos organismos de 10 patógenos humanos pode ser visto na tabela 1.

Classificar as proteínas alvo tem alguns obstáculos, como por exemplo, encontrar um possível alvo, devido à grande quantidade deles, exemplificada na tabela 1. As proteínas se dividem de acordo com o local subcelular: no interior da célula tem as proteínas citoplasmáticas, presentes na membrana plasmática tem as proteínas integralmente na membrana e parcialmente expostas na superfície e, as secretadas, são as proteínas fora da célula, conforme representado na figura 5.

Devido ao grande número de proteínas presentes no organismo, conforme representado acima, foi escolhido apenas local subcelular para esta pesquisa, sendo elas as proteínas PSE. Diminuindo assim o escopo de busca pela proteína alvo para a produção de vacinas.

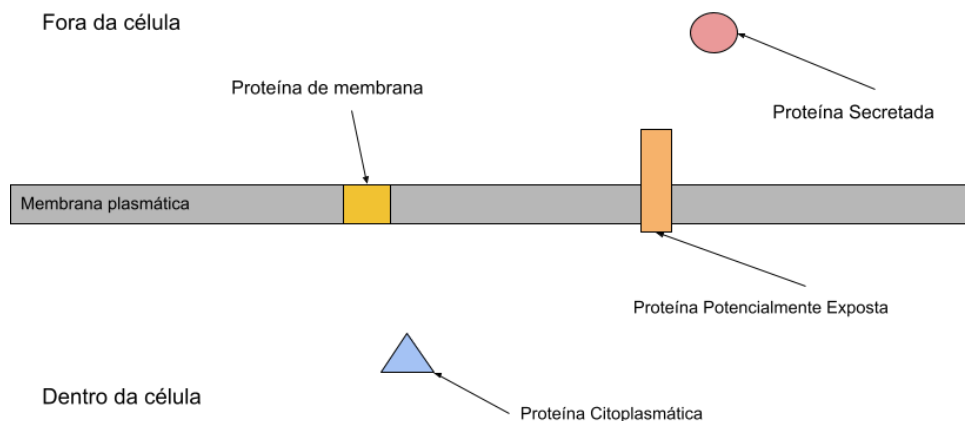


Figura 5 – Representação do local subcelular das proteínas

Fonte: elaborado pela autora (2022).

2.3 Bioinformática

A bioinformática é uma área interdisciplinar que visa a aplicação de técnicas computacionais para auxiliar em estudos biológicos. As ferramentas de bioinformática têm sido utilizadas e produzidas com o intuito de melhorar a compreensão desta área de estudo. Dentre essas, resalta-se a ferramenta Basic Local Alignment Search Tool (BLAST), que permite comparar uma sequência de nucleotídeos ou aminoácidos com outras amostras e identificando as regiões de similaridade entre as elas (<https://blast.ncbi.nlm.nih.gov/Blast.cgi>).

Entre suas aplicações, pode-se destacar a busca pela identificação e classificação de proteínas de patógenos como potenciais alvos para a produção de fármacos e vacinas. Atualmente, tem sido objeto de interesse caracterizando-se como uma importante ferramenta na da área da Saúde graças às descobertas e avanços gerados, especialmente na biomedicina (ARAÚJO et al., 2008).

Duas abordagens principais sustentam esses estudos em bioinformática. Primeiro, o de comparar e agrupar os dados de acordo com semelhanças biologicamente significativas e segundo, a de analisar um tipo de dados para inferir e entender as observações para outro tipo de dados. Portanto, a bioinformática proporciona uma maior profundidade ao estudo biológico, abrangendo a biologia estrutural e genômica. (LUSCOMBE; GREENBAUM; GERSTEIN, 2001).

2.4 Machine Learning

Aprendizado de máquina (*Machine Learning* (ML)) tem como objetivo o desenvolvimento de programas que melhoram o desempenho na realização de tarefas por meio da experiência, adquirindo conhecimento de forma automática, tomando decisões baseado nas experiências acumuladas durante a solução de problemas anteriores (SOUTO et al.,

2003). O objetivo desta seção é introduzir os algoritmos de ML utilizadas nesta pesquisa, com o propósito de facilitar seu entendimento.

2.4.1 Árvore de Decisão

Árvore de decisão é a representação, em forma de árvore, de uma tabela de decisão. Para construir a árvore de decisão, o algoritmo faz o cálculo da entropia dos dados, ou seja, grau de desordem dos dados e o ganho de informação de cada característica, optando pela característica com maior ganho de informação.

$$GI = Entropia_{p_{ai}} - \sum_i (Peso_{filho} * Entropia_{filho}) \quad (2.1)$$

$$Entropia = \sum_i (p_{ai} * \log_2(p_{ai})) \quad (2.2)$$

Assim como a probabilidade, a entropia varia de 0 a 1 e, quanto mais dispersos são os dados, isto é, quanto maior a desordem, maior é a entropia. E quanto menor desordem, menor é a entropia, conforme representado na figura 6.

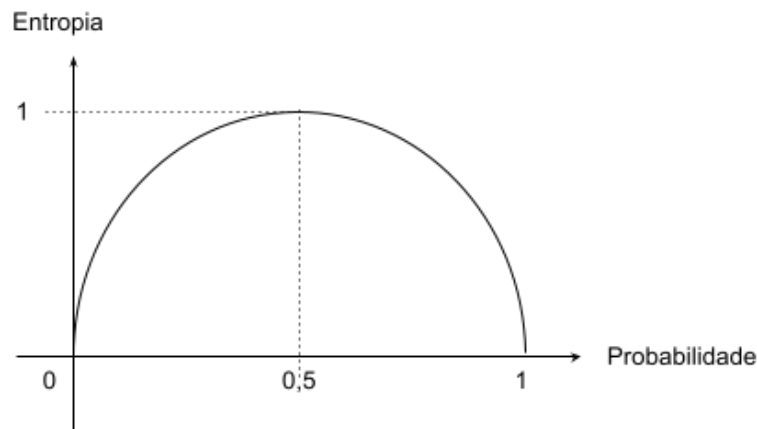


Figura 6 – Entropia X Probabilidade

Fonte: elaborado pela autora (2022).

Seu desempenho pode melhorar com o uso da poda, que remove da árvore os nodos que não têm tanta importância, reduzindo sua complexidade e, assim, reduzindo o *overfitting* e aumentando a precisão.

2.4.2 Random Forest - Floresta aleatória

RF é a união dos preditores de árvores de decisão, sendo que cada árvore tem como entrada um vetor aleatório e com mesma distribuição para todas as outras árvores.

À medida que o número de árvores na RF aumenta, o erro de generalização converge para um limite, esse erro depende da força das árvores de forma individual e da correlação entre elas (BREIMAN, 2001).

Utiliza da classificação ou regressão para o treinamento do modelo, unindo a simplicidade das árvores de decisão com a flexibilidade e aleatoriedade, na seleção dos atributos (*features*), obtendo melhores resultados com ML. O tamanho do conjunto de dados de cada árvore pode ser controlado por um parâmetro ou todo o conjunto de dados pode ser usado para construir cada árvore.

É um modelo vantajoso, pois evita o *Overfitting* (sobre ajuste), tendo assim uma boa precisão. Pode lidar com valores ausentes e ser mais estável. Para evitar o *Overfitting* a RF utiliza o treinamento, validação e teste, construindo modelos mais generalizados, a complexidade depende do tamanho da base de dados.

O erro de generalização para florestas converge à medida que o número de árvores na RF aumenta e depende da força das árvores individuais na floresta e da correlação entre elas, isto é monitorado pelas estimativas internas. Essas estimativas também são usadas para medir a importância da variável, tanto na classificação quanto na regressão (BREIMAN, 2001).

2.4.3 MLP

Inspirando no neurônio biológico estabeleceu-se na Inteligência Artificial um modelo computacional de um neurônio, conforme representado na figura 7, conhecido como perceptron.

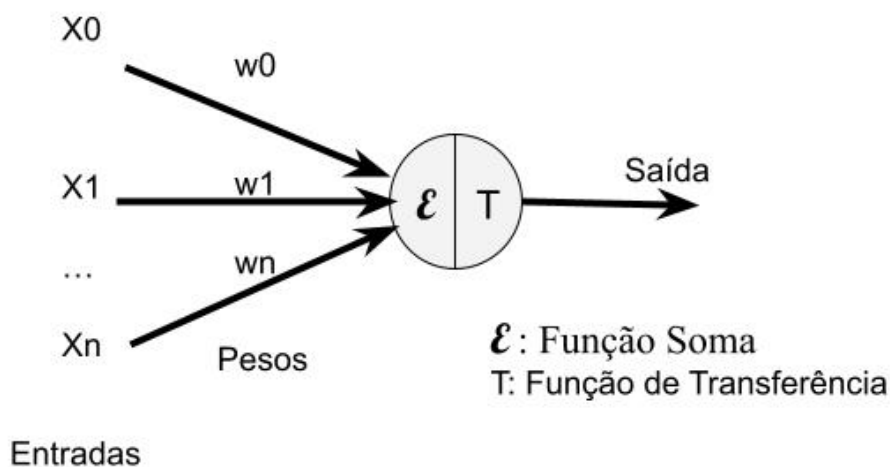


Figura 7 – Representação de um perceptron

Fonte: elaborado pela autora (2022).

$$T = \sum_i (x_i * w_i) \quad (2.3)$$

A rede perceptron multicamadas (MLP) é um algoritmo de aprendizado supervisionado, a regra de aprendizado da rede é a operação de encontrar o valor correto dos pesos aplicados a cada neurônio, esses valores são alterados, em tempo de execução, até que a rede solucione o problema (NORIEGA, 2005).

O MLP usa backpropagation para aprender a classificar instâncias, e esses parâmetros podem ser monitorados e modificados durante o treinamento. A rede possui uma camada de entrada com neurônios como receptores, uma ou mais camadas ocultas de neurônios que computam os dados a cada iteração e, em seguida, a camada de saída que predizem a saída. Tem uma grande variedade de aplicações de classificação e regressão em diversos campos, sendo eles, reconhecimento de padrões, de voz e problemas de classificação (NORVIG, 2013).

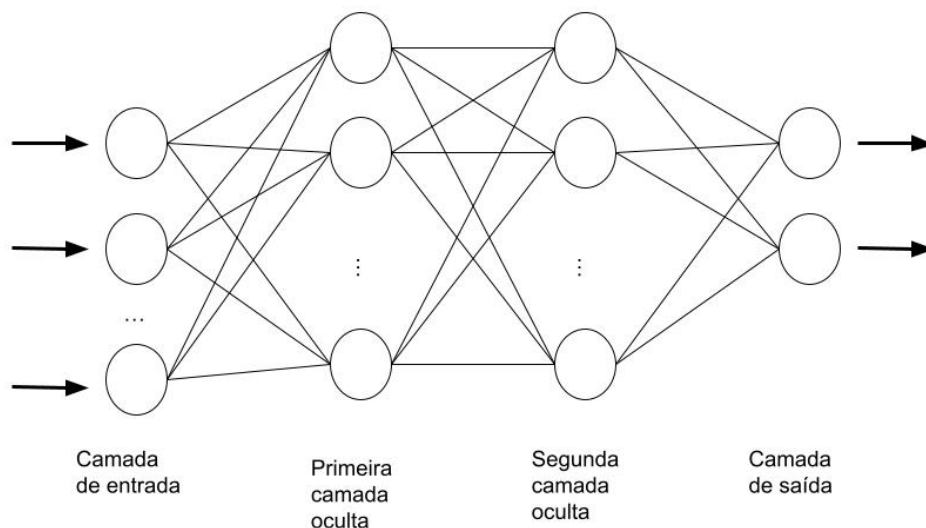


Figura 8 – Representação de uma Rede MLP com duas camadas ocultas

Fonte: elaborado pela autora (2022).

O treinamento da rede MLP tem em dois passos, no primeiro, os dados de entrada passam pela rede e esta calcula sua resposta que é produzida na camada de saída, onde o erro é calculado e no segundo passo, este é propagado a partir da camada de saída até a camada de entrada, e os pesos das conexões das unidades das camadas internas são ajustados. Assim que o erro for reduzido até um nível aceitável a rede pode ser usada para classificar novos dados.

O aprendizado da rede é a partir de conhecimentos empíricos usando uma base de exemplo referente ao problema proposto, sendo capaz de lidar com dados quantitativos, ruidosos e até incompletos. Porém, o tempo de treinamento pode ser um problema

dependendo da arquitetura escolhida para a rede e da quantidade de dados (NORVIG, 2013).

2.4.4 SMO

O SMO é um algoritmo de otimização mínima sequencial que divide um problema grande de otimização de programação quadrática (QP) em uma série de menores problemas de QP que são resolvidos analiticamente, o que evita o uso de uma otimização QP numérica demorada como um *loop* interno, para o treinamento de uma máquina de suporte vetorial (SVM), que é atrativo pela garantia da convergência para um mínimo global da superfície de erro, onde o erro refere-se à diferença entre a resposta desejada e a saída da SVM (PLATT, 1998). Na figura 9 mostra como o SMO separa os elementos em um espaço de busca multidimensional.

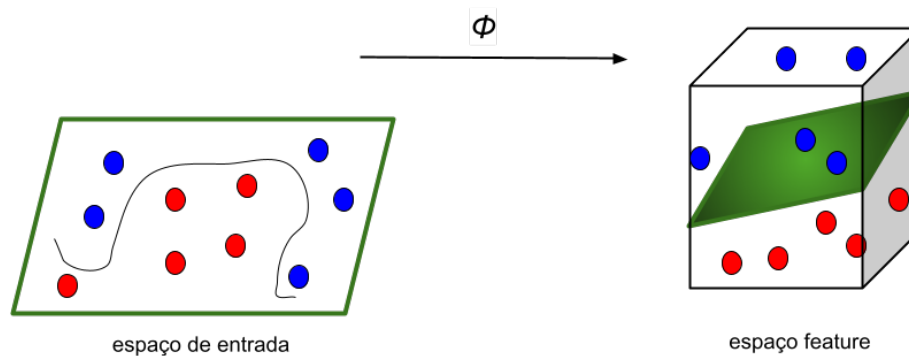


Figura 9 – Representação da separação dos elementos pelo SMO em um espaço de busca multidimensional

Fonte: elaborado pela autora (2022).

SVM realizam um mapeamento não-linear (realizado por um produto interno kernel definido inicialmente) dos dados para um espaço característico no hiperplano para separá-los linearmente em duas classes, baseando-se nos princípios da minimização do risco estrutural, oriundo da teoria do aprendizado estatístico, a qual está baseada no fato de que o erro de aprendizagem é limitado pelo erro de treinamento.

SVM é um modelo vantajoso considerando a memória e quando a margem de separação dos dados é bem definida, também quando o número de dimensões é maior que o número de amostras. Porém, não tem um bom desempenho quando temos um grande conjunto de dados, devido ao tempo de treinamento, ou quando o conjunto de dados tem mais ruído (ANDREOLA, 2009).

O uso do kernel oferece uma solução alternativa para projeção dos dados aumentando o desempenho da SVM, sem aumentar o número de parâmetros ajustados. Uma das maiores vantagens da SVM é a sua flexibilidade e a utilização da representação dual

torna isto possível, pois os dados de treinamento nunca aparecem isolados, mas na forma de produto interno entre pares de amostras. Com a utilização desses conceitos pode-se adaptar o problema de classificação binária (apenas com duas classes), que foi a abordagem que originou a concepção da SVM, para resolver outros tipos de problemas ([SINGLA et al., 2011](#)).

3 Metodologia

Utilizamos técnicas de inteligência artificial, que possibilitam encontrar um conjunto de características para classificar proteínas (PSE) e comparar os resultados entre estas técnicas.

Com o programa *in house* denominado *valifasta* (SANTOS, 2013), que remove caracteres de pontuação, quebras de linha e outros prováveis caracteres ocasionalmente inseridos em seqüências de proteínas. O mesmo também garante que uma proteína tem uma chave de identificação única dentre as contidas em um mesmo arquivo denominado *multifasta*.

O *multifasta* tem o formato com 20 letras dos aminoácidos e uma identificação simples para cada proteína. O código abaixo é uma representação de como é uma proteína em formato FASTA e sua conversão em formato de linha após utilização do *valifasta*.

- Representação das proteínas em formato fasta antes e após conversão com o valifasta.

```
$cat sequences.fasta
>Cp1002_0126 | Hypothetical protein
| Corynebacterium pseudotuberculosis strain 1002
MHFKTRMSLFCTATTAATSLAVASLQPAAAVEQPSNTIVSTIMLPTKATVTKTFTV+SSTKGTARADYSSN
SITVQPGDTISVKIHSQGGY-TEFSELTEFVPSVGRLLHTESITFKEGDSGPHPLKVAGWNATSQADRVTFR
TNDGKPKAITLDTTLETTYT*VGV RATGDPSTRFQLSSSDSNTVFTSASGPKIHVKKTLPSWLSGAFPGAIF
DLSLTNLLSPILRALNIL
>Cp1002_1802 | Putative sterase alpha-B chain
| Corynebacterium pseudotuberculosis strain 1002
MLFPSRFQGTFLKPLITAALAV*FCVGFTPATAQVIPYTPDPGFYTSIPSAENTTPGTVLSQRDVPMPVLD
+VLVKMKRIAYTSTHPNGFSTPVTGAVLLPTAPWRGPGPR-PVALLAPGTQGAGDSCAPSKLLTMGGEYEM
FSAAALLNRGWTVAVTDYQGLGTPGNHTYMNRKAQGAAL.LDLGRAITTLNLPDNNHTPIIPWGYSQGGG
ASAAAAEMHRAYAPDVNVVVLAYAGGVPANLLSVSSSLEGTA*LTGALGYVITGMYEIYPEIREPIHNFLNT
RGQVWLDQTSRDCLPESLLTMPLPDTSILTVSGQRLTSLI+SDDVFQRAISEQQIGLTAPDIPVFVAQGLN
DGIIPAEQARIMVNGWLSQGADV TYW#EDPSPALDKLSGHIHV LASSFLPAVEWAEQRLAALGQPTP

$valifasta -i sequences.fasta -o valifasta.fasta
$cat valifasta.fasta
>Cp1002_0126a
MHFKTRMSLFCTATTAATSLAVASLQPAAAVEQPSNTIVSTIMLPTKATVTKTFTVSSTKGTARADYSSNS
ITVQPGDTISVKIHSQGGYTEFSELTEFVPSVGRLLHTESITFKEGDSGPHPLKVAGWNATSQADRVTFRNT
DGKPKAITLDTTLETTYTVGV RATGDPSTRFQLSSSDSNTVFTSASGPKIHVKKTLPSWLSGAFPGAIFDS
LTNLLSPILRALNIL
>Cp1002_1802
```

MLFPSRFQGTFLKPLITAALAVFCVGFPTATAQVIPYTDPDGFYTSIPSAENTTPGTVLSQRDVPMPVLDV
 LVKMKRIAYTSTHPNGFSTPVTGAVLLPTAPWRGPGPRPVALLAPGTQGAGDSCAPSKLLTMGGEYEMFSA
 AALLNRGWTVAVTDYQGLGTPGNHTYMNRKAQGAALLDLGRAITTLNLPDVNNHTPIIPWGYSGGGASAA
 AAEMHRAYAPDVNVVLAYAGGVPANLLSVSSSLEGTALTGALGYVITGMYEIYPEIREPIHNFNLRGQVW
 LDQTSRDCLPESLLTMLPDTSLITVSGQRLTSLISDDVFQRAISEQQIGLTAPDIPVFVAQGLNDGIIPA
 EQARIMVNGWLSQGADVTYWEDPSPALDKLSGHIHVLASSFLPAVEWAEQRLAALGQPTP

Utilizamos quatro grupos de classificação de aminoácidos, conforme a Figura 10, na busca pelos descritores. Aminoácidos de característica: básica são representados pela cor verde, não polar (hidrofóbicos) representados pela cor azul, polar (hidrofílicas) com a cor roxa e ácidas são representadas pela cor alaranjada. Assim, criamos as características para o treinamento do modelo considerando a quantidade de aminoácidos com a mesma característica química (polar, não polar, básica, ácida).

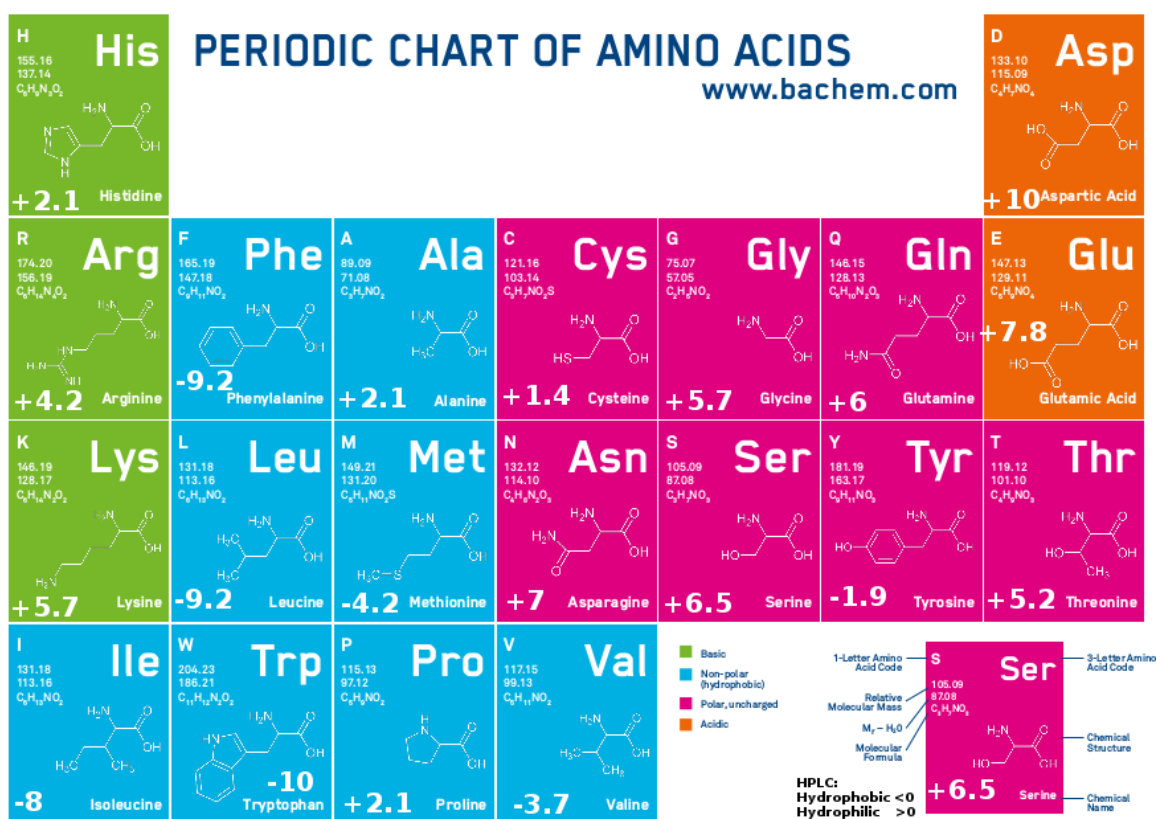
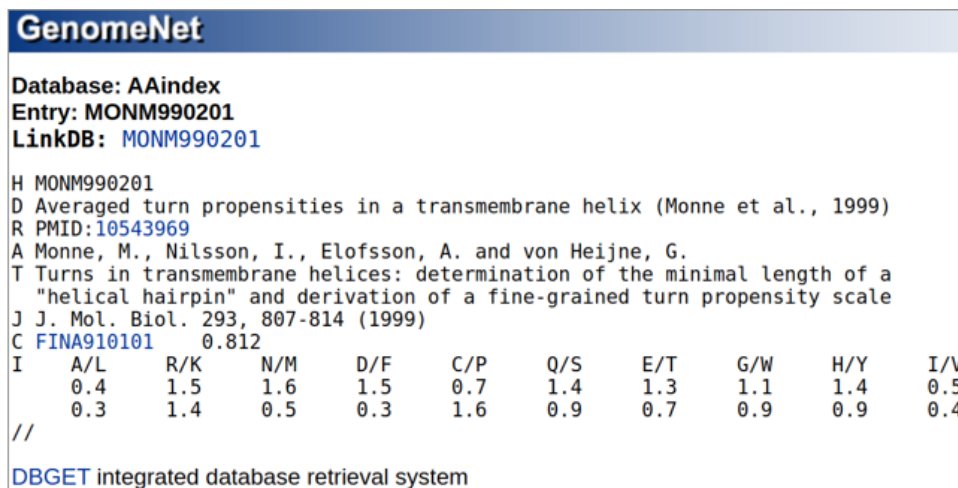


Figura 10 – Tabela química dos aminoácidos

Fonte: (MERGLER, 2020)

Além disso, buscamos no repositório "AAindex1" (KAWASHIMA; OGATA; KANEHISA, 1999) para formar o conjunto de descritores, esse repositório (genome.jp/) contém centenas de índices de propensão para os 20 aminoácidos mais comuns que consideram médias numéricas para características químicas, físicas e estruturais de aminoácidos em

um conjunto de proteínas significativamente representativo dos organismos conhecidos. A Figura 11 é uma captura de tela de um índice de propensão de aminoácidos retirada do AAindex que pretendemos utilizar nesse trabalho.



GenomeNet

Database: AAindex
 Entry: MONM990201
 LinkDB: MONM990201

H MONM990201
 D Averaged turn propensities in a transmembrane helix (Monne et al., 1999)
 R PMID:10543969
 A Monne, M., Nilsson, I., Elofsson, A. and von Heijne, G.
 T Turns in transmembrane helices: determination of the minimal length of a "helical hairpin" and derivation of a fine-grained turn propensity scale
 J J. Mol. Biol. 293, 807-814 (1999)
 C FINA910101 0.812

I	A/L	R/K	N/M	D/F	C/P	Q/S	E/T	G/W	H/Y	I/V
	0.4	1.5	1.6	1.5	0.7	1.4	1.3	1.1	1.4	0.5
	0.3	1.4	0.5	0.3	1.6	0.9	0.7	0.9	0.9	0.4

//

DBGET integrated database retrieval system

Figura 11 – Índice de propensão de aminoácidos - AAindex

Fonte: (KAWASHIMA; OGATA; KANEHISA, 1999)

Com o software WEKA utilizamos a RF, treinando um modelo para encontrar as melhores características que identifiquem se uma proteína é PSE ou não. Nessa etapa, foi usado o conceito de Validação Cruzada (VC). A VC divide o conjunto de treinamento em N subgrupos, treina um modelo de aprendizado em N-1 grupos e testa em um grupo. O processo se repete até que os N subgrupos tenham assumido o papel do grupo de testes. O desempenho do modelo de aprendizado final é uma média dos resultados dessas N execuções. Assim, acontece o processamento das características obtidas na busca por descritores para o treinamento da RF.

O processamento calcula a frequência de aminoácidos das cadeias de proteínas com base nos descritores eleitos como as *features* de cada árvore da RF, assim como as frequências dos 20 aminoácidos e os valores atribuídos de acordo com o AAindex e os descritores selecionados. A Tabela 2 enumera os pesos que representam as características de treinamento selecionados para esta pesquisa.

Para formatar os dados em um modelo de treinamento do software WEKA foi desenvolvido um programa in house chamado *features* (Linguagem Common Lisp), disponível no Anexo A. O programa *features* processa um arquivo *multifasta* gerando um arquivo CSV contabilizando os valores numéricos de todos os descritores selecionados. Os descritores selecionados (Tabela 2), e seus respectivos pesos para os 20 aminoácidos, são armazenados em um arquivo-texto de modo a evitar a recompilação do programa a cada descritor alterado.

O WEKA é um software, de código aberto emitido sob a *GNU General Public*

Reference	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
BASIC	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ACID	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
POLAR	0.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0
NON-POLAR	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0	1.0
CHAM830103	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	2.0	1.0	1.0	1.0	1.0	0.0	1.0	2.0	1.0	1.0	2.0
CHAM830104	0.0	1.0	1.0	1.0	0.0	1.0	1.0	0.0	1.0	2.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0
CHAM830105	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	1.5	1.0	0.0
FAUJ880111	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
MONM990201	0.4	1.5	1.6	1.5	0.7	1.4	1.3	1.1	1.4	0.5	0.3	1.4	0.5	0.3	1.6	0.9	0.7	0.9	0.9	0.4
MONM990101	0.5	1.7	1.7	1.6	0.6	1.6	1.6	1.3	1.6	0.6	0.4	1.6	0.5	0.4	1.7	0.7	0.4	0.7	0.6	0.5

Tabela 2 – Índices de propensão de aminoácidos utilizados para gerar descritores de proteínas na busca por classificar proteínas PSE

License, de ML desenvolvido em Java, contendo um acervo de algoritmos para tarefas de mineração de dados. Possui também ferramentas para preparação de dados, classificação, regressão, clustering (agrupamento), regras de mineração e visualização (HALL et al., 2009).

- Arquivo CSV resultado do processamento do arquivo valifasta.fasta pelo programa features. Esse formato é genérico. Um script bash ainda precisa converter o CSV para o formato ARFF do programa WEKA.

```
ALRKNMDFCPQSETGWHYIVBASICACIDPOLARNON-POLARCHAM830103CHAM830104CHAM830105FAU
J880111MONM990201MONM990101INIBASICINIACIDINIPOLARININON-POLARINICHAM830103I
NICHAM830104INICHAM830105INIFAUJ880111INIMONM990201INIMONM990101ENDBASICENDA
CIDENDPOLARENDNON-POLARENDCHAM830103ENDCHAM830104ENDCHAM830105ENDFAUJ880111E
NDMONM990201ENDDMONM990101MIDBASICMIDACIDMIDPOLARMIDNON-POLARMIDCHAM830103MID
CHAM830104MIDCHAM830105MIDFAUJ880111MIDMONM990201MIDMONM990101
```

```
Cp1002_0126a20879167145111812311132834241525.016.094.093.0241.0114.059.025.0
199.00002202.999983.01.015.021.040.016.010.03.030.030.46.02.023.029.059.034
.012.56.051.753.10000211.010.029.026.078.043.024.511.073.377.59999
```

```
Cp1002_180245161419318143872045610123727357132929.033.0148.0205.0379.0221.01
06.529.0368.1389.899932.01.012.025.038.020.011.02.030.832.45.07.017.031.051
.034.020.55.054.759.110.09.053.067.0121.076.035.010.0120.50001128.0
```

Esse arquivo CSV, representado acima, de exemplo possui três linhas: uma de cabeçalho e duas de dados. A linha de cabeçalho lista os descritores; cada proteína possui um rótulo e os números referentes para cada descritor. Para cada descritor da Tabela 2, são derivados outros três que analisam as porções de início, meio e fim de cada proteína. O motivo dessa derivação de descritores reside no fato que, de acordo com a localização celular, proteínas podem ter padrões de aminoácidos com características físico-químicas distintas nessas porções proteicas.

A título de exemplo, as proteínas secretadas geralmente possuem padrões hidrofóbicos apenas em sua porção inicial. O arquivo CSV é formatado para um ARFF, padrão

de entrada de dados do software WEKA, por meio de um script bash do SO Linux, representado abaixo.

- Programa em bash para converter o CSV genérico para ARFF utilizado pelo WEKA.

```
features valifasta.fasta > weka.arff
head -n 1 weka.arff > weka.attributes
end='head -n 1 weka.arff | wc -w'; sed -i "s/\t$/g" weka.arff; echo
"$end features"
sed -i '1d' weka.arff
sed -i "s/\([a-zA-Z0-9]\+\)/@attribute \1 numeric#/g" weka.attributes
tr '#' '\n' < weka.attributes > weka.attributes2
tr -d '\t' < weka.attributes2 > weka.attributes
./addlocal local weka.arff #insere r tulos das inst ncias de dados
cut -f 2- weka.arff > weka.arff2
sed -i "s/\t/,/g" weka.arff2
echo '@relation localsubcellular' > localsubcellular.arff
cat weka.attributes >> localsubcellular.arff
echo '@attribute class {POSITIVE, NEGATIVE}' >> localsubcellular.arff
echo '@data' >> localsubcellular.arff
cat weka.arff2 >> localsubcellular.arff
```

Desta forma, convertemos uma proteína com representação de aminoácidos em letras para uma cadeia de números, padrão WEKA de entrada de dados, com as características sinalizadoras de secreção não clássica. A representação do rótulo de cada instância de dados de treinamento é inserida no arquivo ARFF pelo script bash, por meio de uma lista previamente confeccionada com as classificações. Estes rótulos das instâncias de treinamento podem ser, por exemplo, “Positive” ou “Negative”. Um exemplo de formatação das proteínas para o formato ARFF do WEKA está no Anexo B.

Além das proteínas coletadas na literatura, utilizamos também o conjunto de proteínas treinadas pelo software *PeNGaRoo* (ZHANG et al., 2019). Este conjunto de dados é composto por dois subgrupos: um grupo para treinamento e um para teste, contendo, respectivamente, 1400 e 1100 proteínas. Ambos os grupos contêm dois arquivos sendo o conjunto positivo as proteínas PSE e o grupo negativo. Esses arquivos estão em um repositório com o nome de *Training Dataset* e *Independent Dataset* no servidor do *PeNGaRoo*.

No final, para comparar os resultados obtidos com a RF, utilizamos os algoritmos MLP e SMO para o treinamento do modelo na classificação de proteínas PSE. Além disso, análise dos dados foi realizada com VC em 10 *folds* e os dados que forem obtidos para analisar contém taxa de acerto, onde esta é calculada com base na correlação entre os dados e os erros, são eles: erro absoluto médio e raiz quadrada do erro médio.

4 Resultados e discussões

Neste capítulo, serão apresentados os experimentos realizados utilizando a metodologia já apresentada. Nele, encontra-se cada etapa dos testes efetuados nas características propostas com a RF, comparação de desempenho com outros algoritmos e uma análise dos resultados.

4.1 Classificação de proteínas PSE com RF

Mantendo em 10 *folds* a validação cruzada e variando as quantidades de árvores utilizadas e de *features*, para cada árvore obtivemos uma acurácia média de 72,83% na classificação das proteínas PSE, variando de 71,62% a 73,19%. Na tabela 3, são apresentadas as execuções da RF no WEKA e os resultados.

Execuções	Quantidade de Árvores	Quantidade de Features	acurácia
Execução 1	50	3	72,86%
Execução 2	50	6	72,48%
Execução 3	50	12	71,62%
Execução 4	100	3	72,29%
Execução 5	100	6	72,95%
Execução 6	100	12	73,14%
Execução 7	200	3	72,81%
Execução 8	200	6	73,19%
Execução 9	200	12	73,00%
Execução 10	500	12	72,71%

Tabela 3 – Experimentos PSE com RF

4.2 Classificação de proteínas PSE com outros Algoritmos

Após a execução dos dados no WEKA utilizando RF, realizamos os testes dos dados com outros algoritmos de inteligência artificial, sendo eles MLP (Perceptron Multi-camadas) e SMO (Otimização sequencial mínima) que é um algoritmo para treinamento de máquinas de suporte vetorial. Os testes com esses algoritmos foram realizados para comparar as soluções para o problema proposto.

4.2.1 MLP

Na tabela 4, estão os resultados da execução do algoritmo MLP variando a taxa de aprendizado e o número de camadas ocultas da rede, assim como, quantos neurônios tem em cada camada. A acurácia do algoritmo foi em média de 65,67%, variando de 63,67% a 67,62%.

Execuções	camadas ocultas	Neurônios em cada camada Oculta	Taxa de aprendizado	Épocas	acurácia
Execução 1	1	12	0,3	500	65,67%
Execução 2	2	12	0,3	500	65,33%
Execução 3	1	12	0,3	1000	67,62%
Execução 4	2	12	0,3	1000	65,24%
Execução 5	1	12	0,5	500	64,81%
Execução 6	2	12	0,5	500	63,67%
Execução 7	1	12	0,5	1000	65,67%
Execução 8	2	12	0,5	1000	66,05%
Execução 9	1	24	0,3	500	66,57%
Execução 10	2	24	0,3	500	66,10%

Tabela 4 – Resultados: Experimentação PSE com MLP

Função de ativação usada foi a Sigmoide que variando de 0 a 1 tenta "empurrar" a saída da rede para os extremos.

$$f(x) = 1/(1 + e^{-x}) \quad (4.1)$$

4.2.2 SMO

Na tabela 5, estão apresentados os resultados da execução do algoritmo SMO para classificar as proteínas. A classificação com o SMO foi em média de 65,60%, variando de 63,81% a 70,62% de acurácia.

Execuções	Dados	constante C	Kernel-Expoente	acurácia
Execução 1	normalizados	1	1	63,81%
Execução 2	padronizados	1	1	65,43%
Execução 3	normalizados	5	1	65,19%
Execução 4	normalizados	10	1	65,43%
Execução 5	normalizados	20	1	65,24%
Execução 6	normalizados	30	1	65,76%
Execução 7	normalizados	1	2	68,33%
Execução 8	normalizados	30	2	69,76%
Execução 9	normalizados	1	3	70,62%
Execução 10	normalizados	1	4	70,24%

Tabela 5 – Resultados: Experimentação PSE com SMO

Obs.: outros testes com dados padronizados não tiveram desempenho.

- Parâmetros SMO:

- Kernel: mede a similaridade/distancia entre o vetor de entrada e o de treinamento.
- C: constante de complexidade
- E: expoente usado no cálculo do Kernel

4.3 Análise dos Resultados

Comparando os algoritmos para solução do problema proposto, podemos ver que as RF tiveram resultados parecidos. O tempo de execução dos algoritmos RF e MLP se aproxima e por isso não foi um ponto usado para comparação, mas o SMO foi o algoritmo menos eficiente, levando até mesmo uma hora para a execução do modelo.

Com as execuções desses algoritmos, pode-se concluir que a utilização das RF é válida, assim como o MLP e SMO, para solução o problema proposto, considerando que 73% de acurácia é aceitável na classificação de proteínas PSE, pois, uma vez que, compartilham domínios hidrofóbicos e isto aumenta a dificuldade em identificá-las, já que são um meio-termo entre de membrana e secretadas.

5 Considerações Finais

O objetivo do trabalho foi testar algoritmos populares em ML utilizando WEKA, para classificação das proteínas PSE. Para isto, foi necessário uma revisão da literatura sobre a classificação de proteínas PSE e sobre os algoritmos de ML escolhidos, sendo eles: RF, MLP e SMO.

A motivação é que classificar essas proteínas possibilita a produção de vacinas para os patógenos causados pelas bactérias e, com a vacina, ser possível a inativação das proteínas expostas na superfície, podendo ser eficiente para a inativação da bactéria.

Para a execução desse estudo foi necessário a construção da base de dados, contendo os descritores das proteínas. O modelo de RF treinado com o WEKA foi usado para poder classificá-las. Após os testes realizados com outros algoritmos podemos comparar a eficiência das RF para a solução do problema proposto.

No geral, os resultados dos experimentos nos mostraram o bom desempenho das RF para classificar as proteínas e, também, que os algoritmos MLP e SMO tiveram resultados próximos. Portanto, pode-se concluir que a utilização dos algoritmos usados nesta pesquisa é válida para solução o problema proposto, uma vez que, consideramos a dificuldade na classificação de proteínas PSE pelos domínios hidrofóbicos presentes na mesma.

Propõe-se o desenvolvimento de um algoritmo genético, para buscar quais descritores produzem o melhor resultado de classificação. Explorando assim, uma quantidade de grandeza fatorial de possibilidades de agrupamentos para todos os índices de propensão de aminoácidos presentes no AAindex, e também a utilização de dados que não participaram do treinamento do modelo ajudaria a evitar o *overfitting* do modelo aos dados.

Referências

- ANDREOLA, R. Support vector machines na classificação de imagens hiperespectrais. 2009. Citado na página 25.
- ARAÚJO, N. D. de et al. A ERA DA BIOINFORMÁTICA: SEU POTENCIAL e SUAS IMPLICAÇÕES PARA AS CIÊNCIAS DA SAÚDE. *Estudos de Biologia*, Pontificia Universidade Católica do Paraná - PUCPR, v. 30, n. 70/72, nov 2008. Acesso em: 2022. Citado na página 21.
- BREIMAN, L. Random forests. machine learning. *Machine Learning*, Springer Science and Business Media LLC, v. 45, n. 1, p. 5–32, 2001. Acesso em: 2021. Citado na página 23.
- CHOU, K.-C.; SHEN, H.-B. MemType-2l: A web server for predicting membrane proteins and their types by incorporating evolution information through pse-PSSM. *Biochemical and Biophysical Research Communications*, Elsevier BV, v. 360, n. 2, p. 339–345, aug 2007. Acesso em: 2021. Citado na página 13.
- CORDWELL, S. J. Technologies for bacterial surface proteomics. *Current Opinion in Microbiology*, Elsevier BV, v. 9, n. 3, p. 320–329, jun 2006. Acesso em: 2022. Citado na página 19.
- COSTA J.V. ; SANTOS, A. *Predição de proteínas de bactérias secretadas por via não clássicas*. 2018. Projeto de pesquisa submetido como requisito parcial ao PIBIC Edital Número 06/2018. Acesso em: 2021. Citado 2 vezes nas páginas 9 e 20.
- FUJIBUCHI, W. et al. Dbget/linkdb: an integrated database retrieval system. <<http://www.genome.ad.jp/>>. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, p. 683–694, 1998. ISSN 2335-6928. Acesso em: 2021. Citado na página 15.
- HALL, M. et al. The WEKA data mining software. *ACM SIGKDD Explorations Newsletter*, Association for Computing Machinery (ACM), v. 11, n. 1, p. 10–18, nov 2009. Acesso em: 2021. Citado 2 vezes nas páginas 15 e 30.
- KAWASHIMA, S.; OGATA, H.; KANEHISA, M. AAindex: Amino acid index database. *Nucleic Acids Research*, Oxford University Press (OUP), v. 27, n. 1, p. 368–369, jan 1999. Acesso em: 2021. Citado 2 vezes nas páginas 28 e 29.
- KROGH, A. et al. Predicting transmembrane protein topology with a hidden markov model: application to complete genomes¹edited by f. cohen. *Journal of Molecular Biology*, Elsevier BV, v. 305, n. 3, p. 567–580, jan 2001. Acesso em: 2021. Citado na página 14.
- LUSCOMBE, N.; GREENBAUM, D.; GERSTEIN, M. What is bioinformatics? an introduction and overview. *Yearbook of Medical Informatics*, Georg Thieme Verlag KG, v. 10, n. 01, p. 83–100, aug 2001. Acesso em: 2022. Citado na página 21.

- MERGLER, M. *Periodic Chart Of Amino Acids Table*. 2020. Disponível em: <<https://www.bachem.com/knowledge-center/posters/amino-acid-chart-and-its-20-proteinogenic-amino-acids/>>. Acesso em: 2021. Citado na página 28.
- MULEY, V. Y.; ACHARYA, V. *Genome-Wide Prediction and Analysis of Protein-Protein Functional Linkages in Bacteria*. [S.l.]: Springer, 2012. ISBN 9781461447054. Acesso em: 2021. Citado na página 19.
- NORIEGA, L. Multilayer perceptron tutorial. *School of Computing. Staffordshire University*, Citeseer, 2005. Citado na página 24.
- NORVIG, P. *Inteligência Artificial*. [S.l.]: Elsevier, 2013. ISBN 9788535237016. Acesso em: 2021. Citado 2 vezes nas páginas 24 e 25.
- OWH, O. W. H. *Global Tuberculosis Report 2015*. World Health Organization, 2015. 170 p. ISBN 9789241565059. Disponível em: <http://apps.who.int/iris/bitstream/handle/10665/191102/9789241565059_eng.pdf;jsessionid=01F9731FF186B64BDFE177ED6D6F8B1A?sequence=1>. Acesso em: 2021. Citado na página 13.
- OWH, O. W. H. *Global tuberculosis report 2021*. World Health Organization, 2021. ISBN 9789240037021. Disponível em: <<https://www.who.int/publications/i/item/9789240037021>>. Acesso em: 2021. Citado na página 13.
- PLATT, J. Sequential minimal optimization: A fast algorithm for training support vector machines. *Advances in Kernel Methods-Support Vector Learning*, v. 208, 07 1998. Acesso em: 2021. Citado na página 25.
- SANTOS, A. R. 2013. Citado 2 vezes nas páginas 14 e 27.
- SANTOS, A. R. et al. Mature epitope density - a strategy for target selection based on immunoinformatics and exported prokaryotic proteins. *BMC Genomics*, Springer Science and Business Media LLC, v. 14, n. S6, oct 2013. Acesso em: 2021. Citado na página 15.
- SAÚDE, M. da. *Manual de Recomendações para o Controle da Tuberculose no Brasil*. Ministério daSaúde, Secretaria de Vigilância em Saúde, Departamento de Vigilância das Doenças Transmissíveis., 2019. ISBN 978-85-334-2696-2. Disponível em: <https://bvsms.saude.gov.br/bvs/publicacoes/manual_recomendacoes_controle_tuberculose_brasil_2_ed.pdf>. Acesso em: 2021. Citado na página 13.
- SINGLA, R. et al. Comparison of SVM and ANN for classification of eye events in EEG. *Journal of Biomedical Science and Engineering*, Scientific Research Publishing, Inc., v. 04, n. 01, p. 62-69, 2011. Acesso em: 2022. Citado na página 26.
- SOUTO, M. D. et al. Técnicas de aprendizado de máquina para problemas de biologia molecular. *Sociedade Brasileira de Computação*, v. 1, n. 2, 2003. Acesso em: 2022. Citado na página 22.
- SU, C.-H. et al. Identification of amino acid propensities that are strong determinants of linear b-cell epitope using neural networks. *PLoS ONE*, Public Library of Science (PLoS), v. 7, n. 2, p. e30617, feb 2012. Acesso em: 2021. Citado na página 14.

VERLI, H. *Bioinformática: da Biologia à Flexibilidade Molecular*. 1. ed. [S.l.]: Sociedade Brasileira de Bioquímica e Biologia Molecular - SBBq, 2014. ISBN 978-85-69288-00-8. Acesso em: 2022. Citado 2 vezes nas páginas [17](#) e [19](#).

WHITELEGGE, J. P. Sequencing covalent modifications of membrane proteins. *Journal of Experimental Botany*, Oxford University Press (OUP), v. 57, n. 7, p. 1515–1522, mar 2006. Acesso em: 2022. Citado na página [18](#).

ZHANG, Y. et al. PeNGaRoo, a combined gradient boosting and ensemble learning framework for predicting non-classical secreted proteins. *Bioinformatics*, Oxford University Press (OUP), aug 2019. Acesso em: 2021. Citado na página [31](#).

Anexos

ANEXO A – Código fonte do programa features.lisp

```

;rlwrap sbcl --dynamic-space-size 1024 --load features.lisp
;(save-lisp-and-die "features" :executable t :save-runtime-options
t :compression 9 :toplevel 'main)
;(save-lisp-and-die "features" :executable t :save-runtime-options
t :toplevel 'main)
;The global set of the random state does not work for the
executable version of the program.
;I will keep it here but changed all random calls to force the
state update.
;(setq *random-state* (make-random-state t))
(defvar *printlist* nil "Vector to store the final result of the
program features in a format to produce the final output of the
program")
;Given a text string, a field delimiter and a position this
function returns the text at the specific location.
(defun nth-string (pos tabular delim)
(let ( (inicio 0) (fim) (retorno) (max (length tabular)) (cont 0) )
;do while not
(do ()
((or (< pos 0) (= cont pos) (> inicio max))))
(setq fim (position delim tabular :start inicio))
(if (not fim) (setq fim max))
(setq retorno (subseq tabular inicio fim)
inicio (+ fim 1)
cont (1+ cont)
);setq
);do
(if (< cont pos) nil retorno)
);let
);defun
(defun parse-float (s)
(let ((readval (handler-case
(read-from-string s)
(sb-int:simple-reader-error nil)
(end-of-file nil))))
(cond ((realp readval ) (+ readval 0.0))
(t (error (concatenate 'string "not a float: " s))))))
;Returns a list where each component has a pair of text identifier
for a fasta sequence
;and correspondent amino acids as a vector.

```



```

    (list (nth-string 1 line #\Space))
  );append
  filecontent (append filecontent
    (list (loop for i from 2 to 21 collect
      (parse-float (nth-string i line #\Space))
    );loop
    );list
  );append
  );setf
);if
  );when there is content
  );loop to read number sequence
(close inputfile)
);when inputfile
(values filelabels filecontent) ;return value
);let
);defun
  ; Collect a list of needle's positions in a haystack
;(all-positions 'G *sequence*)
(defun all-positions (needle haystack)
  (loop
    for element in haystack
    and position from 0
    when (eql element needle)
    collect position))
  ;Determines the size, mean and standard deviation from a numerical
  sample,
  ;in general, a list of positions of a pattern
(defun statistics( data )
  (if data
    (let ( (n 0) (mean 0.0) (var 0.0) (diff) )
      (loop for x across data do
        (incf n)
        (setq diff (- x mean))
        var (+ var (/ (* diff diff (- n 1)) n))
        mean (+ mean (/ diff n))
      );setq
    );loop
    (if (> n 1)
      (values n mean (sqrt (/ var (- n 1))) )
      (values n mean (sqrt (/ var n ) ) )
    )
  );let
);if
);defun
  ;Considering that each search pattern is a list of lisp symbols, I
  need to convert the sequence to a string.

```

```

(defun convert-to-key ( nucseq )
  (let ( (var "") )
    (loop for i in nucseq do
      (setf var (concatenate 'string var (symbol-name i )))
    )
    var
  )
)

(defun histo-fasta( filename )
  (let ( (multifastalist (read-fasta filename ))
        (aataargets (list 'A 'R 'N 'D 'C 'Q 'E 'G 'H 'I 'L 'K 'M 'F 'P 'S 'T
                          'W 'Y 'V))
        (histolist nil)
        (fastaname)
        (histogram nil)
  )
    (dolist (fastalist multifastalist histogram)
      (setq fastaname (car fastalist)
            histolist nil)
      );setq
      (dolist (aa aataargets histolist)
        (setq histolist (append histolist (list (count aa (cadr fastalist))))))
      );inner do
      (setq histogram (append histogram (list (list fastaname histolist))))
      );outer do
      histogram
    );let
  )

(defun dohistogram( fastalist )
  (let ( (aataargets (list 'A 'R 'N 'D 'C 'Q 'E 'G 'H 'I 'L 'K 'M 'F
                          'P 'S 'T 'W 'Y 'V))
        (histolist nil)
  )
    (setq histolist nil);setq
    (dolist (aa aataargets histolist)
      (setq histolist (append histolist (list (count aa fastalist))))
    );do
    histolist
  );let
)

(defun dohistogramini( fastalist )
  (let ( (aataargets (list 'A 'R 'N 'D 'C 'Q 'E 'G 'H 'I 'L 'K 'M 'F
                          'P 'S 'T 'W 'Y 'V))
        (histolist nil)
        (size (length fastalist))
        (limit 40)
  )

```

```

)
(dolist (aa aatargets histolist)
  (setq histolist (append histolist (list (count aa fastalist :end (if
    (> size limit) limit size) ))))
);do
histolist
);let
)
(defun dohistogramend( fastalist )
  (let ( (aatargets (list 'A 'R 'N 'D 'C 'Q 'E 'G 'H 'I 'L 'K 'M 'F
    'P 'S 'T 'W 'Y 'V))
    (histolist nil)
    (size (- (length fastalist) 60))
  )
    (dolist (aa aatargets histolist)
      (setq histolist (append histolist (list (count aa fastalist :start
        (if (> size 0) size 0) ))))
    );do
    histolist
  );let
  )
  (defun dohistogrammid( fastalist )
    (let ( (aatargets (list 'A 'R 'N 'D 'C 'Q 'E 'G 'H 'I 'L 'K 'M 'F
      'P 'S 'T 'W 'Y 'V))
      (histolist nil)
      (size (length fastalist) )
      (ratio)
      (start)
      (end)
    )
      (setf ratio (round (/ size 3))
        start (if (> size ratio) ratio size)
        end (- size ratio)
        end (if (> end 0) end size))
      (dolist (aa aatargets histolist)
        (setq histolist (append histolist (list (count aa fastalist :start
          start :end end ))))
      );do
      histolist
    );let
  )
  ;The main function settles everything for the execution of the
  program.
  ;This function also is the entry point for the executable created
  from this lisp code.
  (defun main ()
    (if (> (length sb-ext:*posix-argv*) 1)

```

```
(let (
  ;primeiro par metro      o nome do arquivo
  (fastafila nil)
  (propensityfile nil)
  (searchspace)
  (milestone)
  (checkmilestone)
  (multifasta)
  (multifasta-size)
  (listof-sequence-names)
  (sequence)
  (sequence-name)
  (sequence-size)
  (sequence-number 0)
  (sequence-histogram)
  (pos)
  (histoalphabet '( A L R K N M D F C P Q S E T G W H Y I V ))
  (alphabet)
  (physicochemicals)
); let pars

(setf fastafila (if (nth 1 sb-ext:*posix-argv*) (nth 1
sb-ext:*posix-argv*) ) );setf
(if (not (probe-file fastafila))
  (progn
    (format t "~%~a~%" "ERROR: Amino acid fasta file is missing")
    (SB-EXT:EXIT)
  )
);if

(setf propensityfile (if (nth 2 sb-ext:*posix-argv*)
(nth 2 sb-ext:*posix-argv*)
"propensity.dat")
);setf
(if (not (probe-file propensityfile ))
  (progn
    (format t "~%~a~%" "ERROR: Propensity data file is missing")
    (SB-EXT:EXIT)
  )
);if

(multiple-value-bind ( labels props) (read-propensity propensityfile)
  (setf
    labels (append labels
  (loop for place in (list "INI" "END" "MID") append
    (loop for label in labels collect (concatenate 'string place
label)))
  )
)
```

```

)
  alphabet (append histoalphabet labels)
  physicochemicals props
  multifasta (read-fasta fastafile)
  multifasta-size (length multifasta)
  *printlist* (make-array (list (+ multifasta-size 1) (+ (length
alphabet) 1) ) :initial-element nil)
  );setf
);multiple-value-bind

;replicates the labels for the begin, middle and end of each
protein sequence
;come a o processamento de todas as sequencias
(loop for fasta in multifasta do
  (setf sequence (cadr fasta)
  sequence-name (car fasta)
  listof-sequence-names (append listof-sequence-names (list
sequence-name))
  sequence-size (length sequence)
  sequence-histogram (dohistogram sequence)
  searchspace sequence-size
  milestone (round (* sequence-size 0.01))
  milestone (if (zerop milestone) 1 milestone)
  checkmilestone milestone
  );setf

;come a o processamento para uma sequencia
(loop for pos from 0 to (- (length histoalphabet) 1) do
;medidor de progresso
; (when (= (mod pos checkmilestone) 0)
; (setq checkmilestone (+ checkmilestone milestone))
;(print (round (* 100 (/ checkmilestone (- searchspace 1))))))
; );when checkmilestone
(setf (aref *printlist* (+ sequence-number 1) (+ pos 1)) (nth pos
sequence-histogram))
);loop

;after the histogram calculation we start to compute the
physicochemicals properties
(setf pos (length histoalphabet))
(loop for physico in physicochemicals do
(setf (aref *printlist* (+ sequence-number 1) (+ pos 1))
(loop for weigth-list in (list physico) sum
(loop for product in (mapcar #'* sequence-histogram weigth-list)
sum product)
)
)
)

```

```
(incf pos)
)

;Giving a try: lets compute the number of aminoacids just at the
region known to host the signal peptide
(setf sequence-histogram (dohistogramini sequence))
(loop for physico in physicochemicals do
(setf (aref *printlist* (+ sequence-number 1) (+ pos 1))
(loop for weigth-list in (list physico) sum
(loop for product in (mapcar #'* sequence-histogram weigth-list)
sum product)
)
)
(incf pos)
)

;Giving another try: lets compute the number of aminoacids just at
the end of the protein
(setf sequence-histogram (dohistogramend sequence))
(loop for physico in physicochemicals do
(setf (aref *printlist* (+ sequence-number 1) (+ pos 1))
(loop for weigth-list in (list physico) sum
(loop for product in (mapcar #'* sequence-histogram weigth-list)
sum product)
)
)
(incf pos)
)

;Giving another try: lets compute the number of aminoacids just at
the middle
(setf sequence-histogram (dohistogrammid sequence))
(loop for physico in physicochemicals do
(setf (aref *printlist* (+ sequence-number 1) (+ pos 1))
(loop for weigth-list in (list physico) sum
(loop for product in (mapcar #'* sequence-histogram weigth-list)
sum product)
)
)
(incf pos)
)

(incf sequence-number)
);loop for multifasta

;final pretty print
;write the name of the sequences in the first line of the
```



```
*printlist*
;In the first line, the first column of *printlist* must be an
empty string. Starting the names from the second column or x=1
(setf (aref *printlist* 0 0) "");empty string
;Inserting the attribute names in first line
(loop for y from 1 to (length histoalphabet) do
  (setf (aref *printlist* 0 y)
        (symbol-name (nth (- y 1) histoalphabet))))
(loop for y from (length histoalphabet) to (length alphabet) do
  (setf (aref *printlist* 0 y)
        (nth (- y 1) alphabet)))

;Inserting the name of the sequences in the first column of all
lines
(loop for y from 1 to multifasta-size do (setf (aref *printlist* y 0)
  ) (nth (- y 1) listof-sequence-names)))
;Once we wrote the first row, now we need to write the data
concerning each sequence name
(loop for x from 0 to multifasta-size do

  (loop for y from 0 to (length alphabet) do
    (if (aref *printlist* x y)
      (format t "~a~a" (aref *printlist* x y) #\Tab )
      (format t "~a~a" 0 #\Tab )
    ))
    (format t "~%" )
  )
);let
);if
);defun
```

ANEXO B – Formatação das proteínas no formato ARFF

Exemplo de formatação das proteínas da Figura 1 convertidas para o formato ARFF do WEKA de acordo com a metodologia desse projeto de pesquisa.

```

@relation localsubcellular
@attribute A numeric
@attribute L numeric
@attribute R numeric
@attribute K numeric
@attribute N numeric
@attribute M numeric
@attribute D numeric
@attribute F numeric
@attribute C numeric
@attribute P numeric
@attribute Q numeric
@attribute S numeric
@attribute E numeric
@attribute T numeric
@attribute G numeric
@attribute W numeric
@attribute H numeric
@attribute Y numeric
@attribute I numeric
@attribute V numeric
@attribute BASIC numeric
@attribute ACID numeric
@attribute POLAR numeric
@attribute NON numeric
-@attribute POLAR numeric
@attribute CHAM830103 numeric
@attribute CHAM830104 numeric
@attribute CHAM830105 numeric
@attribute FAUJ880111 numeric
@attribute MONM990201 numeric
@attribute MONM990101 numeric
@attribute INIBASIC numeric
@attribute INIACID numeric
@attribute INIPOLAR numeric
@attribute ININON numeric
-@attribute POLAR numeric
@attribute INICHAM830103 numeric

```

```
@attribute INICHAM830104 numeric
@attribute INICHAM830105 numeric
@attribute INIFAUJ880111 numeric
@attribute INIMONM990201 numeric
@attribute INIMONM990101 numeric
@attribute ENDBASIC numeric
@attribute ENDACID numeric
@attribute ENDPOLAR numeric
@attribute ENDNON numeric
-@attribute POLAR numeric
@attribute ENDCHAM830103 numeric
@attribute ENDCHAM830104 numeric
@attribute ENDCHAM830105 numeric
@attribute ENDFAUJ880111 numeric
@attribute ENDMONM990201 numeric
@attribute ENDMONM990101 numeric
@attribute MIDBASIC numeric
@attribute MIDACID numeric
@attribute MIDPOLAR numeric
@attribute MIDNON numeric
-@attribute POLAR numeric
@attribute MIDCHAM830103 numeric
@attribute MIDCHAM830104 numeric
@attribute MIDCHAM830105 numeric
@attribute MIDFAUJ880111 numeric
@attribute MIDMONM990201 numeric
@attribute MIDMONM990101 numeric
@attribute class {POSITIVE, NEGATIVE}
@data
20,8,7,9,1,6,7,14,5,11,18,12,3,11,13,28,34,2,4,15,25.0,16.0,94.0,93.0,241.0,114.0,
78.0,43.0,24.5,11.0,73.3,77.59999, POSITIVE
45,16,14,19,3,18,14,38,7,20,45,6,10,12,37,27,35,7,13,29,29.0,33.0,148.0,205.0,379.0,
121.0,76.0,35.0,10.0,120.50001,128.0, NEGATIVE
```

ANEXO C – Índice de propensão de aminoácidos disponíveis no repositório AAindex.

Os valores representam respectivamente os aminoácidos: A R N D C Q E G H I L
K M F P S T W Y V

```

BASIC: 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 1.00
        0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
ACID: 0.00 0.00 0.00 1.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00
       0.00 0.00 0.00 0.00
0.00 0.00 0.00
POLAR: 0.00 0.00 1.00 0.00 1.00 1.00 0.00 1.00 0.00 0.00 0.00 0.00
       0.00 0.00 0.00 1.00
1.00 0.00 1.00 0.00
NONPOLAR: 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 1.00 0.00
          1.00 1.00 1.00
0.00 0.00 1.00 0.00 1.00
CHAM830103: 0.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 1.0 2.0 1.0 1.0 1.0 1.0
            0.0 1.0 2.0 1.0 1.0 2.0
CHAM830104: 0.0 1.0 1.0 1.0 0.0 1.0 1.0 0.0 1.0 1.0 2.0 1.0 1.0 1.0
            0.0 0.0 0.0 1.0 1.0 0.0
CHAM830105: 0.0 1.0 0.0 0.0 0.0 1.0 1.0 0.0 1.0 0.0 0.0 1.0 1.0 1.0
            0.0 0.0 0.0 1.5 1.0 0.0
FAUJ880111: 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0
            0.0 0.0 0.0 0.0 0.0 0.0
MONM990201: 0.4 1.5 1.6 1.5 0.7 1.4 1.3 1.1 1.4 0.5 0.3 1.4 0.5 0.3
            1.6 0.9 0.7 0.9 0.9 0.4
MONM990101: 0.5 1.7 1.7 1.6 0.6 1.6 1.6 1.3 1.6 0.6 0.4 1.6 0.5 0.4
            1.7 0.7 0.4 0.7 0.6 0.5
BEGF750101: 1.00 0.52 0.35 0.44 0.06 0.44 0.73 0.35 0.60 0.73 1.00
            0.60 1.00 0.60 0.06 0.35 0.44 0.73 0.44 0.82
BROC820102: 3.9 3.2 -2.8 -2.8 -14.3 1.8 -7.5 -2.3 2.0 11.0 15.0 -2.5
            4.1 14.7 5.6 -3.5 1.1
17.8 3.8 2.1
FAUJ880112: 0.00 0.00 0.00 1.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00
            0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00
GEIM800103: 1.55 0.20 1.20 1.55 1.44 1.13 1.67 0.59 1.21 1.27 1.25
            1.20 1.37 0.40 0.21
1.01 0.55 1.86 1.08 0.64

```

GEIM800105: 0.84 1.04 0.66 0.59 1.27 1.02 0.57 0.94 0.81 1.29 1.10
0.86 0.88 1.15 0.80
1.05 1.20 1.15 1.39 1.56
LEWP710101 0.22 0.28 0.42 0.73 0.20 0.26 0.08 0.58 0.14 0.22 0.19 0.27
0.38 0.08 0.46
0.55 0.49 0.43 0.46 0.08
NAKH900102: 3.73 3.34 2.33 2.23 2.30 2.36 3.00 3.36 1.55 2.52 3.40
3.36 1.37 1.94 3.18
2.83 2.63 1.15 1.76 2.53
NAKH900108: -0.70 -0.91 1.28 -0.93 -0.41 -0.71 -1.13 -0.12 0.04 1.77
1.02 -0.40 0.86 1.29
-0.42 0.14 -0.13 0.26 1.29 -0.19
O0BM850104: -2.49 2.55 2.27 8.86 -3.13 1.79 4.04 -0.56 4.22 -10.87
-7.16 -9.97 -4.96 -6.64
5.19 -1.60 -4.75 -17.84 9.25 -3.97
PALJ810115 0.91 0.77 1.32 0.90 0.50 1.06 0.53 1.61 1.08 0.36 0.77 1.27
0.76 0.37 1.62 1.34
0.87 1.10 1.24 0.52
PONP800106: 10.67 11.05 10.85 10.21 14.15 11.71 11.71 10.95 12.07
12.95 13.07 9.93 15.00
13.27 10.62 11.18 10.53 11.41 11.52 13.86
QIAN880116: -0.19 0.03 0.02 -0.06 -0.29 0.02 -0.10 0.19 -0.16 -0.08
-0.42 -0.09 -0.38 -0.32
0.05 0.25 0.22 -0.19 0.05 -0.15
RICJ880107: 1.1 1.5 0.0 0.3 1.1 1.3 0.5 0.4 1.5 1.1 2.6 0.8 1.7 1.9
0.1 0.4 0.5 3.1 0.6 1.5
ROBB760111: -3.7 1.0 -0.6 -0.6 4.0 3.4 -4.3 5.9 -0.8 -0.5 -2.8 1.3
-1.6 1.6 -6.0 1.5 1.2 6.5 1.3 -4.6
ROSM880103: 0.4 0.3 0.9 0.8 0.5 0.7 1.3 0.0 1.0 0.4 0.6 0.4 0.3 0.7
0.9 0.4 0.4 0.6 1.2 0.4
VENT840101: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 1.00
0.00 0.00 1.00 0.00
0.00 0.00 1.00 1.00 1.00
AURR980101: 0.94 1.15 0.79 1.19 0.60 0.94 1.41 1.18 1.15 1.07 0.95
1.03 0.88 1.06 1.18
0.69 0.87 0.91 1.04 0.90
AURR980105: 0.67 0.76 1.28 1.58 0.37 1.05 0.94 0.98 0.83 0.78 0.79
0.84 0.98 0.96 1.12
1.25 1.41 0.94 0.82 0.67
AURR980118: 0.93 0.96 0.82 1.15 0.67 1.02 1.07 1.08 1.40 1.14 1.16
1.27 1.11 1.05 1.01
0.71 0.84 1.06 1.15 0.74
ZH0H040103: 13.4 8.5 7.6 8.2 22.6 8.5 7.3 7.0 11.3 20.3 20.8 6.1 15.7
23.9 9.9 8.2 10.3
24.5 19.5 19.5