Universidade do Minho
Escola de Engenharia

Rita Sofia Conde Lopes

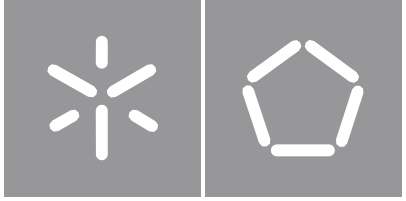
Desenvolvimento e validação de uma ferramenta para identificação de transportomas em genomas sequenciados

março de 2022

Desenvolvimento e validação de uma ferramenta para identificação de transportomas em genomas sequenciados

Rita Lopes

UMinho | 2022



Universidade do Minho
Escola de Engenharia

Rita Sofia Conde Lopes

Desenvolvimento e validação de uma ferramenta para identificação de transportomas em genomas sequenciados

Dissertação de Mestrado
Mestrado em Bioinformática

Trabalho efetuado sob a orientação do
Pedro Alexandre Dias Soares
Isabel João Soares da Silva

março de 2022

AGRADECIMENTOS

O que confere à espécie humana características tão peculiares e, de certo modo, tão fascinantes, é a sua inigualável capacidade de raciocinar e de se organizar em comunidade, possibilitando a realização de feitos, por vezes, inimagináveis e a contínua conquista e inovação. Apesar de cada um de nós ser único, todos temos algo em comum, que é o facto de necessitarmos uns dos outros para atingirmos os nossos objetivos e a felicidade plena.

Tal como num organismo vivo, cada porção de informação genética, DNA, é essencial para o tornar um ser único e especial. No entanto, para que o organismo seja funcional e coeso, e assim ser capaz de alcançar aquilo para que está predestinado, é necessário que todas as porções de informação genética sejam interdependentes. Assim, a biodiversidade, onde cada organismo desempenha o seu papel, torna o planeta Terra um lugar único no Universo.

Deste modo, começo por agradecer ao Doutor Pedro Soares (orientador) e à Doutora Isabel João Silva (coorientadora), pela constante disponibilidade em partilhar conhecimentos e ajuda, essenciais à concretização deste trabalho. Agradeço com especial carinho, também, à Professora Margarida Casal, toda a compreensão e dedicação, bem como o conhecimento e as vivências que partilhou comigo, fazendo de mim uma pessoa muito mais rica e conhecedora.

De seguida, agradeço, inquestionavelmente, aos meus pais e irmã pelo amor, carinho e apoio incondicional, indispensáveis para finalizar desta etapa tão difícil e exigente da minha vida. Eles ajudaram-me a manter o foco, e a não desistir perante as adversidades e as ”pedras”que que foram surgindo no sinuoso ”caminho”.

Também, um agradecimento muitíssimo especial ao Eng. João Marcelo pelo seu constante e incansável apoio psicológico e técnico na realização deste trabalho e do curso, mesmo nos momentos em que a sua atividade profissinal exigia maior presença dele. Com o seu apoio, foi possível demonstrar que tudo é possível, ainda que, por vezes, pareça o contrário, desde que haja empenho, dedicação e interajuda. Sem ele, a finalização desta etapa teria sido muito mais difícil.

Sem esquecer, a ajuda incansável da Ana Marta Sequeira, que mesmo sem usufruir de qualquer tipo de benefício, mostrou-se sempre disponível para ajudar no desenvolvimento do projeto, facultando-me todos os materiais e informações necessárias à sua concretização.

Finalmente, mas não menos importante, agradeço aos meus amigos e colegas de curso, Alexandre Carvalho e Sofia de Beir, que com pequenos gestos, palavras de incentivo e, acima de tudo, compreensão, me ajudaram a levar o curso com mais alegria e leveza. A todos, devo também o meu sucesso neste mestrado e à sua finalização.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

RESUMO

Por definição, o genoma de um indivíduo é todo o seu ácido desoxirribonucléico (DNA), podendo-se inferir o proteoma a partir do mesmo, uma vez que corresponde à porção de DNA que dá origem ao ácido ribonucleico mensageiro (mRNA). Sabe-se que, em todos os genomas analisados, cerca de 30% do DNA que é transcrito para mRNA codifica para proteínas tranportadoras - transportoma. Este, por sua vez, refere-se ao conjunto das proteínas transportadoras de membrana, que apresentam um papel fundamental a nível biológico, tal como transporte de fármacos, e que constituem importantes alvos terapêuticos.

Atualmente existe a necessidade de criar ferramentas automáticas que a partir de um proteoma completo permitam inferir quais prováveis proteínas transportadoras membranares. Devido à falta de ferramentas bioinformáticas integradas que auxiliem o estudos dos transportadores, surge a necessidade de criar novas ferramentas, como as apresentadas neste projeto.

Para desenvolver estas tecnologias, usou-se por base três diferentes metodologias: i) plataformas disponíveis *online*, TMHMM, *Pred-TMBB*, *Prosite* e CDD, acedidas por meio de API's e bibliotecas - *TransPredict*; ii) bibliotecas e *ferramentas online* usadas para gerar *features* para criar os *datasets* e os modelos de *Machine Learning*; iii) modelos de *Deep Learning*.

Ao longo das etapas de desenvolvimento das diversas metdologias foram extraídas matrizes de confusão de todas as abordagens, de modo a simplicar a análise dos resultados obtidos. No caso dos modelos de ML e DL, fez-se, ainda, a avaliação dos modelos com os dados de teste, antes de serem aplicados nos genomas da *Escherichia coli* e da *Saccharomyces cerevisiae*, bem como uma análise exploratória para caracterizar a composição do *dataset* utilizado para treinar os modelos, principalmente de ML.

De todas as abordagens testadas, as que obteveram melhores resultados foi a ferramenta *TransPredict* e os modelos de DL, com resultados próximos.Verificou-se que os modelos de ML ficaram áquem do esperado.

Apesar de se ter detetado algumas limitações e havendo melhorias a implementar, a tarefa foi terminada com sucesso, e este projeto tem potencial para ser mais explorado e desenvolvido, uma vez que constitui um marco importante na investigação dos transportadores, nas diversas áreas de aplicação dos mesmos.

Palavras-chave: *Deep Learning*, Genoma, *Machine Learning*, Transportadores, *Transpredict*.

ABSTRACT

By definition, the genome of an individual it's its entire deoxyribonucleic acid (DNA), and the proteome can be inferred from it, since it corresponds to the portion of DNA that gives rise to messenger ribonucleic acid (mRNA). It is known that, in all the analyzed genomes, about 30% of the DNA that is transcribed to mRNA encodes a transporter protein - transportome. This, in turn, refers to the set of membrane transport proteins, which play a fundamental role at the biological level, such as drug transport, and which constitute important therapeutic targets.

Currently there is a need to create automatic tools that, from a complete proteome, allow inferring which likely membrane transport proteins are. These integrated bioinformatics tools will allow the study of transporter proteins at a large scale.

To develop these technologies, three different methodologies used in this work: i) available online platforms, TMHMM, Pred-TMBB, Prosite and CDD, accessed through API's and libraries - TransPredict; ii) libraries and online tools used to generate features to create datasets and Machine Learning models; iii) Deep Learning models.

Throughout the development stages of the various methodologies, confusion matrices were extracted from all approaches, in order to simplify the analysis of the results obtained. In the case of the ML and DL models, the models were also evaluated with the test data, before being applied to the genomes of *Escherichia coli* and *Saccharomyces cerevisiae*, as well as an exploratory analysis to characterize the composition of the dataset used to train the models, mainly ML.

Of all the approaches tested, the tool TransPredict and the DL models presented the best results, with similar results. It was found that the ML models were below expectations.

Although some limitations were detected and are still improvements to be implemented, the task was successfully completed. Nevertheless this project has the potential to be further explored and developed, since it constitutes an important milestone in the investigation of transporters, in the different areas of application of the same.

Keywords: Deep Learning, Genome, Machine Learning, Transporters, Transpredict.

CONTEÚDO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÓNIMOS

DNA Ácido desoxirribonucleico
RNA Ácido ribonucleico
mRNA Ácido ribonucleico mensageiro
CDD *Conserved Domain Database*
BLAST *Basic Local Alignment Search Tool*
ML *Machine Learning*
DL *Deep Learning*
IMP Proteínas de membrana integrais
PTM Modificações pós-transcricionais
ATP Adenosina trifosfato
NCBI *National Center for Biotechnology Information*
TCDB *Transporter Classification Database*
TC *Transport classification*
DR *Databank Reference*
EMBL *European Molecular Biology Laboratory*
BOMP *β-barrel Outer Membrane Protein Predictor*
SVM *Support Vector Machine*
Psi-BLAST *Position-Specific Iterative Basic Local Alignment Search Tool*
MSE *Mean squared error* **NA** *Missing values*
AI Inteligência artificial
RF *Random Forest*
GB *Gradient Boosting*
LR *Logistic regresison*
KNN *k-nearest neighbors*
GNB *Gaussian Naive Bayes*
ANN *Artificial Neural Networks*
VP Verdadeiro positivo
FP Falso positivo
FN Falso negativo
VN Verdadeiro negativo
ROC *Receiver Operating Characteristic*
AUC Área sob a curva
CCM Coeficiente de correlação de *Matthews*

DNN *Deep Neural Networks*
CNN *Convulotinal Neural Networks*
RNN *Recurrent Neural Networks*
AE Arquiteturas Emergentes
FNN *Feedforward Neural Network*
LSTM *Long short-term memory*
BRNN *Birectional Recurrent Neural Networks*
API *Application Programming Interface*
HTTP *Hypertext Transfer Protocol*
TMH *Transmembrane Helices*
FR *Final result*
URL *Uniform Resource Locator*
PDB *Protein Data Bank*

1

INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

Por definição, o genoma de um indivíduo é todo o conjunto do ácido desoxirribonucléico (DNA) do organismo, podendo-se inferir o proteoma a partir do mesmo, uma vez que corresponde à porção de DNA que dá origem ao ácido ribonucleico mensageiro (mRNA) e que, por sua vez, origina uma proteína. Desse proteoma inferido obtém-se o "transcriptoma mRNA". Sabe-se que, em todos os genomas analisados, cerca de 30% do DNA que é transcrito para mRNA codifica para proteínas tranportadoras - transportoma. Este, por sua vez, refere-se ao conjunto das proteínas transportadoras de membrana, que apresentam um papel fundamental a nível biológico, tal como transporte de fármacos, e que constituem importantes alvos terapêuticos [1]. Além da importância dos transportadores no ramo da indústria farmacêutica, estes são, também, muito relevantes quando falamos da biotecnologia e da produção de biocompostos com elevados rendimentos. Estes requerem a otimização de diversos processos, incluindo transporte de solutos através da membrana para melhorar a entrada de substratos na célula, troca de produtos nos organelos celulares e efluxo de metabolitos para o meio extracelular, metabolitos esses com elevado interesse para o ser humano [2]. Neste campo, as proteínas transportadoras de membrana são elementos cruciais na otimização de importação de substratos e exportação do produto final, levando ao aumento do produto final no meio extracelular e níveis de produtividade mais elevados [3].

Tal como a genómica, a proteómica começou a ser aprofundada apenas há algumas décadas. Assim, esta é atualmente estudada nas mais variadas áreas, como em universidades para fins científicos, institutos governamentais, empresas de biotecnologia e empresas farmacêuticas. Isto porque o estudo da mesma apresenta inúmeras aplicações, tais como descoberta de novos fármacos, diagnóstico e monitorização de doenças [1].

Apesar da importância dos transportomas celulares (também relatado como o segundo maior componente do proteoma da membrana, no caso dos humanos), os transportadores são surpreendentemente pouco estudados [4]. Até ao momento, a classificação de transportadores com base em diferentes famílias/subfamílias, bem como os seus substratos específicos, continua a ser um desafio importante na biologia estrutural e funcional.

Assim, pelo facto deste campo da biologia estar a ser pouco estudado e explorado, surge a necessidade de aprofundar mais esta temática, quer criando novas ferramentas bioinformáticas quer recorrendo às já existentes.

Atualmente, não existem ferramentas bioinformáticas robustas e capazes de efetuar um sistema de identificação de possíveis transportadores tendo como base de estudo um proteoma. No entanto, existem inúmeras ferramentas ao nível da estrutura, tais como: o TMHMM prevê a existência de α-hélices presentes em proteínas de membrana transmembranares [5] e o *Pred-TMBB* prevê segmentos transmembranares e topologias de proteínas da membrana externa, tipo β-folha [6]. Com o *Prosite* detetam-se assinaturas biologicamente significativas, descritas como padrões ou perfis [7]. Com o auxílio do *Conserved Domain Database* (CDD), detetam-se padrões de domínios conservados [8]. As ferramentas TMHMM [5] e *Pred-TMBB* [6], CDD [8] e *Prosite* [7], *Basic Local Alignment Search Tool* (BLAST) [9] e ferramentas de criação de árvores filogenéticas ([10] e [11]), disponíveis *online* inferem, isoladamente, informação importante para a melhor compreensão do proteoma.

Dos primeiros estudos desenvolvidos no ramo da bioinformática, na área da proteómica, foi a classificação e atribuição de proteínas de transporte a uma classe de transportadores em particular, com base no alinhamento de múltiplas sequências. Recentemente, vários métodos baseados em técnicas de *Machine Learning* (ML) foram desenvolvidos, para fazer uma discriminação funcional de proteínas de membrana, prever família de transportadores a partir da sequência de proteínas ou, até mesmo, realizar uma previsão automática de transportadores e uma categorização de sequências de proteínas [12], [13]. De entre eles, pode-se referir, por exemplo, *Gromiha et al.* que analisaram a composição de aminoácidos de proteínas de transporte e desenvolveram modelos baseados em redes neuronais para classificar essas proteínas de transporte como proteínas de canal/poro, transportadores eletroquímicos e transportadores ativos [14]. *Li et al.* desenvolveram uma abordagem baseada em ML geral que integrou um conjunto de regras, baseadas nas características da sequência do transportador obtidas a partir de proteomas curados como referência. Este estudo cobriu as principais famílias/subfamílias de transportadores definidas no banco de dados de classificação de transportadores [15].

1.2 Motivação e Objetivos

Apesar da importância dos transportomas celulares (também relatado como o segundo maior componente do proteoma da membrana, no caso dos humanos), os transportadores são surpreendentemente pouco estudados. Assim, pelo facto deste campo da biologia ser pouco estudado e explorado, surge a necessidade de aprofundar mais esta temática, quer criando novas ferramentas bioinformáticas, quer recorrendo às já existentes.

O principal objetivo do presente trabalho é desenvolver uma ferramenta capaz de realizar um sistema de identificação de possíveis transportadores, dado um proteoma totalmente submetido ou proteoma inferido. Esta classificação é obtida pela utilização das três seguintes metodologias: inferência da estrutura secundária, homologia geral com transportadores bem caracterizados nas bases de dados e deteção de motivos conservados associados a proteínas transportadoras. Adicionalmente, proceder-se-á à criação de metolodologias de ML e *Deep Learning* (DL), de modo a ser possível identificar, também, os transportadores a partir de um genoma. Finalmente, o objetivo passa por validar a ferramenta *TransPredict* e, também, por comparar as diversas metodologias de modo a inferir qual a mais precisa e sensível à deteção de transportadores.

1.3 Estrutura da Dissertação

Para além deste capítulo introdutório, onde é feita uma breve contextualização da temática e são apresentados os principais objetivos da realização deste trabalho, este documento contém mais quatro capítulos essenciais à reflexão do trabalho desenvolvido, face aos objetivos propostos anteriormente.

No **Capítulo 2** é apresentado o estado da arte, isto é, são abordados os principais desenvolvimentos nas áreas presentes no domínio desta dissertação, expondo as principais ideias e trabalhos já realizados na área em estudo. Neste ponto, irão ser abordados os seguintes temas: conceitos gerais das proteínas, proteínas de membrana integrais (IMP), os tipos de transportadores, os *softwares*/plataformas relevantes no desenvolvimento das ferramentas, os modelos biológios usados e, finalmente, conceitos e modelos de ML e DL.

No **Capítulo 3** são apresentadas, de um modo holístico, as metodologias utilizadas para o desenvolvimento e melhoria da ferramenta bioinformática *TransPredict*, bem como as metodologias utilizadas para o desenvolvimento dos modelos de ML e DL.

No **Capítulo 4**, são apresentados os resultados obtidos com as diferentes metedologias referidas no Capítulo 3. Neste ponto, será feita, ainda, uma discussão dos resultados obtidos, fazendo ilações dos mesmos.

Finalmente, no **Capítulo 5**, são apresentadas as conclusões finais do projeto desenvolvido, bem como o trabalho futuro que ainda é possível fazer, no sentido de o melhorar e enriquecer.

2

ESTADO DA ARTE

No presente capítulo serão apresentados alguns dos principais temas e estudos relacionados com as temáticas abordadas na dissertação. Assim, foram identificados sete principais temas que permitem contextualizar o trabalho desenvolvido. Num instante inicial, serão apresentadas as temáticas mais biológicas, como a caracterização geral das proteínas, que posteriormente facilitará a compreensão dos resultados, bem como a distinção dos diversos tipos de transportadores. De seguida, serão apresentados os modelos biológicos usados para testar as ferramentas desenvolvidas, bem como as *tools*, *softwares*/bibliotecas e bases de dados utilizadas. Finalmente, serão abordados os conceitos e metodologias de ML e DL, que estão inerentes ao desenvolvimento dos métodos computacionais, que permitirão a identificação dos transportadores, a partir de genomas/proteomas.

2.1 PROTEÍNAS: CONCEITOS GERAIS

As proteínas são macromoléculas complexas que desempenham papéis fundamentais nos organismos vivos. Estas desempenham numerosas funções nas células, sendo necessárias à estrutura, atividade e regulação [16].

As proteínas são extensos polímeros constituídos por aminoácidos, ligados entre si por ligações peptídicas que se estabelecem entre o grupo carboxilo de um aminoácido e o grupo amina de outro (Figura 2.1). Existem diferentes tipos de aminoácidos, que combinados podem formar diferentes proteínas [16]. O tipo de ligações que se estabelecem e a sequência de aminoácidos determina a estrutura tridimensional da proteína, que por sua vez, determina a sua função específica. Relativamente à polaridade, os aminoácidos podem ser polares (cadeia lateral polar) ou apolares (cadeia lateral hidrofóbica), em que os polares podem, ainda, subdividir-se em polares com e sem carga (positiva ou negativa) [17].

Dependendo do grupo prostético presente nas proteínas, estas são classificadas como glicoproteínas, cromoproteínas, fosfoproteínas, nucleoproteínas ou lipoproteínas [17]. Dado que as proteínas constituem a maior fração da matéria viva e intervêm em inúmeros processos biológicos, tornam-se uma família versátil apresentando diversas funções: catálise, controlo, estrutura, contração, proteção, regulação genética, recetores membranares e transporte [17].

Atualmente, sabe-se que o transcriptoma representa apenas uma pequena percentagem do material genético (estimada em menos de 5% do genoma no caso dos humanos) [18].

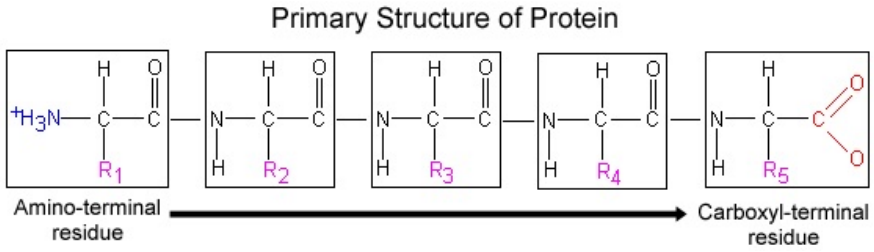


Figura 2.1: Estrutura primária das proteínas (adaptado de [19])

2.1.1 *Estrutura das proteínas*

A nível da organização estrutural, tem-se em conta os seguintes conceitos: estrutura **primária**, **secundária**, **terciária** e **quaternária**. Assim, segundo o comitê de nomenclatura da União Internacional de Bioquímica, temos [4]:

Estrutura primária

Determinada pela sequência linear de aminoácidos, sem ter em conta o arranjo da cadeia peptídica no espaço [16]. Aqui os aminoácidos, encontram-se ligados covalentemente (ligação peptídica) [17].

Estrutura secundária

Determinada pelo arranjo espacial da cadeia peptídica principal, sem ter em consideração a conformação das cadeias laterais ou outros segmentos da cadeia principal [4]. Esta resulta dos impedimentos estéricos da ligação peptídica e dos resíduos de aminoácidos das cadeias laterais [17]. Assim, existem 2 tipos de estruturas secundárias:

- α**-hélice** definem-se entre pequenas zonas da cadeia entre aminoácidos adjacentes. É um dos elementos da estrutura secundária mais abundante, talvez devido à facilidade de o arranjo maximizar o número de pontes de hidrogénio entre os grupos carboxilo e amina, originando uma estrutura rígida, com forma cilíndrica [17].
- β**-folha** (conformações estendidas) onde se estabelecem pontes de hidrogénio entre diferentes cadeias peptídicas. Estas são constituídas por 2 ou mais cadeias β, conformação esta estabilizada por pontes de hidrogénio. Dependendo da orientação das cadeias β, estas formam uma folha β paralela, antiparalela ou mista [17]. Este é o último nível de organização das proteínas fibrosas.

Estrutura terciária

Determinada pelas cadeias laterais e outros segmentos adjacentes da cadeia principal, sem considerar as cadeias peptídicas vizinhas. As cadeias laterais carregadas positiva e negativamente tendem a atrair-se; cadeias laterais com cargas idênticas repelem-se [4]. Uma vez que na estrutura terciária as interações são de longa distância, esta resulta no enrolamento da proteína no espaço. Neste nível, existem fatores que estabilizam a estrutura, tais como: interações hidrofóbicas, ligações iónicas, forças *Van der Waals*, pontes de hidrogénio e ligações dissulfureto [17].

Estrutura quaternária

Arranjo de subunidades idênticas ou diferentes de uma proteína grande, na qual cada subunidade é uma cadeia peptídica separada [4]. As proteínas oligoméricas podem associar-se em conjuntos de 2, 3, 4, 5, 6 ou mais subunidades, designando-se, respetivamente, por dímeros, trímeros, tetrâmeros, pentâmeros, hexâmeros, e assim sucessivamente [17]. Nestas proteínas, as subunidades são ligadas entre si por pontes dissulfureto [4].

2.2 Proteínas de membrana integrais

As IMP funcionam como importantes transmissores, canais, recetores e enzimas, responsáveis pela transdução de sinal, processos regulatórios e interações célula-célula e célula-ambiente. Assim, as IMP tornam-se alvos interessantes para intervenções terapêuticas [20]. Estas são definidas, empiricamente, como proteínas que não podem ser extraídas da bicamada fosfolipídica sem destabilização da mesma [21]. Podem ser classificados como: **monotópicas** que se ligam fortemente à membrana, mas não a abrangem; **bitópicas**, que abrangem a membrana uma vez; e **politópicas**, que atravessam a membrana várias vezes [21].

2.3 Tipos de transportadores

O transporte de solutos, como nutrientes, iões ou resíduos metabólicos, através da membrana celular é de extrema importância em todos os sistemas vivos, pelo que até 14% do genoma de todos os organismos representam informação para proteínas de transporte. Quase todos os processos de transporte transmembranar são conduzidos por proteínas integrais da membrana, por vezes associado a proteínas recetoras extracitoplasmáticas e/ou reguladoras e acopladas a consumo de energia [22]. Assim, o transporte através da membrana, ocorre através de proteínas transportadoras, cuja regulação é química ou elétrica. Os transportadores permitem a passagem de moléculas por um processo passivo (transporte passivo e difusão facilitada) sem gasto de energia ou ativo (transporte ativo) com gasto de energia

[17]. Relativamente ao transporte ativo, pode-se ainda distinguir entre transporte ativo primário, em que a proteína envolvida no transporte utiliza energia obtida a partir de uma reação química exoenergética, sendo diretamente dependente de adenosina trifosfato (ATP), e transporte ativo secundário, onde o transporte depende indiretamente do ATP, resultante do gradiente de concentração iónica estabelecida pelo transporte ativo primário [17].

As proteínas de transporte dividem-se em 3 grandes categorias: permeases, canais e bombas (Figura 2.2) [23].

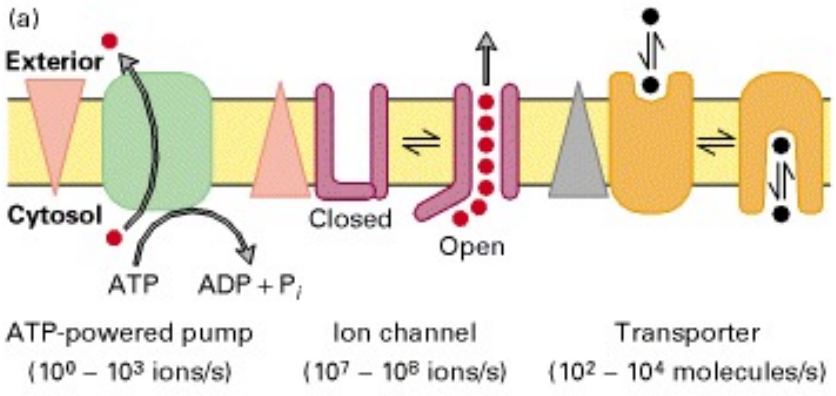


Figura 2.2: Tipos de proteínas de transporte: bomba (à esquerda), canais (ao centro), permeases (à direita) (adaptado de [23])

As **permeases** [23], medeiam a difusão facilitada de solutos e metabolitos [22]. Estas ligam-se ao soluto específico a ser transportado, passando por uma série de alterações conformacionais de modo a transferir o mesmo [23].

Os **canais**, ao contrário das anteriores, formam poros aquosos constituídos por resíduos específicos da proteína constituinte [22] que se estendem pela bicamada fosfolipídica; quando esses poros estão abertos, permitem a passagem de solutos específicos (geralmente iões inorgânicos com tamanho e carga adequados) pela membrana [23]. No entanto, outros substratos hidrofílicos e hidrofóbicos podem ser transportados por esses sistemas de transporte [22]. Assim, é expectável que o transporte através de canais ocorra a uma velocidade maior que o transporte mediado pelas permeases [23].

As **bombas** dependentes de ATP (ou simplesmente bombas) são ATPases que usam a energia da hidrólise de ATP para mover iões ou pequenas moléculas, através da membrana contra um gradiente de concentração química ou potencial elétrico [23]. Esse processo, transporte ativo, é um exemplo de reação química acoplada. A reação geral - hidrólise de ATP e o movimento "ascendente"de iões ou pequenas moléculas - é energeticamente favorável. Ao contrário dos canais iónicos, as bombas de iões transportam ativamente iões contra um gradiente de concentração, enquanto os canais iónicos permitem que estes fluam passivamente a favor do gradiente de concentração [23].

2.4 MODELOS BIOLÓGICOS

A anotação do genoma de um organismo envolve a identificação de genes, os limites dos genes em termos de locais de início e término precisos e a descrição dos produtos génicos. As funções conhecidas e previstas são atribuídas a cada produto génico com base em evidências experimentais ou análise de sequências. Como os dois tipos de evidência estão em constante expansão, nenhuma anotação é completa em nenhum momento [24].

Um genoma pode variar significativamente entre estirpes e, portanto, o genoma de referência serve como âncora para explorar a diversidade de alelos e complementos génicos e para explorar o modo como essas diferenças contribuem para a variação metabólica e fenotípica [25].

2.4.1 *Escherichia coli - estirpe K-12, sub-estirpe MG1655*

Classificação taxonómica: Bacteria; Proteobacteria; Gammaproteobacteria; Enterobacterales; Enterobacteriaceae; *Escherichia*; *Escherichia coli*; *Escherichia coli* str. K-12 substr. MG1655 [26]

Devido à sua posição extraordinária como um modelo preferido na genética, bioquímica, biologia molecular e biotecnologia, a bactéria *E. coli* K-12 foi o primeiro organismo a ser sugerido como um candidato à sequenciação do genoma completo. A disponibilidade da sequência completa da *E. coli* estimulou o desenvolvimento de vários estudos, tendo em vista o conhecimento mais completo deste importante organismo, quer ao nível biológico, clínico e industrial. Desde o início do projeto de sequenciação do genoma da *E. coli*, seis outros genomas completos foram publicados. As sequências do genoma, especialmente aquelas de organismos experimentais bem estudados, ajudam a integrar um vasto recurso de conhecimento biológico e servem como um guia para futuras experiências. A disponibilidade do conjunto completo de genes também permite abordagens globais da função biológica em células vivas, levando a novas formas de entender a história evolutiva das bactérias [27].

A estirpe K-12 da *E. coli* é, indiscutivelmente, um dos organismos mais estudados. Originalmente isolado em 1922, foi descoberta a capacidade da estirpe K-12 de realizar recombinação génica por conjugação e, logo depois, por transdução generalizada. A estirpe K-12 foi amplamente distribuída por inúmeros laboratórios em todo o mundo [24].

2.4.2 *Saccharomyces cerevisiae - S288C*

Classificação taxonómica: Eukaryota; Fungi; Dikarya; Ascomycota; Saccharomycotina; Saccharomycetes; Saccharomycetales; Saccharomycetaceae; *Saccharomyces*; *Saccharomyces cerevisiae*; *Saccharomyces cerevisiae* S288C [28]

A levedura *S. cerevisiae*, um dos eucariontes unicelulares mais intensamente estudados, tem sido comumente usado como organismo modelo para descobrir processos celulares semelhantes e funções específicas de proteínas de outros organismos. Muitas vias funcionais importantes, como o metabolismo lipídico e o ciclo celular, foram identificadas como processos celulares semelhantes entre leveduras e humanos. Através do desenvolvimento histórico da biologia de sistemas em leveduras, a *S. cerevisiae* tem sido amplamente estudada, desde componentes de sistemas individuais até complexas interações, a fim de decifrar o quadro completo dos processos celulares [29].

Atualmente, a *S. cerevisiae* é também amplamente usada em estudos de engenharia metabólica e biologia de sistemas, tanto ao nível industrial como académico [30]. A sua utilização e importância nas fermentações tradicionais de alimentos não têm comparação com outros organismos, ao nível da relevância biotecnológica. Mais recentemente, a *S. cerevisiae* tem vindo a ser aplicada, por exemplo, na produção de biocombustíveis a partir de materiais celulósicos e potencialmente outros substratos, tendo um grande impacto na geração de novas fontes de energia [31].

A *S. cerevisiae* S288C é uma estirpe haplóide com mutação no gene *GAL2*, obtida em laboratório e com uma longa história de uso em estudos genéticos e de biologia molecular. Esta estirpe tem uma genealogia complexa, uma vez que foi produzida por numerosos cruzamentos premeditados, primeiro por Carl Lindegren e, anos depois, por Robert Mortimer [25].

2.5 Plataformas/*softwares* relevantes

2.5.1 *TMHMM*

O TMHMM permite prever a topologia de proteínas de membrana, com base no modelo oculto de *Markov*. O TMHMM prevê corretamente 97-98% das hélices transmembranares, sendo capaz de distinguir as proteínas solúveis das proteínas da membrana, com especificidade e sensibilidade superiores a 99%. Essa precisão pode diminuir na presença de péptidos de sinal. O elevado grau de precisão destes modelos, permitem prever, com elevado grau de confiança, proteínas de membrana a partir de uma grande coleção de genomas [5].

2.5.2 *Pred-TMBB*

O *Pred-TMBB* prevê filamentos transmembranares e topologias das proteínas da membrana externa, tipo β-folha. Tal como o TMHMM, esta plataforma usa como método base o modelo oculto de *Markov*, um modelo probabilístico composto por vários estados conectados

por meio das probabilidades de transição, treinado de acordo com o critério de máxima verossimilhança condicional [6].

Com esta plataforma é possível obter: a topologia prevista de uma determinada proteína; a pontuação indicativa da probabilidade da proteína ser uma proteína com uma estrutura em β-folha da membrana externa; probabilidades posteriores para a previsão da fita transmembranar; e uma representação gráfica da posição assumida das fitas transmembranares em relação à bicamada fosfolipídica [6]

2.5.3 *Prosite*

O banco de dados do *Prosite* é uma coleção de assinaturas biologicamente significativas, descritas como padrões ou perfis. Cada assinatura está vinculada a uma documentação que fornece informações biológicas úteis da família das proteínas, domínio ou local funcional identificado pela assinatura [7].

O *Prosite* usa 2 tipos de assinaturas para identificar regiões conservadas, ou seja, padrões e perfis generalizados. Cada assinatura está vinculada a um documento de anotação onde o utilizador pode encontrar informações sobre a família ou domínio de proteínas detetadas pela assinatura: origem do nome, ocorrência taxonómica, arquitetura do domínio, função, estrutura 3D, principais características da sequência, tamanho do domínio e referências [7].

Os padrões ou expressões regulares são ferramentas úteis para identificar regiões curtas e conservadas, como locais catalíticos, locais de ligação, modificações pós-transcricionais (PTMs) ou *zinc fingers*, sendo intuitivos para o utilizador comum. No entanto, os padrões podem levar a falsos positivos/negativos pois se uma nova sequência contiver um aminoácido numa posição conservada que não estava presente no alinhamento inicial na construção do padrão, esta pode não ser reconhecida. Assim, os padrões são atualizados regularmente, para introduzir novas variabilidades [7].

2.5.4 *Conserved Domain Database - CDD*

O CDD do *National Center for Biotechnology Information* (NCBI) é um recurso público de anotação de proteínas que deteta assinaturas de domínios conservados. A plataforma permite ao utilizador comparar as sequências em estudo com as arquivadas que detêm anotações de domínio pré-calculada para sequências rastreadas no banco de dados de proteínas do *Entrez* do NCBI. Esta apresenta informação curada, visando aumentar a cobertura e fornecer classificações mais fidedignas dos domínios proteicos, enriquecendo os dados funcionais e estruturais. A curação dos dados disponíveis no CDD permite gerar modelos de alinhamento de fragmentos de sequência representativos, de acordo com os

limites do domínio e modelam os núcleos estruturalmente conservados das famílias de domínios, além de anotar os domínios conservados [8].

2.5.5 ***Transporter Classification Database*** - *TCDB*

O TCDB é um banco de dados relacional e curado que contém informações das sequências, classificação, informações estruturais, funcionais e evolutivas sobre sistemas de transporte de vários organismos. Este repositório contém informações compiladas a partir de >10000 referências, abrangendo, aproximadamente, 3000 transportadores representativos e transportadores putativos, classificados em >400 famílias . O sistema de classificação de transportadores é um sistema de nomenclatura aprovado pela União Internacional de Bioquímica e Biologia Molecular para classificação de proteínas de transporte [32].

O TCDB fornece vários métodos para aceder aos dados: acesso passo a passo à classificação hierárquica, pesquisa direta por sequência ou número do transportador e pesquisa de texto completo. A ontologia funcional subjacente à estrutura de dados facilita pesquisas poderosas de consulta que produzem dados de uma forma rápida e fácil. Além de fornecer informações curadas e apresentar uma ferramenta de classificação de proteínas de membrana recentemente identificadas, o TCDB permite, também, fazer anotações de transportadores codificados no genoma [32].

O sistema *Transport classification* (TC) é dividido em 9 classes de famílias de transportadores, sendo que cada uma dessas classes é subdividida em subclasses, conforme representado na Tabela 2.1 [33]. As primeiras 5 classes são classes bem definidas. Assim, quando existem informações suficientes disponíveis, é possível atribuir um número TC ao transportador em questão [32].

Tabela 2.1: Sistema TC hierárquico de classificação de famílias de transportadores

Classe	**Subclasse**
	1.A: Canais do tipo α 1.B: Porinas β-folha 1.C: Toxinas Formadoras de Poros (Proteínas e Péptidos) 1.D: Canais Sintetizados Não Ribossomicamente 1.E: Holinas 1.F: Poros de fusão de vesículas 1.G: Poros de fusão viral 1.H: Canais Paracelulares 1.I: Canais delimitados por membrana

Tabela 2.1

Classe	Subclasse
1. Canais / poros	1.J: Aberturas piramidais de saída de virião 1.K: Canais de injeção de DNA de fago 1.L: Nanotubos de tunelamento, TNTs 1.M: *Spanins* mediadores de fusão de membrana 1.N: Poros de fusão celular 1.O: Força Física (Sonoporação / Eletroporação / Tensão, etc.) - Poros induzidos 1.P: Complexo de penetração sem invólucro 1.Q: Poros septais fúngicos 1.R: *Site* de contacto de membrana (MCS) para transporte interorganelar 1.S: Poros de proteína de parede de micro / nanocompartimento bacteriano 1.T: Complexos de classificação endossómica necessários ao transporte (ESCRT) 1.U: Peptido permeável à célula (CPP) 1.V: Poros septais filamentosos cianobacterianos 1.W: Subclasse de proteína portal de fago
2. Transportadores movidos a potencial eletroquímico	2.A: Transportadores (uniporte, simporte, antiporte) 2.B: Transportadores sintetizados não ribossomicamente 2.C: Energizadores acionados por gradiente de iões 2.D: Transportador de lípidos transcompartimentais
3. Transportadores ativos primários	3.A: Transportadores acionados por hidrólise de ligação P-P 3.B: Transportadores acionados por descarboxilação 3.C: Transportadores movidos por transferência de grupo metilo 3.D: Transportadores acionados por oxirredução 3.E: Transportadores movidos por absorção de luz
	4.A: Translocadores de grupo conduzidos por fosfotransferência 4.B: Transportadores de captação de ribonucleosídeo de nicotinamida 4.C: Transportadores acoplados a Acil CoA ligase 4.D: Polissacarídeo Sintase / Exportadores

Tabela 2.1

Classe	Subclasse
4. Grupo de Translocadores	4.E: Translocadores de Grupo Vacuolar Polifosfato Catalisado por Polimerase 4.F: Colina / Etanolaminafosfotransferase 1 (CEPT1) 4.G: Protease de membrana integral de libertação de péptidos (translocadores IMP-PR) 4.H: Lisilfosfatidilglicerol Sintase / Flipases
5. Portadores de eletrões transmembranares	5.A: Transportadores de transferência transmembranares, de 2 eletrões 5.B: Transportadores de transferência transmembranares, de 1 eletrão
8. Fatores acessórios envolvidos no transporte	8.A: Proteínas auxiliares de transporte 8.B: Toxinas/agonistas de proteínas/peptídeos sintetizados ribossomalmente que visam canais e transportadores 8.C: Toxinas sintetizadas não ribossomalmente que têm como alvo canais, transportadores e bombas 8.D: Membranas miméticas para solubilizar proteínas integrantes da membrana
9. Sistemas de Transporte Incompletamente Caracterizados	9.A: Transportadores reconhecidos de mecanismo bioquímico desconhecido 9.B: Proteínas de transporte putativas 9.C: Transportadores funcionalmente caracterizados sem sequências identificadas

2.5.6 *Swiss-Prot (UniProt)*

Swiss-Prot é uma base de dados curada de sequências de proteínas que permite fornecer um elevado nível de anotação, com o mínimo nível de redundância possível e um elevado nível de integração noutras bases de dados biológicas, como o *Databank Reference* (DR) *Prosite*, *Pfam* e *European Molecular Biology Laboratory* (EMBL) [34].

Relativamente à anotação no *Swiss-Prot* é possível distinguir duas classes - *core data* e anotações. Por um lado, *core data* refere-se à sequência, informação citada (referências bibliográficas) e à taxonomia, enquanto as anotações consistem na descrição da sequência, onde constam os seguintes itens: função da proteína, PTM, domínios, estrutura secundária

e quaternária, similaridades com outras proteínas, doenças associadas à deficiência na proteínas em questão, variantes, entre outras [34].

Mais recentemente, a *Swiss-Prot* selecionou alguns organismo que são alvo de sequenciação de genoma ou projetos de mapeamento, onde pretendem que seja o mais completo possível, fornecer o maior nível de anotação possível, referências cruzadas com bases de dados especializadas no tema e, finalmente, documentos. De entre os organismos selecionados, estão contemplados a *E. coli*, *S. cerevisiae*, *Arabidopsis thaliana*, *Drosophila melanogaster*, o *Mus musculus* e, até mesmo, o *Homo sapiens* [34].

2.5.7 *Phobius*

O servidor *Phobius* é um meio fácil e preciso que permite prever os péptidos sinal e a topologia das proteínas transmembranares, a partir de uma sequência de aminoácidos. Este é baseado no modelo oculto de *Markov* que modela as diferentes regiões de sequência de um péptido sinal e as diferentes regiões de uma proteína transmembranar, numa série de estados interconectados. O *input* (sequências de aminoácidos) deve estar num formato fasta e todas as previsões feitas pelo *Phobius* podem, opcionalmente, ser acompanhadas por um gráfico de probabilidades [35].

2.5.8 *β-barrel Outer Membrane Protein Predictor - BOMP*

O *β-barrel Outer Membrane Protein Predictor* (BOMP), é baseado em duas componentes distintas para proceder ao reconhecimento de proteínas integrais com estrutura β-folha. A primeiro componente do programa consiste em detetar o padrão C-terminal típico de muitas proteínas integrais em estruturas β-folha. A segunda componente calcula uma pontuação β-folha integral da sequência, com base na extensão em que a sequência contém porções de aminoácidos típicos de proteínas transmembranares do tipo β-folha [36].

2.5.9 *LocTree3*

A previsão da localização subcelular da proteína é uma informação importante para nos permitir inferir acerca da função da mesma. Assim, para cada sequência de proteína dada como *input*, o *LocTree2* aplica um modelo de ML, *Support Vector Machine* (SVM), para prever a localização subcelular da mesma. O método gera um valor que reflete a confiança em cada previsão. No entanto, este *software* foi aperfeiçoado, dando lugar ao *LocTree3* como um servidor *web* público. Além dos fundamentos básicos do *LocTree2*, adicionou-se a inferência baseada na homologia [37].

As principais melhorias são: (i) inclusão de transferência de anotação de homólogos próximos com localização determinada experimentalmente, por meio de *Position-Specific Iterative Basic Local Alignment Search Tool* (PSI-BLAST); (ii) redução do tempo de execução, pela implementação rápida do *kernel* do perfil SVM; (iii) ontologia de genes, anotações para resultados de previsão; (iv) criação de memória *cache*, para guardar resultados para o processamento mais rápido, no caso de pesquisas repetidas [37].

2.5.10 *BioPython*

O *Biopython* é um conjunto de ferramentas disponíveis gratuitamente, desenvolvido por uma equipa internacional de programadores, direcionado para computação biológica, escrito em linguagem *Python*. Este apresenta uma vasta coleção de classes, módulos e funções para análises de sequências biológicas, alinhamentos de sequências, estruturas de proteínas, filogenia, visualização de dados biológicos, deteção de motivos conservados, além de facilitar o acesso a bases de dados biológicas [38]. Os recursos do *Biopython* incluem *parsers* aplicável a vários formatos de arquivo de bioinformática (BLAST, Fasta, Genbank, etc), acesso a serviços *online* (NCBI, *Expasy*, etc), interfaces para programas comuns e não tão comuns, como o *ClustalW*, uma classe de sequência padrão, módulos de agrupamento, estrutura de dados em árvore k-dimensional e, até mesmo, documentação [38].

2.5.11 *Scikit-learn*

Scikit-learn é uma biblioteca de ML de *open source* que fornece suporte à ML supervisionada e não supervisionada [39]. Esta oferece, também, várias ferramentas para ajuste de modelos, pré-processamento de dados, seleção e avaliação de modelos. O *scikit-learn* fornece algoritmos para tarefas de ML, incluindo classificação, regressão, redução de dimensionalidade e armazenamento em *cluster* [40].

Concebido como uma extensão da biblioteca *SciPy*, o *scikit-learn* é baseado numa das mais populares bibliotecas *Python NumPy* e *matplotlib*. *NumPy* permite realizar operações eficientes em grandes *arrays* e matrizes multidimensionais. Já o *matplotlib* fornece ferramentas de visualização, como gráficos e *SciPy* fornece módulos para computação científica [40].

2.5.12 *TensorFlow*

O *TensorFlow* é uma plataforma *open source*, usada no desenvimento de métodos de ML, com um ecossistema vasto e ajustável de ferramentas e bibliotecas, que permite aos utilizadores construir modelos ML inovadores [41].

Geralmente, o *TensorFlow* consiste em duas secções: i) construir um grafo de cálculos (fase de construção) e executar o grafo computacional (fase de execução). Na fase de construção, utilizam-se funções do *TensorFlow* para construir um grafo computacional que representa um modelo de ML. O grafo é composto por arestas e nós. As arestas representam dados na forma de *tensor* (vetor, matriz ou *array* de dados dimensionais mais elevados) que fluirá pelo grafo, dando o nome da biblioteca, *TensorFlow*. Os nós são intitulados de operações que representam cálculos (adição, multiplicação, etc) em tensores. Uma operação leva zero ou mais tensores como entrada e produz zero ou mais tensores de saída. O *TensorFlow* fornece blocos de construção básicos, tais como camadas totalmente conectadas, camadas convolucionais, módulo de rede neuronal recorrente e funções de ativação não linear. Além de tudo, o *TensorFlow* oferece várias funções de perda, como entropia cruzada e erro quadrático médio (MSE) [42].

2.5.13 *ProPythia*

O *ProPythia* consiste num pacote *python* genérico e modular cuja finalidade é permitir a fácil implementação de abordagens de ML e DL, com o intuito de resolver uma infinidade de problemas na análise e classificação de sequências de proteínas . Este *package* facilita a implementação, comparação e validação das principais tarefas em *pipelines* de ML ou DL, incluindo módulos para: ler e alterar sequências, calcular recursos de proteínas, pré-processar conjuntos de dados, executar seleção de recursos e redução de dimensionalidade, realizar agrupamentos e análise de variedades e treinar/otimizar modelos de ML e/ou DL, permintindo utilizá-los para previsões de proteínas.

Este possui uma arquitetura modular adaptável, tornando-se uma ferramenta versátil e fácil de usar, útil para transformar dados de proteínas em conhecimento válido, mesmo para pessoas que não estão familiarizadas com os códigos ML [43].

2.6 CONCEITOS E MODELOS DE *machine learning*

Nos últimos anos, a ML tornou-se uma das mais importantes e crescentes tecnologias de informação e um ramo da inteligência artificial (AI). Atualmente, observa-se uma difusão óbvia e evidente dessas aplicações, com melhoria contínua e criação de novas ferramentas, cada vez mais poderosas.

A ML tem sido aplicada a vários problemas de biologia computacional, onde a geração de dados biológicos e clínicos aumentou exponencialmente, nomeadamente após o aparecimento das tecnologias de sequenciação de última geração. Dado que a maioria das tarefas no campo da análise de sequência de proteínas são tarefas de classificação binária ou

multitarefa, a ML tem sido muito usada na caracterização e identificação de diferentes tipos de péptidos e proteínas, de uma forma eficiente e de baixo custo [44].

Assim, o principal foco da ML é estudar, construir/projetar e melhorar modelos matemáticos que podem ser treinados (uma vez ou continuamente) com dados contextualizados, de modo a inferir o futuro e tomar decisões sem o conhecimento completo de todos os elementos influentes (fatores externos) [45], sem que haja necessidade da intervenção humana, ou pelo menos reduzir o quanto possível a mesma [42].

Deste modo, pode-se dizer que quase todos os algoritmos de ML usados podem ser tratados como um problema de otimização. Neste contexto, é necessário definir/encontrar parâmetros que minimizam uma função objetivo, como uma soma ponderada de dois termos. Normalmente, a função objetivo tem dois componentes: um **regularizador**, que controla a complexidade do modelo, e a **perda**, que mede o erro do modelo nos dados de treino. Por outro lado, o parâmetro de regularização define o *trade-off* entre os dois objetivos de minimizar a perda do erro de treino e a complexidade do modelo, de modo a evitar *overfitting* (2.6.9) [42].

Todo o processo de aprendizagem requer um determinado conjunto de dados [42]:

- **Conjunto de dados de treino:** base de conhecimento usada para ajustar parâmetros do algoritmo de ML. Nesta fase, utiliza-se o conjunto de treino para encontrar os pesos ótimos, com a regra *back-prop*, e todos os parâmetros a serem definidos antes do início do processo de aprendizagem (hiperparâmetros).

- **Conjunto de validação:** conjunto de exemplos usados para ajustar os parâmetros de um modelo de ML. Por exemplo, usa-se o conjunto de validação para encontrar o número ideal de unidades ocultas ou determinar um ponto de paragem para o algoritmo de retropropagação. Alguns *developers* de ML referem-se a este como um *development set* ou *dev set*.

- **Conjunto de dados de teste:** usado para avaliar o desempenho do modelo em dados não vistos - inferência do modelo. Após avaliação do modelo final no conjunto de teste não é necessário proceder a outros ajustes do mesmo.

A teoria da aprendizagem usa ferramentas matemáticas que derivam da teoria da probabilidade e da teoria da informação. Assim, neste sentido, podem-se referir três paradigmas de aprendizagem [42]:

- **Aprendizagem supervisionada:** tarefa de aprendizagem automática mais simples e mais conhecida. Baseia-se numa série de exemplos pré-definidos, nos quais a categoria a que cada um dos insumos deve pertencer já é conhecida. Nesse caso, a questão crucial é o problema da generalização. Após a análise de uma pequena amostra típica de exemplos, o sistema deve produzir um modelo que funcione bem para todas as entradas possíveis.

- **Aprendizagem não supervisionada:** é fornecido um conjunto de entradas ao sistema durante a fase de treino. Em contraste com a aprendizagem supervisionada, os objetos de entrada não são rotulados na sua classe. Esse tipo de modelo é importante porque, no cérebro humano, é provavelmente muito mais comum do que os modelos supervisionados.

- **Aprendizagem por reforço:** abordagem de AI que se concentra na aprendizagem do sistema por meio das interações com o ambiente. Com este tipo de modelos, o sistema é capaz de adaptar os seus parâmetros com base no *feedback* recebido do ambiente, que então fornece *feedback* sobre as decisões tomadas.

As propriedades estruturais e físico-químicas derivadas de sequência de peptidos/-proteínas têm sido usadas no desenvolvimento de modelos de ML para prever, entre outras, classes estruturais e funcionais de proteínas. Este tipo de abordagem, assume que a função de uma proteína pode ser prevista a partir das suas propriedades físico-químicas que podem ser calculadas a partir do conhecimento da sequência do péptido [46]. Depois de obter o subconjunto de descrições mais relevante na previsão da propriedade de interesse, é então possível usar algoritmos de ML, com ou sem supervisão, para avaliar e prever as propriedades de interesse, nas diferentes sequências de proteínas [44].

2.6.1 *Features Selection/Engeneering e Dataset*

De facto, *Features Engeneering* é a primeira etapa no desenvolvimento de um modelo de ML ou DL, envolvendo todas as técnicas adotadas para corrigir os conjuntos de dados, melhorar a relação sinal-ruído e reduzir a dimensionalidade. A maioria dos algoritmos tem suposições fortes sobre os dados de entrada e o seu desempenho pode ser afetado negativamente quando os conjuntos de dados brutos são usados. Por outro lado, os dados raramente são isotrópicos, ou seja, existem frequentemente recursos que determinam o comportamento geral de uma amostra, enquanto outros não fornecem informações adicionais relevantes. Assim, é importante ter uma visão clara do conjunto de dados a usar e conhecer os algoritmos mais comumente usados, de modo a reduzir o número de recursos ou selecionar apenas as melhores *features* [45].

A seleção das *features* é a primeira, e até mesmo, por vezes, a mais importante etapa na *pipeline* de um modelo de ML/DL. Isto porque, nem todas as *features* são úteis para o caso em estudo e, algumas delas, são expressas usando notações diferentes, daí que seja frequente a necessidade de pré-processar o conjunto de dados antes de qualquer outra operação [45].

O desempenho dos métodos baseado em ML dependem, em grande medida, da escolha da representação dos dados, bem como a escolha da representação depende do modelo a utilizar [47]. Em ML, as *features* permitem a um sistema inferir as representações necessárias

para a deteção ou classificação de recursos, a partir de dados brutos [47]. Assim, um dos problemas que se coloca é a forma de selecionar um subconjunto das variáveis originais ou um subconjunto das características originais, sendo que a primeira é chamada de seleção de variável e a seguinte de seleção de recurso/escolha de representação [47].

Relativamente aos *Datasets*, quando um conjunto de dados é suficientemente grande, é boa prática dividi-lo em conjuntos de dados de treino e teste, sendo o primeiro usado para treinar o modelo e o segundo para testar o desempenho. Existem duas principais regras para esta divisão [45]:

- Ambos os conjuntos de dados devem refletir a distribuição original;
- O conjunto de dados original deve ser aleatoriamente misturado antes da fase de divisão, a fim de evitar uma correlação entre os elementos consequentes.

O passo seguinte, fundamental na criação de um *dataset* com qualidade, é a **remoção de *Missing values* (NA)**.Para isso, existem algumas opções que podem ser aplicadas: remoção de toda a linha; criação de um sub-modelo para prever esses valores; estratégia automática, em que coloca valores a partir dos valores já conhecidos, do *dataset* [45]. A primeira opção é a mais drástica, devendo ser considerada apenas quando o conjunto de dados é muito grande, o número de NA's é alto e qualquer previsão pode ser arriscada. A segunda opção torna-se mais difícil de aplicar pelo facto de ser necessário determinar uma estratégia supervisionada para treinar um modelo para cada valor e, finalmente, prever o seu valor. Em suma, avaliando os prós e contras das opções disponíveis, a terceira opção torna-se, provavelmente, a mais viável [45].

Outro ponto importante é a **normalização e escalonamento dos dados**, pois um conjunto de dados genérico é composto por diferentes valores (numéricos ou categóricos), que podem ser extraídos e apresentarem diferentes distribuições, diferentes escalas e, às vezes, *outliers*. Um algoritmo de ML não é capaz de distinguir, naturalmente, essas diversas situações e, portanto, é aconselhável padronizar os conjuntos de dados antes de processá-los. Um dos problemas com maior relevância deriva do facto dos dados apresentarem uma média diferente de 0 e uma variância maior que 1, daí a necessidade de uma normalização [45].

2.6.2 *Random forest (RF)*

Os modelos de ML *Random forest* incluem um conjunto de árvores de decisão e incorpora naturalmente a seleção de recursos e interações, no processo de aprendizagem. É um modelo não paramétrico, interpretável, eficiente e possui alta precisão de previsão para muitos tipos de dados. Trabalhos recentes no ramo da biologia computacional têm aumentado o seu uso, devido às suas vantagens em lidar com dimensões pequenas de amostra, espaço de recursos de alta dimensão e estruturas de dados complexas [48].

É possível aplicar um classificador ***ExtraTreesClassifier***, uma classe que implementa um meta-estimador e que permite ajustar várias árvores de decisão aleatórias (também conhecidas como *extra-trees*), em várias sub-amostras do conjunto de dados e usa a média para melhorar a precisão preditiva e o sobreajuste do controle [49].

2.6.3 *Gradient Boosting (GB)*

Os modelos de *Gradient Boosting* são modelos de regressão aditivos com ajustes sequenciais de uma função parametrizada aos pseudo-resíduos atuais, por mínimos quadrados, a cada iteração. Os pseudo-resíduos representam o gradiente da perda funcional. Nestes modelos pretendem-se minimizar os pseudo-resíduos, em relação aos valores do modelo em cada ponto dos dados de treino, avaliado na atual iteração [50].

2.6.4 *Support vector machine (SVM)*

Algoritmicamente, os modelos SVM permitem criar limites de separação otimizados entre conjuntos de dados, tendo como base a resolução de problemas de otimização quadrática restrita. Usando diferentes funções de *kernel*, podem ser incluídos diferentes graus de não linearidade e flexibilidade [51]. Os modelos SVM são usados, principalmente, para problemas de classificação [52].

2.6.5 *Logistic regression(LR)*

Os modelos de ML *Logistic regression* fornecem um método para modelar uma variável de resposta binária, assumindo valores de 1 e 0. Deste modo, o objetivo passa por determinar o hiperplano ótimo, que separa as duas classes [51].

2.6.6 *K-nearest neighbours (KNN)*

Nos algoritmos do tipo KNN, os dados de treino (que estão bem definidos) são alimentados pelo utilizador. Quando os dados do teste são apresentados, este compara os dois. K dados mais correlacionados são obtidos do conjunto de treino [52].

2.6.7 *Gaussian Naive Bayes (GNB)*

O *Gaussian Naive Bayes* (GNB) é um conjunto de algoritmos de ML supervisionada, que aplica o teorema de *Bayes* com a suposição *naive* de independência entre cada par de *features*,

implementando a classificação e assumindo a distribuição Gaussiana [53]. Um classificador NB calcula a probabilidade de uma determinada instância (exemplo) pertencer a uma dada classe [52]. O GNB é usado, principalmente, para problemas de *clustering* e classificação.

2.6.8 *Artificial Neural Networks (ANN)*

As *Artificial Neural Networks* (ANN) derivam do conceito biológico neurónio, que se pode dividir em 3 principais elementos: dendrites, corpo celular e axónio. As dendrites recebem os sinais elétricos, o corpo celular processa o sinal elétrico e a saída do processo é transportada pelo axónio para os terminais dendríticos, enviando a informação para neurónio seguinte. Uma rede neuronal artificial comporta-se do mesmo modo, funcionando em 3 camadas: camada de entrada que recebe o *input*, a camada oculta processa a informação e a camada de saída envia o *output* determinado. Atualmente, existem 3 tipos de redes neuronais artificiais: supervisionada, não supervisionada e de reforço [52].

As ANN tiram vantagem do conceito de DL, sendo que estas são uma representação abstrata do sistema nervoso humano, que contém uma coleção de neurónios que comunicam entre si, por meio de conexões chamadas axónios [42].

Deste modo, é possível fazer uma analogia entre um neurónio biológico e o neurónio artificial (Figura 2.3) [42]:

- Uma ou mais conexões de entrada (*input*), com a tarefa de recolher sinais numéricos de outros neurónio - a cada conexão é atribuído um peso que será usado para considerar cada sinal enviado;
- Uma ou mais conexões de saída (*output*) que transportam o sinal para os outros neurónios;
- Uma função de ativação, que determina o valor numérico do sinal de saída, com base nos sinais recebidos das conexões de entrada com outros neurónios. São adequadamente recolhidos os pesos associados a cada sinal recebido e o limite de ativação do próprio neurónio.

O processo de aprendizagem de uma rede neuronal apresenta-se como um processo iterativo de otimização dos pesos, consequentemente, supervisionado. Os pesos são modificados em função do desempenho da rede num conjunto de exemplos pertencentes ao conjunto de treino. O objetivo é minimizar a função de perda, que indica o grau em que o comportamento da rede se desvia do comportamento desejado. O desempenho da rede é, então, verificado num conjunto de teste que consiste em objetos diferentes daqueles do conjunto de treino [42].

Um algoritmo do tipo supervisionado usualmente usado é o algoritmo de retropropagação. As etapas básicas do procedimento de treino são as seguintes [42]:

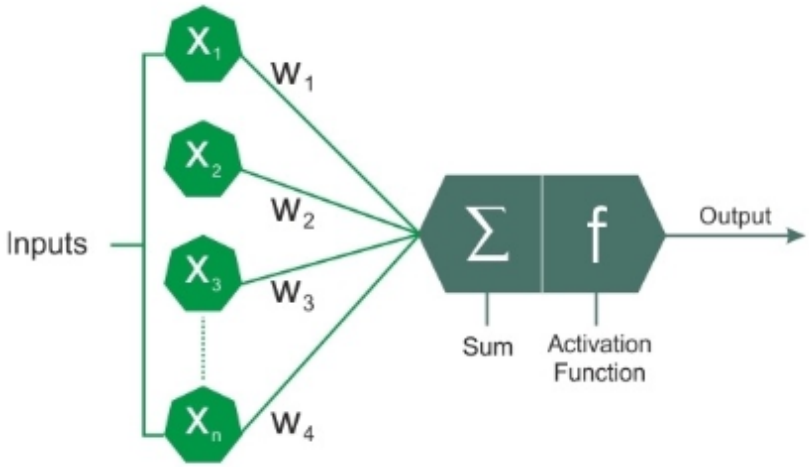


Figura 2.3: Modelo do neurónio artificial (adaptado de [42])

1. Inicialização da rede com pesos aleatórios;
2. Para todos os casos de treino:
   - *Forward pass*: Calcula o erro da rede, ou seja, a diferença entre a saída desejada e a saída real;
   - *Backward pass*: Para todas as camadas, começando com a camada de saída de volta à camada de entrada - i) mostra a saída da camada de rede com a entrada correta (função de erro); ii) adapta os pesos na camada atual para minimizar a função de erro. Esta é a etapa de otimização da retropropagação.

O processo de treino termina quando o erro no conjunto de validação começa a aumentar, uma vez que este pode representar o início de um processo de *overfitting*, ou seja, a fase em que a rede tende a interpolar os dados de treino em detrimento da generalização [42].

Em suma, os algoritmos de DL são um conjunto de ANN's, que podem executar melhores representações de conjuntos de dados em grande escala, de modo a construir modelos que aprendam essas representações extensivamente [42].

2.6.9 *Overfitting e Underfitting*

Como referido anteriormente, o objetivo de um modelo de ML/DL é aproximar uma função desconhecida que associa elementos de entrada aos de saída. No entanto, um conjunto de treino é normalmente uma representação de uma distribuição global, mas não pode conter todos os elementos possíveis; caso contrário, o problema poderia ser resolvido com uma associação um para um [45]. Da mesma forma, não conhecemos a expressão analítica de uma possível função subjacente, portanto, ao treinar, é necessário ajustar o

modelo. Porém é necessário mantê-lo capaz de generalizar quando um *input* desconhecido é apresentado. Infelizmente, essa condição ideal nem sempre é fácil de encontrar, sendo importante considerar duas possibilidades [45]:

- ***Underfitting*:** verifica-se quando o modelo não é capaz de identificar a dinâmica demonstrada pelo mesmo conjunto de treino (provavelmente porque a capacidade é limitada);
- ***Overfitting*:** verifica-se quando o modelo apresenta uma capacidade excessiva, deixando de ser capaz de generalizar considerando a dinâmica original proporcionada pelo conjunto de treino. Ele pode associar quase perfeitamente todas as amostras conhecidas aos valores de saída correspondentes, mas quando uma entrada desconhecida é apresentada, o erro de predição correspondente pode ser muito alto.

Deste modo, é essencial evitar fenómenos de *underfitting* e *overfitting*, sendo que o primeiro é mais fácil de detetar, considerando o erro de previsão. Por outro lado, o *overfitting* pode ser mais difícil de identificar, considerando um resultado inicial aparentemente de um ajuste perfeito [45].

2.6.10 *Estimativa de erros*

Atualmente existem muitas métricas que podem ser usadas para avaliar o aumento da capacidade do programa em executar uma determinada tarefa, de uma forma mais eficaz. Para problemas de ML supervisionado, muitas métricas de desempenho medem o número de erros de previsão. Pode-se dizer que existem duas causas fundamentais para o erro de previsão: o *bias* do modelo e a sua variância [40].

Por um lado, um modelo com um baixo *bias* e elevada variância significa que teremos resultados mais próximo do objetivo pretendido, mas que estão mais dispersos, e consequentemente, mal agrupados. Por outro lado, um modelo com baixo número de *bias* e baixa variância significa que os resultados estão próximos do objetivo desejado, com um bom agrupamento, como se pode perceber pela Figura 2.4. Assim, idealmente, um modelo terá *bias* e variância baixos, no entanto a tentativa de diminuir um destes leva, frequentemente, ao aumento do outro. A isto chama-se o *bias-variance trade-off* [40].

Os sistemas de ML devem ser avaliados usando medidas de desempenho que representam os custos associados a erros [40].

Neste sentido, surge o conceito de **matriz de confusão** (Tabela 2.2), que sendo a base das métricas usadas para avaliar um modelo de ML, é uma matriz que ajuda a determinar o desempenho dos modelos de classificação, para um determinado conjunto de dados de teste. A matriz é dividida em duas dimensões - valores previstos e reais.

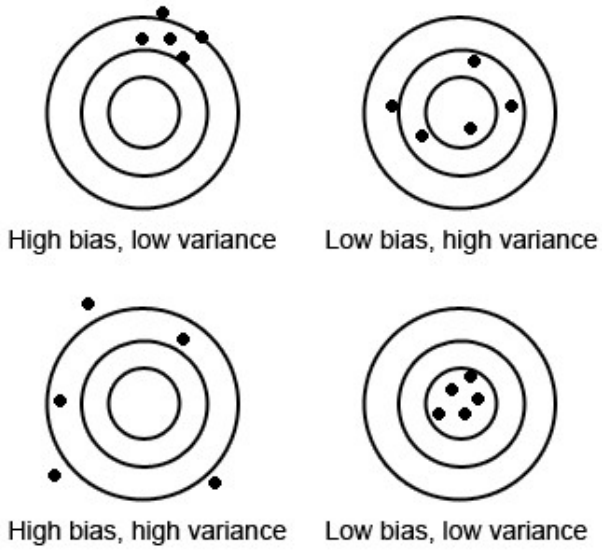


Figura 2.4: *Bias-variance trade-off* (adaptado de [40]).

Tabela 2.2: Matriz de confusão

	Reais positivos	**Reais negativos**
Preditivo Positivo	Verdadeiros positivos (VP)	Falsos positivos (FP)
Preditivo Negativo	Falsos negativos (FN)	Verdadeiros negativos (VN)

Os valores previstos são os valores resultantes da previsão do modelo, e os valores reais são os valores verdadeiros para as observações fornecidas. A necessidade da matriz de confusão em ML, prende-se com o facto desta avaliar o desempenho dos modelos de classificação, quando fazem previsões sobre os dados de teste, e informa o quão bom é o modelo de classificação. Por outro lado, esta indica-nos os erros cometidos pelos classificadores, e também o tipo de erro - erro do tipo I e tipo II. Finalmente, a matriz de confusão, permite-nos calcular diferentes parâmetros do modelo, como exatidão, precisão, entre outros [54].

A **exatidão**, ou seja a fração de instâncias classificadas corretamente, é uma medida intuitiva do desempenho do programa. Embora a exatidão meça o desempenho do programa, ela não estabelece diferença, por exemplo, entre transportadores que foram classificados como não transportadores e não transportadores que foram classificados como transportadores. Em algumas aplicações, os custos associados a todos os tipos de erros podem ser os mesmos [40], [54].

É possível medir cada um dos resultados de previsão possíveis para criar diferentes visões do desempenho do classificador. Quando o sistema classifica corretamente uma

proteína como transportador, a previsão constitui um **verdadeiro positivo** (VP). Por outro lado, quando o sistema classifica incorretamente um não transportador como transportador, a previsão é um **falso positivo** (FP). Da mesma forma, um falso negativo, que constitui um erro Tipo II, é uma previsão incorreta de que a proteína é não transportadora, e um **verdadeiro negativo** (VN), erro Tipo I, é uma previsão correta de que a proteína é não transportadora. Esses quatro resultados podem ser usados para calcular várias medidas comuns de desempenho de classificação, incluindo exatidão, precisão e *recall* [40].

A exatidão é calculada pela fórmula 1:

$$Exatidão(E) = \frac{VP + VN}{VP + VN + FP + FN} \tag{1}$$

A **precisão** é a fração, por exemplo, de proteínas que foram previstos como transportadores que são realmente transportadores. Assim, a precisão é calculada com a fórmula 2:

$$Precisão(P) = \frac{VP}{VP + FP} \tag{2}$$

O ***recall*** é a fração de transportadores que o sistema identificou. O *recall* é calculada pela fórmula 3:

$$Recall(R) = \frac{VP}{VP + FN} \tag{3}$$

Individualmente, a precisão e o *recall* raramente são informativos, uma vez que ambos são visões incompletas do desempenho de um classificador. Tanto a precisão como o *recall* podem falhar na distinção de classificadores com bom desempenho de certos tipos de classificadores com desempenho insatisfatório [40].

Se dois modelos apresentarem baixa precisão e alto *recall* ou vice-versa, torna-se difícil comparar esses modelos [54]. Assim, neste sentido, surge a medida **F1**, uma medida ponderada, das pontuações de precisão e *recall*. Essa pontuação ajuda-nos a avaliar o *recall* e a precisão ao mesmo tempo [40]. A pontuação F é máxima se o *recall* for igual à precisão. Calcula-se usando a fórmula 4 [54]:

$$F1 = 2 \times \frac{P \times R}{P + R} \tag{4}$$

A *Receiver Operating Characteristic*, ou curva ROC, visualiza o desempenho de um classificador [40]. A ROC é traçada entre a taxa de VP (eixo Y) e a taxa de FP (eixo x) [54]. Ao contrário da precisão, a curva ROC é insensível a conjuntos de dados com proporções de classe desequilibradas, ilustrando o desempenho do classificador para todos os valores do limite de discriminação. As curvas ROC representam o *recall* do classificador contra a sua

queda. *Fall-out*, ou taxa de FP, é o número de FP dividido pelo número total de FN. Este último é calculado usando a seguinte fórmula 5 [40]:

$$F = \frac{FP}{VN + FP} \tag{5}$$

A **AUC é a área sob a curva** ROC, sendo que esta reduz a curva ROC a um único valor, representando o desempenho esperado do classificador [40].

Atualmente, o **coeficiente de correlação de *Matthews* (CCM)** tem-se tornado uma métrica amplamente usada na área da biomédica. O CCM pode ser calculado diretamente da matriz de confusão, conforme mostrado na Equação 6. O CMM é um coeficiente de correlação entre as classificações binárias observadas e previstas, podendo apresentar valores entre -1 que indica predição inversa (discordância completa) e 1 que indica uma predição perfeita (concordância completa). Esta é considerada, também, uma medida balanceada que pode ser usado mesmo em situações de conjuntos de dados desequilibrados [55].

$$CCM = \frac{VP \times VN - FP \times FN}{\sqrt{(VP + FN)(VP + FP)(VN + FP)(VN + FN)}} \tag{6}$$

Finalmente, a validação cruzada é uma técnica usada para validar a eficiência do modelo, treinando-o no subconjunto de dados de entrada e testando num subconjunto não visto dos dados de entrada. Pode-se dizer que é uma técnica que verifica como um modelo estatístico se generaliza para um conjunto de dados independente. Em ML, existe sempre a necessidade de testar a estabilidade do modelo. Para isso, primeiro reserva-se uma amostra específica do conjunto de dados, que não faz parte do conjunto de dados de treino, e de seguida testa-se o modelo nessa amostra, antes da implementação. As etapas básicas das validações cruzadas são [54]:

- Reservar um subconjunto do conjunto de dados como conjunto de validação;
- Fornecer o treino para o modelo, usando o conjunto de dados de treino;
- Avaliar o desempenho do modelo usando o conjunto de validação. Se o modelo funcionar bem com o conjunto de validação, executar as etapa seguintes.

Para proceder à validação cruzada, exestem diversos métodos, tais como: abordagem do conjunto de validação, *Leave-P-out cross-validation*, *Leave one out cross-validation*, validação cruzada *K-fold* e validação cruzada *k-fold* estratificada. Das diferentes abordagens, uma das mais usadadas é a validação cruzada *K-fold* [54].

A validação cruzada *k-fold* é uma abordagem em que se divide o conjunto de dados de entrada em k grupos de amostras, chamadas de *folds*, de igual tamanho. Para cada conjunto de treino, a função de previsão usa k-1 *folds*, sendo que o resto das *folds* são usadas para o

conjunto de teste. Esta é uma abordagem de validação cruzada muito utilizada por ser fácil de aplicar e a saída é menos tendenciosa que a dos outros métodos [54].

As etapas para validação cruzada *k-fold* passam por:

- Dividir o conjunto de dados de entrada em k grupos;
- Para cada grupo:
  - Considerar um grupo como reserva ou conjunto de dados de teste;
  - Utilizar os restantes grupos como o conjunto de dados de treino;
  - Ajustar o modelo, baseado-se no conjunto de dados de treino e avaliar o desempenho do modelo, usando o conjunto de dados de teste.

Exemplificando, numa validação cruzada de 5-*folds*, o conjunto de dados é agrupado em 5 *folds*. Na primeira iteração, a primeira *fold* é reservada para testar o modelo, e as restantes são usadas para treinar o modelo. Na segunda iteração, a segunda *fold* é usada para testar o modelo, sendo que as restantes são usadas para treinar o modelo, e assim sucessivamente, até que cada *fold* não seja usada na *fold* de teste [54].

2.7 CONCEITOS E MODELOS DE *deep learning*

DL é um subconjunto da ML, sendo essencialmente uma rede neuronal com três ou mais camadas. Essas redes neuronais pretendem simular o comportamento do cérebro humano, ainda que longe de corresponder à sua capacidade, permitindo que este "aprenda"com grandes quantidades de dados. Embora uma rede neuronal com uma única camada possa fazer previsões aproximadas, camadas ocultas adicionais ajudam a otimizar e refinar a precisão. O DL permite fazer muitas aplicações e serviços de AI que melhoram a automação, realizando tarefas analíticas e físicas sem intervenção humana [56].

O DL diferencia-se do ML clássico pelo tipo de dados que usa e pelos métodos utilizados no processo de "aprendizagem". Assim, por um lado, os algoritmos usados em ML aproveitam dados estruturados e rotulados para fazer previsões, o que significa que os recursos são específicos e definidos a partir dos dados de *input* para o modelo e organizados em tabelas. Esse facto não significa, necessariamente, que não use dados não estruturados, mas sim que, geralmente, existe necessidade de algum tipo de pré-processamento para os tornar num formato estruturado. Por outro lado, os modelos de DL eliminam a necessidade de pré-processamento de dados. Estes algoritmos podem receber e processar dados não estruturados, como texto e imagens, e automatizar a extração de recursos, removendo a dependência da necessidade de especialistas humanos [56].

Atualmente, os métodos de DL são extensamente utilizados nas mais variadas áreas, tais como a visão computacional, previsão, análise de semântica, processamento de linguagem

natural e recuperação de informação. Ainda assim, pode-se destacar a sua importância ao nível do processamento de imagem, problemas como reconhecimento de fala e, finalmente, predição a partir de texto [57].

A forma como se conectam os nós e o número de camadas presentes (os níveis de nós entre a entrada e a saída e o número de neurónios por camada) define a arquitetura da rede neuronal. Existem vários tipos de arquiteturas em redes neuronais. Podemos categorizar as arquiteturas DL em quatro grupos: Redes Neuronais Profundas (DNN), Redes Neuronais Convolucionais (CNN), Redes Neuronais Recorrentes (RNN) e Arquiteturas Emergentes (AE) [42].

As DNNs (Figura 2.5) são ANN fortemente orientados para DL, onde os procedimentos normais de análise não são aplicáveis, devido à complexidade dos dados a serem processados. Tais redes são, portanto, uma excelente ferramenta de modelação. As DNN são redes neuronais muito semelhantes às ANN, mas devem implementar um modelo mais complexo - um maior número de neurónios, camadas ocultas e conexões -, embora sigam os princípios de aprendizagem que se aplicam a todos os problemas de ML. O cálculo em cada camada transforma as representações na camada abaixo em representações um pouco mais abstratas [42].

Em redes de multicamadas, é possível identificar os neurónios artificiais das camadas, de modo que cada neurónio seja conectado a todos os da próxima camada, garantindo que [42]:

- há conexões entre neurónios pertencentes à mesma camada;
- não há conexões entre neurónios pertencentes a camadas não adjacentes;
- o número de camadas e neurónios por camada depende do problema a ser resolvido.

As camadas de entrada e de saída definem entradas e saídas, e existem camadas ocultas, cuja complexidade leva a diferentes comportamentos da rede. Finalmente, as conexões entre os neurónios são representadas por tantas matrizes quanto os pares de camadas adjacentes. Cada *array* contém os pesos das conexões entre os pares de nós de duas camadas adjacentes. As redes *feedforward* (FNN) são redes sem *loops*, dentro das camadas [42].

FNN consiste num grande número de neurónios, organizados em camadas: uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Cada neurónio de uma camada está conectado com todos os neurónios da camada anterior, embora as conexões não sejam todas iguais devido aos diferentes pesos que apresentam. Os pesos dessas conexões codificam o conhecimento da rede. Os dados entram nas entradas e passam pela rede, camada por camada, até chegarem às saídas. Durante esta operação, não há *feedback* entre as mesmas. Portanto, esses tipos de redes são chamados de redes FNN [42].

Para as mais variadas aplicações e possíveis áreas alvo, existem diversos modelos de DL, repetivamemte indicados. No caso específico do projeto apresentado, dados de sequências biológicas, os modelos de DL que mais se adequam são as CNN e RNN, nomeadamente as

Long short-term memory (LSTM) e *Bi-directional* RNN (BRNN), tal como pode ser verificado em várias publicações científicas [58],[59] e pela dissertação de mestrado da colega Andrea Silva.

2.7.1 *Convolutional Neural Networks*

Uma CNN (Figura 2.6) compreende três principais camadas, cuja função varia consoante a mesma. Os tipos de camadas neuronais são os seguintes [60]:

- ***convolutional layers**:* uma CNN utiliza vários *kernels* para envolver a imagem inteira, bem como os mapas de recursos intermediários, gerando vários mapas de recursos. Devido às vantagens da operação de convolução, vários trabalhos a propuseram como um substituto para camadas totalmente conectadas, com o objetivo de obter tempos de aprendizagem mais rápidos.

- ***pooling layers**:* responsáveis por reduzir as dimensões espaciais (altura e largura) do volume de entrada para a próxima camada convolucional. Esta não afeta a dimensão de profundidade do volume. A operação realizada por esta camada é chamada, também, de sub-amostragem ou *downsampling*, pois a redução do tamanho leva à perda simultânea de informações. No entanto, essa perda é benéfica para a rede porque a diminuição de tamanho resulta em menos sobrecarga computacional para as próximas camadas da rede, evitando o sobreajuste;

- ***fully connected layers**:* o raciocínio de alto nível na rede neuronal é realizado nestas camadas. Como o próprio nome indica, os neurónios de uma camada totalmente conectada têm conexões completas com todas as ativações na camada anterior. A sua ativação pode, deste modo, ser calculada com uma multiplicação de matriz seguida por um desvio de polarização.

Uma das dificuldades que podem surgir com o treino de CNN é a grande quantidade de parâmetros que o modelo tem de "aprender", podendo levar ao problema de *overfitting*. Além disso, as CNN são frequentemente submetidas a um pré-treino, ou seja, a um processo que inicializa a rede com parâmetros pré-treinados, em vez de parâmetros definidos aleatoriamente. O pré-treino pode acelerar o processo de aprendizagem e aumentar a capacidade de generalização da rede [60].

No geral, as CNN mostraram superar significativamente as abordagens tradicionais relativamente à ML, numa ampla gama de tarefas de visão computacional e reconhecimento de padrões. O seu desempenho excecional, aliado à relativa facilidade de treino são os principais motivos que justificam o aumento da sua utilização nos últimos anos [60].

2.7.2 *Recurrent Neural Networks*

Numa RNN (Figura 2.7), há uma correspondência *one-to-one* entre as camadas da rede e as posições específicas na sequência. A posição na sequência também é conhecida como *time-stamp*. Assim, em vez de um número variável de entradas numa única camada de entrada, a rede contém um número variável de camadas, sendo que cada camada tem uma única entrada correspondente a esse *time-stamp*. Deste modo, as entradas podem interagir diretamente com as camadas ocultas *down-stream*, dependendo apenas das posições na sequência. Cada camada usa o mesmo conjunto de parâmetros para garantir uma modelação semelhante em cada registo de *time-stamp* e, portanto, o número de parâmetros também é fixo. Por outras palavras, a mesma arquitetura em camadas é repetida no tempo e, portanto, a rede é referida como recorrente. As RNN classificam-se, também, como FNN, com uma estrutura específica baseada na noção de camadas de tempo, para que possam incorporar uma sequência de entrada e produzir uma sequência de saída. Cada camada temporal pode receber um ponto de dados de entrada (um ou vários atributos) e, opcionalmente, produzir uma saída multidimensional. Este tipo de modelos são particularmente úteis em diversas áreas, como nas áreas de sequência a sequência, como tradução automática ou para prever o próximo elemento de uma sequência [61].

No entanto, existem desafios significativos nas RNN, nomeadamente ao nível dos parâmetros de uma rede neuronal recorrente. Um dos principais problemas, neste contexto, é o problema do gradiente de desaparecimento e explosão. Este problema é particularmente prevalente no contexto de redes profundas, como RNN. Como resultado, uma série de variantes da RNN, como as LSTM e as BRNN, foram propostas. Redes neuronais recorrentes e as suas variantes têm sido usadas no contexto de uma variedade de aplicações, como aprendizagem sequência a sequência, legendagem de imagens, tradução automática e análise de sentimento [61].

2.7.3 *Long short-term memory e Bi-directional RNN*

A rede neuronal LSTM (Figura 2.8) foi recentemente redescoberta no contexto de DL, pelo facto de neste tipo de DNN não se colocar o problema de gradientes de desaparecimento, oferecendo ótimos resultados e desempenho. As redes baseadas em LSTM são ideais para a previsão e classificação de sequências temporais, substituindo, atualmente, muitas abordagens tradicionais de DL [42].

O nome desta rede neuronal significa que os padrões de curto prazo não são esquecidos, a longo prazo. Uma rede LSTM é composta por células (blocos LSTM) conectadas entre si. Cada bloco LSTM contém três tipos de portas: uma porta de entrada, uma porta de saída e uma porta de esquecimento, que implementa as funções de escrita, leitura e reinicialização

na memória da célula, respectivamente [42]. As LSTM são um tipo de rede neuronal onde a saída das etapas anteriores é alimentada como entrada para a etapa atual, portanto, lembra algumas informações sobre a sequência. Estas são ótimas quando se fala em contextos curtos, mas tem a limitação de memorizar sequências mais longas devido ao problema do gradiente de desaparecimento [62].

Neste sentido, as redes LSTM são versões melhoradas das RNN, uma vez que são especializadas em memorizar informações por um período prolongado, devido a um mecanismo de passagem, que as torna seletivas. Este define as informações anteriores a serem lembradas e as que deverão ser esquecidas, bem como a quantidade de entrada de corrente a adicionar, para construir o estado atual da célula [62].

Atualmente, de modo a evitar problemas de gradiente de desaparecimento, surgem também as BRNN que assentam na ideia de que a saída no tempo (t) pode depender dos elementos anteriores e futuros na sequência. Para lidar com isso, a saída de dois RNN deve ser aplicada: um executa o processo numa direção e o outro executa o processo na direção oposta [42].

A classificação final ou camada de *clustering* de um modelo de DL conduzido por camadas de redes neuronais totalmente conectadas, pode resultar em *overfitting* [63]. Assim, surgem os **modelos híbridos**, mais especificamente as redes híbridas CNN-LSTM, que constitui um modelo versátil e adequado para uma ampla gama de tarefas de classificação. As redes LSTM podem recordar seletivamente padrões por um longo período de tempo e as redes CNN extrair os recursos importantes [64].

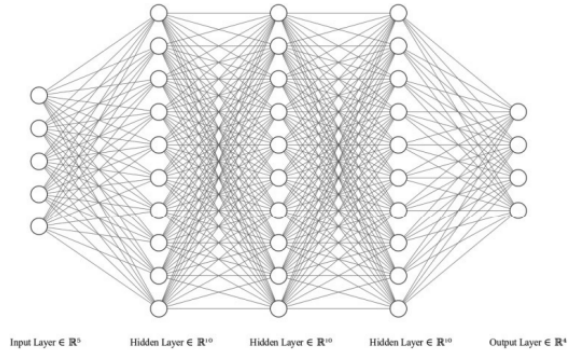


Figura 2.5: *Perceptron* multicamadas (3 camadas ocultas) ou FNN

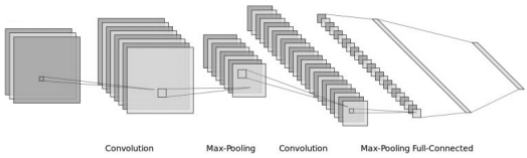


Figura 2.6: Arquitetura CNN

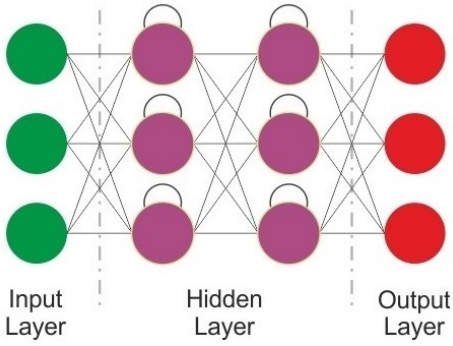


Figura 2.7: Arquitetura RNN

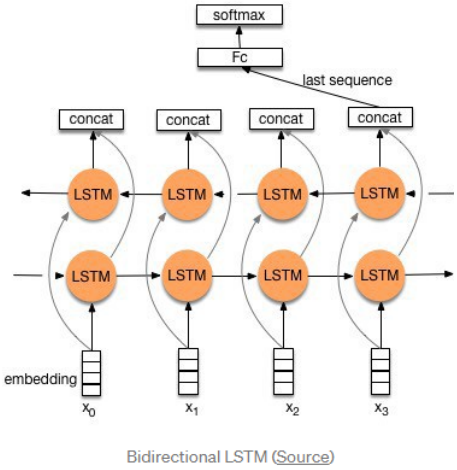


Figura 2.8: Arquitetura *Bidirecional LSTM*

3

MÉTODOS

Neste capítulo serão explicados os passos necessários à construção das diferentes abordagens da ferramenta de deteção de transportadores, a partir de um genoma/proteoma inferido.

O presente capítulo será sub-dividido em duas principais secções: i) ferramenta desenvolvida com base na obtenção de resultados através do acesso remoto a ferramentas bioinformáticas *online* e ii) geração de modelos baseados em ML e DL, para previsão de transportadores.

O ponto fulcral no desenvolvimento da primeira metodologia consiste no acesso remoto, por *Application Programming Interface* (API), aos servidores das plataformas utilizadas na obtenção dos dados necessários à previsão de transportadores, num genoma/proteoma.

Um dos pontos inicias e cruciais da segunda metodologia passa pela criação de um *dataset* que será usado para criar e otimizar (treinar) o modelo a criar e, finalmente, auxiliar o utilizador a prever a existência de transportadores num genoma/proteoma a analisar. O ponto seguinte passa pela criação dos modelos e a sua avaliação com as técnicas adequadas, como *cross validation* e estimação de erros.

Para o desenvolvimento das diferentes metodologias de classificação de transportadores, foram utilizadas diversas plataformas/*softwares* apresentadas na Secção 2.5, que pelas suas especificidades, foram aplicadas consoante as necessidades.

3.1 TRANSPREDICT - FERRAMENTA DE CLASSIFICAÇÃO DE TRANSPORTADORES

Para o desenvolvimento da ferramenta em questão, utilizaram-se como base algumas das plataformas referidas anteriormente - *TMHMM*, *Pred-TMBB*, *Prosite* e CDD. A ferramenta foi desenvolvida em *Python* e apresenta a seguinte estrutura geral (Figura 3.1):

- **Camada *input***: recebe como *input*, um ficheiro do tipo fasta, com a sequência que se pretende analisar/estudar.
- **Camada de processamento de dados**: fase em que se executam os algoritmos através de bibliotecas (*TMHMM*) **ou** se acede aos dados das plataformas (*Pred-TMBB*, *Prosite* e CDD), através de pedidos *Hypertext Transfer Protocol* (HTTP) API dos servidores

das mesmas. Neste ponto, faz-se um processamento do resultado, conjugando a informação com o resultado final, apresentado no *output*.

- **Camada de *output***: extração da informação em formato *.txt* ou obtenção da informação retirada das plataformas na linha de comandos, com a seguinte estrutura:
  - TMHMM: número de TMHs, estrutura α-hélice ou não
  - *Pred-TMBB*: *score* da sequência, estrutura β-folha ou não
  - *Prosite*: número de *hits*; para cada *hit*: *keywords* e *penalty keywords* encontradas na descrição, código da família e *score*
  - CDD: número de *hits*; para cada *hit*: *keywords* e *penalty keywords* encontradas na descrição, código da família e *e-value*
  - *Final result* (FR): percentagem final

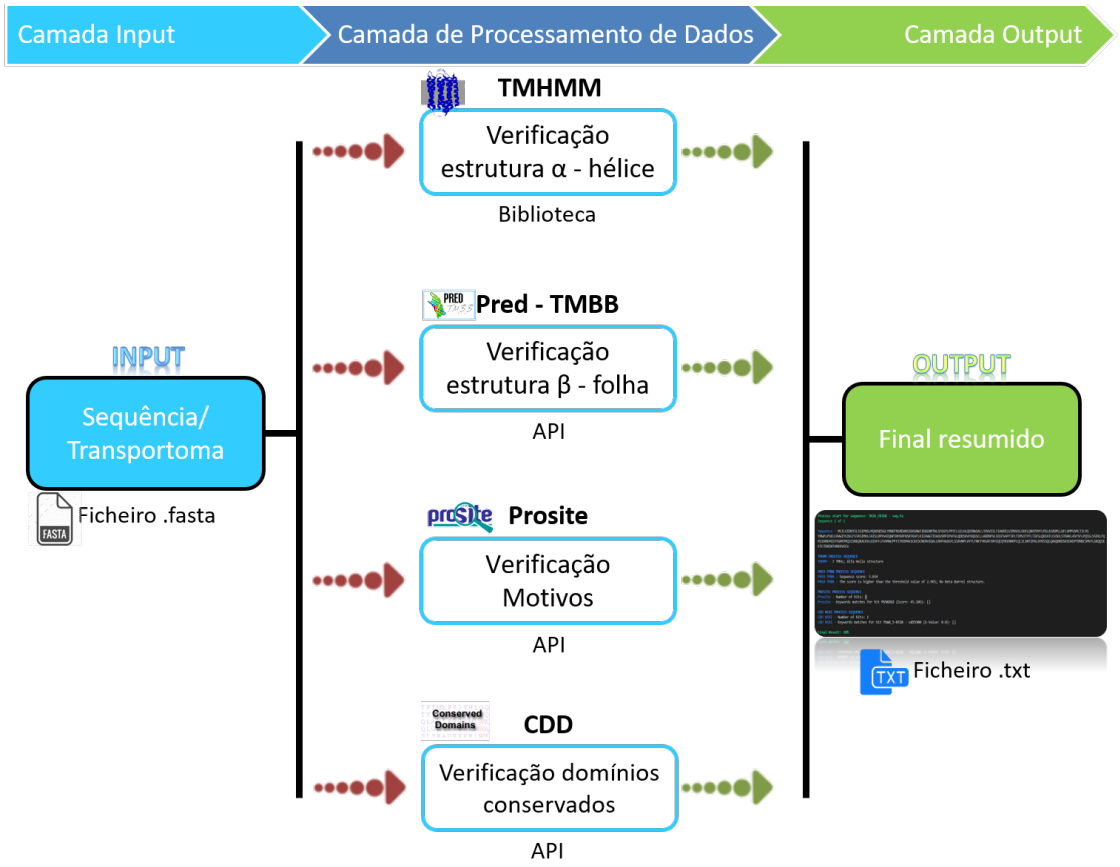


Figura 3.1: *Pipeline* geral da ferramenta *TransPredict*.

De modo a facilitar a visualização e a leitura de dados na linha de comandos, foi implementada uma classe onde é possível aplicar diferentes esquemas de cores, ao gosto do utilizador.

3.1.1 *Input*

Relativamente ao *input*, a ferramenta criada aceita apenas formato fasta. O formato fasta é amplamente usado no contexto da bioinformática e bioquímica, pelo facto de representar sequências nucleotídicas ou peptídicas. Uma sequência neste formato, começa com uma descrição de linha única, seguida por linhas de dados da sequência. A linha da descrição é diferenciada das seguintes pelo símbolo ">", na primeira coluna [65]. A palavra após ">"refere-se ao identificador da sequência e o restante conteúdo à descrição opcional [66]. É, ainda, recomendável que todas as linhas de texto tenham no máximo 80 caracteres [65].

Neste ponto, para proceder à leitura e validação da sequência, utilizou-se a classe *SeqIO*, da biblioteca *Biopython*. A função *Bio.SeqIO.parse()*, usa um identificador de arquivo de entrada (ou nome de arquivo) e formata uma *string*. Esta função retorna os objetos do tipo *SeqRecords* [67].

3.1.2 *Processamento de dados - Bibliotecas*

No que diz respeito ao processamento de dados, como já foi referido, foi necessário aceder a plataformas disponíveis na *web*, e como tal, foram utilizadas bibliotecas específicas para as mesmas.

Relativamente à execução do TMHMM, utilizou-se a biblioteca específica [68] para o processamento do algoritmo do modelo oculto de *Markov*, um modelo estatístico que determina os parâmetros ocultos a partir de parâmetros observáveis [69]. A base algorítmica utilizada neste modelo é o Algoritmo de *Viterbi*, um algoritmo de programação dinâmica atualmente utilizada para encontrar sequências mais prováveis de estados ocultos, especialmente no contexto das fontes de informação de *Markov* e dos modelos ocultos de *Markov* [70]. Com esta biblioteca é possível obter-se a anotação como uma sequência e as probabilidades de cada domínio, domínios esses internos, transmembranares e externos.

Relativamente às restantes plataformas - *Pred-TMBB*, *Prosite* e CDD -, pelo facto de estas serem acedidas por meio de pedidos *HTTP API* dos servidores, foi necessário usar a biblioteca *requests*. Esta biblioteca disponibiliza funções que permitem fazer os pedidos de uma forma mas simples, sem necessidade de adicionar manualmente cadeias de consulta aos seus *Uniform Resource Locator* (URL) ou de encriptação de dados [71].

3.1.3 *Output*

Uma vez que o *output* final pretende espelhar o resultado de um *screening* de várias plataformas, o que nos permite detetar e caracterizar, da forma mais precisa possível, a presença de transportadores dentro de um proteoma total, foi necessário tratar e crivar

os dados fornecedios pelas mesmas, bem como atribuir pesos proporcionais ao nível de importância/relevância do resultado intermédio.

Assim, neste sentido, com a execução do TMHMM e do *Pred-TMBB*, é possível inferir acerca da estrutura secundária da proteína. Esta constitui uma condição necessária, mas não suficiente, para classificar a proteína como transportadora. No FR, tem de se verificar uma das condições, ser α-hélice, β-folha ou ambos.

Quanto aos domínios conservados funcionais, o *Prosite* e o CDD indicam-nos quais os *hits* e os respetivos *scores*/*e-value*, bem como, através da descrição da família obtida podemos filtrar os domínios funcionais que se referem a transportadores, através de uma pesquisa por *keywords*. As *keywords* selecionadas foram: *'transport'*, *'channel'*, *'permease'*, *'pump'*, *'facilitator'*, *'symporter'*, *'uniporter'*, *'antiporter'*, *'porin'*, *'carrier'*, *'influx'*, *'eflux'*, *'import'*, *'importer'*, *'export'*, *'exporter'*. Dependendo da janela de valores dos *scores* (no caso do *Prosite*) e *e-values* (no caso do CDD), foi atribuída uma percentagem ao melhor *hit* de 100%, 70%, 30% ou 0%, tal como se pode verificar na Tabela 3.1. Para o resultado final, verificam-se os domínios encontrados no *Prosite* e CDD, tendo em conta o *hit* com melhor *score*/*e-value*, respetivamente.

Tabela 3.1: Relação de percentagem de *hits* de acordo com o valor dos *scores*/*e-values* dos mesmos.

	100%	70%	30%	0%
***Prosite* (*score*)**	s>=18.3	14.3>=s>18.3	9.3>=s>14.3	s<9.3
CDD (*e-value*)	e <=10^{-10}	10^{-10}> e >=10^{-6}	10^{-6}> e >=10^{-1}	e >10^{-1}

Por último, é apresentada o FR que nos indica, numa dada sequência, a certeza com que podemos concluir que estamos perante um transportador. Esse resultado final é obtido com base em pesos conferidos a cada informação obtida, nas respetivas plataformas, por nós definidos. Numa primeira abordagem, aplicou-se a fórmula apresentada na Equação 7.

$$FR = 20\% \times structure + 40\% \times Prosite + 40\% \times CDD \tag{7}$$

FR - *Final result*
structure - Valor do TMHMM ou *Pred-TMBB*
CDD - valor do maior *hit* com *keywords*
Prosite - valor do maior *hit* com *keywords*

De seguida, após uma análise exaustiva do *output* gerado pelo *TransPredict* na predição de transportadores no genoma da *E. coli* e *S. cereviseae*, reajustaram-se os pesos conferidos a cada parcela, tendo-se obtido a fórmula representada na Equação 8.

$$FR = 20\% \times structure + 20\% \times Prosite + 60\% \times CDD \tag{8}$$

Decidiu-se, ainda, aplicar penalizações e "beneficiar"certos parâmetros/resultados:

- Relativamente à parte estrutural (TMHMM e *Pred-TMBB*), aplica-se um incremento de 5% à parcela da estrutura, quando na presença de ambas (α-hélice e β-folha). Na ausência de qualquer uma das estruturas, aplica-se uma penalização de 5% à parcela da mesma;

- No que diz respeito ao TMHMM, na presença de 6 ou mais TMH's é atribuída a totalidade dos 20%, referentes à estrutura. Já no caso de obter um resulado entre 3 e 5 TMH's, atribui-se apenas 50% dos 20% possíveis (10%);

- Em relação às *keywords* extraídas do *Prosite* e do CDD no caso de *hit*, na presença das *keywords* ***sensor*** e ***transfer*** aplica-se uma penalização de 20% ao valor anteriormente definido pelo melhor *hit*. Já no caso das *keywords* ***enzyme*** e ***transferase*** aplica-se uma penalização de 40%, estes termos excluem a possibilidade da proteína ser um transportador. Por outro lado, na presença das *keywords* ***antiporter***, ***symporter*** e ***permease***, é aplicado um incremento de 15%, uma vez que são termos que estão diretamente ligados a transportadores.

De modo a facilitar o tratamento de dados, foi produzido um resumo final, em ficheiro *.txt* com a seguinte estrutura, em forma tabular: nome e descrição, *score*, TMHMM (indica o número de TMH's e se é α-hélice), *Pred-TMBB* (indica se é β-folha), *Prosite* (por cada *hit*, o código do *hit*, o *score* e a lista de *keywords* e *penalty keywords*), CDD (por cada *hit*, o código do *hit*, o *e-value* e a lista de *keywords* e *penalty keywords*). Em adição, gerou-se uma matriz de confusão, de modo a visualizar graficamente os VP, VN, FP e FN. Estes resultados foram obtidos por comparação direta com as sequências extraídas da base de dados do TCDB (da classe 1 à 5).

Definiu-se que para um FR superior a 70%, poder-se-á considerar que estamos perante um possível transportador.

3.1.4 *Modo de execução*

Antes da excecução da ferramenta é necessário instalar, através do *pip install*, os seguintes *packages* específicos: *requests*, *biopython*, *tmhmm.py* e *beautifulsoup4*.

De seguida, para executar a ferramenta, corre-se o *script* no terminal do editor de código, com a seguinte sintaxe: *python* "ficheiro"(no caso *main.py*) argumento com o nome do ficheiro ("*seq.example.faa*").

```
$ python main.py seq.example.faa
```

3.2 FERRAMENTA DE CLASSIFICAÇÃO DE TRANSPORTADORES, BASEADA EM ML E DL

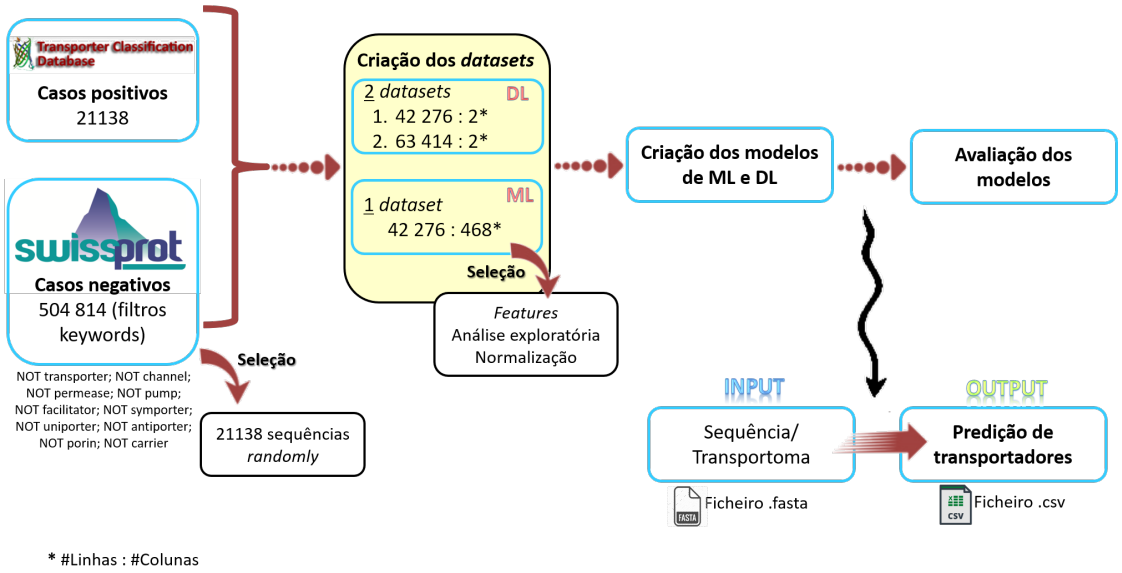


Figura 3.2: *Pipeline* geral da ferramenta baseada em ML e DL.

3.2.1 *Data e Dataset*

Para o desenvolvimento desta dissertação foram usados trabalhos anteriores, baseados em técnicas de ML e DL, cujo objetivo era diferenciar proteínas transportadoras das restantes, com base em características (*features*) extraídas das sequências de aminoácidos. Os referidos trabalhos foram desenvolvidos por Daniel Varzim e Andrea Silva, nas suas dissertações de Mestrado em Bioinformática, da Universidade do Minho.

Dado que o principal objetivo da presente dissertação assenta na comparação de três abordagens - ferramenta baseada em plataformas disponíveis publicamente *online* desenvolvida na unidade curricular de Projeto em Bioinformática e as baseada em ML e DL (Figura 3.2) -, a criação do conjunto de dados e dos modelos foi semelhante à dos colegas referidos.

Para proceder à criação de um modelo capaz de distinguir transportadores (casos positivos) de não transportadores (casos negativos), foram usadas como base sequências de aminoácidos bem anotadas e revistas para gerar recursos.

Relativamente aos casos positivos, estes foram obtidos pelo *download* de um arquivo fasta do TCDB com todas as respetivas proteínas, tendo-se obtido 21138 casos positivos. No caso negativos, estes foram obtidos a partir do *Swiss-Prot*, aplicando alguns filtros - *NOT 'transporter'*, *NOT 'channel'*, *NOT 'permease'*, *NOT 'pump'*, *NOT 'facilitator'*, *NOT 'symporter'*, *NOT 'uniporter'*, NOT *'antiporter'*, *NOT 'porin'* e *NOT 'carrier'* -, perfazendo um total de

504 814 sequências, num arquivo em formato fasta. De seguida, procedeu-se à seleção aleatória de um igual número de sequências às obtidas a partir do TCDB, 21138. Finalmente, juntaram-se as sequências de ambas as plataformas, constituindo o *dataset* total, resultando na obtenção de 42276 sequências.

Para proceder à criação do *dataset* final usado na criação dos **modelos de ML**, foi gerado um ficheiro final onde, para cada sequência (total de 42276 sequências) foram geradas todas as *features* descritas na secção 3.2.2.

Já no que diz respeito aos *datasets* criados para a geração de **modelos de DL**, foram criados 2 *datasets* distintos: um com uma proporção de 1:2 de casos positivos para negativos e outro com igual proporção de casos positivos e negativos. Assim sendo, foi gerado um ficheiro apenas com as sequências (63414 e 42276 sequências, respetivamente) e indicação se eram transportadores (1) ou não (0), de forma binária. Nesta abordagem usou-se ambos os *datasets* com a finalidade de testar qual o *dataset* que apresentava melhores resultados. Assim, utilizou-se ambos os *datasets* na fase de teste das redes e apenas o *dataset* com proporção de 1:2 de casos positivos para negativos na fase de previsão dos transportadores, nos modelos biológicos, por este apresentar melhores resultados. Adicionalmente, estes *datasets* foram divididos em 3 partes - dados de treino (80%), validação (10%) e teste (10%).

3.2.2 *Features*

Devido à importância que as *features* têm no desempenho dos **modelos de ML** criados, e de modo a desenvolver um modelo razoável/bom, foi selecionado um conjunto de 10 atributos de entrada (recursos), gerados para cada proteína, de modo a tentar criar o modelo que melhor se adequasse ao trabalho desenvolvido. As *features* são as seguintes:

- **Ocorrência de aminoácidos**: matriz de 20 colunas em que cada coluna contém o número de vezes que um determinado aminoácido está presente (naa) na sequência da proteína (Equação 9).

$$OcorrênciaAA(aa) = naa \tag{9}$$

- **Composição de aminoácidos:** número de vezes que cada aminoácido está presente na sequência proteica, correspondendo a um *array* com 20 colunas, dividido pelo comprimento da sequência (N) (Equação 10);

$$ComposiçãoAA(aa) = \frac{naa}{N} \tag{10}$$

- **Ocorrência físico-química de aminoácidos:** representado numa matriz de 11 colunas, com a soma da ocorrência de cada aminoácido (aa), inseridos em grupos específicos (nj) (Equação 11). De entre os grupos específicos podem-se referir: aminoácidos alifáticos

(I, L e V) e aromáticos (F, H, W e Y); aminoácidos polares (D, E, R, K, Q e N) e neutros (A, G, H, P, S, T e Y), aminoácidos hidrofóbicos (C , F, I, L, M, V e W); aminoácidos carregados (D, E, K, H e R), subdivididos em aminoácidos carregados positivamente (K, R e H) e negativamente (D e E); aminoácidos minúsculos (A, C, D, G, S e T), pequenos aminoácidos (E, H, I, L, K, M, N, P, Q e V) e, finalmente, aminoácidos grandes (F , R, W e Y).

$$OcorrênciaAA.FQ(j) = \sum njaa \tag{11}$$

- **Composição físico-química de aminoácidos:** matriz de 11 colunas, em que cada coluna contém a ocorrência físico-química do aminoácido mostrada anteriormente, dividida pelo número total de aminoácidos na sequência (Equação 12).

$$ComposiçãoAA.FQ(j) = \frac{\sum njaa}{N} \tag{12}$$

- **Composição dipeptídica:** matriz de 400 colunas que contêm o número de vezes que um determinado dipéptido ocorre dividido pelo número total de dipéptidos, ou seja, o número de aminoácidos na sequência menos um.

- **α-hélice:** representada por 1 coluna. Contém o número de α-hélices estimadas pela ferramenta *Phobius*.

- **β-folha:** matriz de 1 coluna, indicando "1"ou "0", dependendo se o BOMP prevê uma formação de β-folha ou não.

- **Péptidos sinal:** matriz com 400 colunas contendo as proteínas com um péptido sinal, estimado pela ferramenta *Phobius*, representado com "1"e "0"as proteínas que não apresentam qualquer péptido sinal.

- **Localização subcelular:** matriz com 1 coluna que contém números entre 0 e 24, de acordo com a localização subcelular prevista pelo *LocTree3*. Os números representam as seguintes localizações: 0 - erro; 1 - "cloroplasto"; 2 - "membrana do cloroplasto"; 3 - "citosol"; 4 - "retículo endoplasmático"; 5 - "membrana do retículo endoplasmático"; 6 - "extracelular"; 7- "*fimbrium*"; 8- "complexo de Golgi"; 9 - "membrana do complexo de Golgi"; 10 - "mitocôndria"; 11 - "membrana mitocondrial"; 12 - "núcleo"; 13 - "membrana nuclear"; 14 - "membrana externa"; 15 - "espaço periplasmático"; 16 - "peroxissoma"; 17 - "membrana peroxissomal"; 18 - "membrana plasmática"; 19 - "plastídio"; 20 - "vacúolo"; 21 - "membrana do vacúolo"; 22 - "secretado"; 23 - "citoplasma"; 24 - "membrana interna".

- **Nº de domínios *Pfam* relacionados ao transportador:** matriz composta por 1 coluna que contém o número de domínios *Pfam* relacionados com transportador, apresentados

pelas proteínas. Estes são obtidos pelo site *Pfam* com as palavras-chave específicas para o objetivo pretendido, obtendo um total de 4042 transportadores com domínios *Pfam* relacionados.

3.2.3 *Análise exploratória e pré-processamento*

De modo a analisar e investigar o conjunto de dados, resumir as suas principais características, e antes de proceder ao pré-processamento do *dataset* e à implementação dos modelos de ML, realizou-se a análise exploratória do mesmo. Com a análise exploratória torna-se mais fácil detetar padrões e anomalias nos dados utilizados, bem como confirmar características que se pretendiam obter, por exemplo, a partir da filtração de dados.

Para o desenvolvimento da ferramenta de classificação de transportadores, baseada em ML, procedeu-se a uma **análise exploratória** simples, fazendo uso, apenas, de gráficos univariados, nomeadamente, histogramas e gráficos circulares. Neste ponto, fez-se, sucintamente e paralelamente, uma análise relativa ao *dataset* total (transportadores e não transportadores) e ao *dataset* positivo (apenas transportadores).

Numa abordagem inicial, fez-se apenas uma validação da composição do *dataset*, de modo a verificar a relação transportador/não-transportador de 50%-50% no *dataset* total e uma composição exclusiva de transportadores no *dataset* positivo.

De seguida, no respeita às *features* referentes à estrutura, α-hélice e β-folha, verificou-se a composição relativa de estruturas α-hélice e β-folha no *dataset* total e positivo. No caso da estrutura α-hélice, procedeu-se, ainda, ao estudo de prevalência, por criação de um histograma, da quantidade de domínios transmembranares presentes, em ambos os *datasets*.

Numa abordagem posterior, verificou-se qual a distribuição da presença de péptido sinal, tanto no *dataset* total como no *dataset* positivo, pela criação de gráficos circulares.

De seguida, procedeu-se à elaboração de histogramas, para avaliar do 'top 10' de dipéptidos mais prevalentes, quer no *dataset* total quer no positivo.

Finalmente, e também por obtenção de histogramas, realizou-se um estudo da localização celular dos transportadores e não transportadores.

Para preparar os dados brutos e torná-los adequados à utilização em **modelos de ML**, procedeu-se ao **pré-processamento** dos mesmos. Esta torna-se a primeira, e crucial, etapa para criar um modelo de ML. Geralmente, os dados do "mundo real"contêm ruídos, NA's e, por vezes, formatos inutilizáveis, o que se traduz numa impossibilidade de os usar diretamente. Assim, no pré-processamento faz-se uma limpeza de dados de modo a torná-los adequados aos modelos de ML, levando a uma maior precisão e eficiência do mesmo.

Para tornar os dados adequados à criação de modelos de ML, o pré-processamento baseou-se na exclusão de *features* que não eram vantajosas para os modelos, isto é, que

não tinham qualquer influência positiva sobre o mesmo. Para isso, utilizou-se a classe *VarianceThresholde* do módulo *preprocessing*, da biblioteca *scikit-learn* [39]. Uma outra fases relevante no pré-processamento, é a remoção de NA's. Esta executou-se pela aplicação da classe *SimpleImputer*, do *scikit-learn* [39]. A classe *SimpleImputer* fornece estratégias básicas para imputar valores de NA's, sendo que estes foram imputados com um valor da média de cada coluna, onde constavam.

No que diz respeito ao pré-processamento dos dados, para a criação de **modelos de DL**, procedeu-se de uma forma distinta à usada na criação de modelos de ML. Este iniciou-se com a conversão de cada aminoácido das sequências proteicas num número inteiro, por um processo de *integer_enconding*, de modo a possibilitar a implementação do passo seguinte. Este passa por normalizar o tamanho das sequências, para o igualar ao tamanho da maior sequência, pela utilização da função *tf.keras.preprocessing.sequence.pad_sequences* [41], do *TensorFlow*. Como *checkpoint*, neste ponto, criou-se de um *boxplot* (Figura 3.3) de modo a verificar a existência de *outliers* e eliminar sequências que, por constituirem valores atípicos, podem prejudicar o desempenho do modelo. Finalmente, após remoção dos *outliers*, vetorizaram-se os inteiros numa matriz de classe binária, com a função *tf.keras.utils.to_categorical* [41], do *TensorFlow*. Esta última função faz com que a variável codificada de inteiro seja removida, adicionando uma nova variável binária, para cada valor inteiro exclusivo [72].

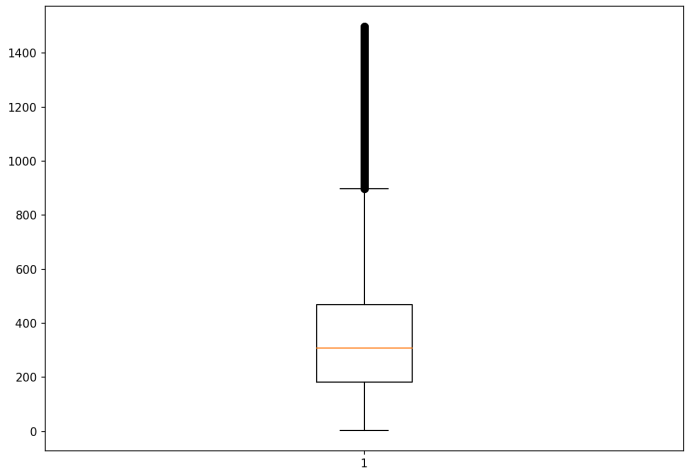


Figura 3.3: Distribuição do tamanho das sequência do *dataset*

3.2.4 *Criação e avaliação dos modelos*

Após a criação do *dataset*, tendo por base as sequências positivas e negativas, bem como as *features* pretendidas, a realização da análise exploratória para melhor compreensão do *dataset* e fazer todo o pré-processamento necessário e crucial, atingiu-se o ponto de criação dos modelos - principal objetivo.

No âmbito da criação dos **modelos de ML**, foram usadas funções disponíveis na biblioteca do *scikit-learn*, para os diferentes modelos criados. Com o objetivo de verificar qual o modelo que melhor se adequa à problemática - previsão de transportadores -, criaram-se 6 modelos distintos, de acordo com os resultados obtidos nas dissertações dos colegas Daniel Varzim e Andrea Silva. Os modelos escolhidos e criados foram:

- **GNB** - implementação da função *BaggingClassifier(GaussianNB())*;
- **RF** - implementação da função *BaggingClassifier(RandomForestClassifier())* e, também, *BaggingClassifier(ExtraTreesClassifier())*, de modo a melhorar o desempenho da função anterior;
- **KNN** - implementação da função *BaggingClassifier(KNeighborsClassifier())*;
- **LR** - implementação da função *BaggingClassifier(linear_model.LogisticRegression())*;
- **GB** - implementação da função *GradientBoostingClassifier()*;
- **SVM** - implementação da função *svm.SVC()*.

Durante o processo de geração dos modelos, foram também executados vários processos de validação, pela análise das métricas adequadas. As métricas usadas foram a exatidão, precisão, *recall*, F1, *ROC-AUC* e CCM. Além disso, fez-se uma validação cruzada 5-*fold*, fazendo uso da função *model_selection.cross_val_score* [39], disponível no *scikit-learn*. Para que o utilizador possa analisar melhor o desempenho do modelo, é gerado para ficheiro .txt com um resumo das principais métricas - exatidão, precisão, *recall*, F1, ROC-AUC e CCM - para cada modelo de ML gerado, baseadas na matriz de confusão respetiva.

Já no que diz respeito à obtenção dos **modelos de DL**, usaram-se funções disponíveis no *ProPythia*, baseadas nas bibliotecas *Keras* e *TensorFlow*, para os diferentes tipos de redes neuronais. De modo a inferir o modelo que melhor se adequa ao objetivo do trabalho, criam-se 3 modelos de DL - CNN, LSTM bidirecionais e híbridas CNN-LSTM. Para criar modelos com precisões e resultados minimamente razoáveis, foi usado o *dataset* não balanceado e aplicadas diferentes parametrizações, aos diferentes níveis. Os parâmetros, com base no *ProPythia*, testados foram:

- *callbacks*: aplicados em todos os modelos o *earlystopping* (importante para evitar fenómenos de *overfitting* [41]), *reduce learing rate* e *model checkpoint*;
- *dropout rate*: aplicado um valor de 0.5/*layer*, em todos os modelos. Esta técnica consiste em desconectar as unidades e as suas conexões, retirando-as temporariamente da rede, durante o treino, evitando que as unidades se co-adaptem em demasia. A unidade descartada é escolhida aleatoriamente e é retida com uma probabilidade fixa, independente de outras unidades [41].
- *hidden-layers*: aplicadas 3 *hidden-layers* nos modelos CNN e 2 nos modelos LSTM bidirecionais;
- *epochs*: valor de 100 em todos os modelos (nunca atingidos devido à utilização do *callback early stopping*);
- *batch size*: valor de 128, em todos os modelos.

3.2.5 *Obtenção de resultados pela previsão de transportadores no genoma de E. coli e S. cerevisiae*

Todo o desenvolvimento da ferramenta baseada em modelos de ML, tem como finalidade prever transportadores a partir de genomas inferidos e, por isso, procedeu-se à previsão dos mesmos em dois genomas de referência, sendo estes o da *E. coli* e da *S. cerevisiae*. Assim, utilizaram-se os métodos referidos anteriormente de geração das *features*, para a geração das mesmas para os dois genomas a testar. Estes sofreram igualmente o mesmo processo de pré-processamento do da criação dos seis modelos. Finalmente, fez-se a previsão dos transportadores, para cada um dos modelos previamente gerados. O resultado final é extraído sob a forma de ficheiro .csv, com a identificação da sequência e a classificação binária dos transportadores (1 - transportador, 0 - não transportador), para cada modelo.

Além dos ficheiros .csv com os resultados, propriamente ditos, foi gerado, também, uma matriz de confusão de cada modelo, bem como as métricas referidas anteriormente, para todos (ML e DL).

4

RESULTADOS E DISCUSSÃO

À semelhança da estrutura adotada no Capítulo 3, o presente capítulo apresenta duas secções, referentes aos resultados de cada metodologia do desenvolvimento do trabalho e, adicionalmente, uma final onde vão ser discutidos os resultados obtidos em cada uma dessas metodologias respetivamente, e também, relacionar os resultados entre as mesmas.

4.1 RESULTADOS DO TRANSPREDICT

De modo a validar a ferramenta desenvolvida, bem como a sua capacidade de deteção de proteínas transportadoras, testaram-se sequências de aminoácidos de várias proteínas (transportadoras e não transportadoras), cujo função é bem conhecida. Além das sequências de aminoácidos de proteínas, testou-se, ainda, o proteoma da *E. coli* e da *S. cerevisiae*, rigorosamente caracterizado e anotado.

Numa primeira abordagem, e de modo a aferir a validade da ferramenta desenvolvida no que diz respeito a sequências de aminoácidos de proteínas, testaram-se as seguintes sequências:

- do transportador *Carboxylic acid transporter protein* codificada pelo gene *JEN1*, presente na *S. cerevisiae*. O resultado obtido usando a ferramenta desenvolvida foi de 100%, o que está de acordo com a informação da classificação e anotação, disponível no *UniProtKB* [73].
- da proteína de sinalização *Bovine rhodopsin* codificada pelo gene *1F88*, presente no *Bos taurus*. Neste último caso, o resultado foi de 20%, o que está de acordo com a sua classificação do *Protein Data Bank* (PDB) de proteína de sinalização [74].

Numa segunda abordagem, gerou-se os *outputs*, para a totalidade dos dois genomas de referência utilizados, *E. coli* e *S. cerevisiae*, disponíveis no https://github.com/ritaconde/TransPredict.

Tabela 4.1: Matriz de confusão e métricas de estimativa de erro, dos resultados do *TransPredict* da *E. coli* e *S. cerevisiae*

	Reais Positivos	Reais Negativos		Estimativas de erro
Preditivo Positivo	278	77	***E. coli***	CCM: 0,44 ACC: 0,84 PRE: 0,78 Recall: 0,32 F1: 0,46
Preditivo Negativo	571	3316		
Preditivo Positivo	229	132	***S. cerevisiae***	CCM: 0,43 ACC: 0,91 PRE: 0,63 Recall: 0,36 F1: 0,46
Preditivo Negativo	417	5223		

De modo a validar a ferramenta utilizada na identificação de transportadores, no proteoma da *E. coli* e da *S. cerevisiae*, gerou-se uma matriz de confusão, bem como as métricas de estimativa de erro, representada em forma de tabela resumo (Tabela 4.1).

Após a análise dos resultados obtidos nos genomas da *E. coli* e *S. cerevisiae*, definiu-se, para se considerar a proteína como um transportador, um valor de *threshold* do FR de 70%.

4.2 Resultados da Ferramenta de classificação, baseada em ML e DL

4.2.1 *Análise exploratória dos modelos de ML*

De modo a melhor conhecer os dados que constituem o *dataset*, procedeu-se à análise exploratória dos mesmos, relativamente a cada *feature*, quer na parte do *dataset* positivo, quer no *dataset* negativo.

Numa primeira abordagem, decidiu-se verificar/comprovar, através de um gráfico circular (Figura 4.1), a relação de 50-50% entre transportadores e não transportadores, do *dataset* utilizado.

De seguida, verificou-se a distribuição de estruturas α-hélice no *dataset* completo e na porção positiva do *dataset*, respetivamente, através de um gráfico circular (Figura 4.2, 4.3). Procedeu-se, ainda, à criação de um histograma, de modo a verificar a distribuição de TMH's, respetivamente, no grupo do *dataset* completo e no positivo do *dataset* (Figura 4.4, 4.5).

De igual modo, verificou-se a distribuição de estruturas β-folha no *dataset* completo e na porção positiva do *dataset*, respetivamente, pela criação de um gráfico circular (Figura 4.6, 4.7).

Relativamente à presença de péptidos sinal, verificou-se num gráfico circular (Figura 4.8, 4.9), a presença relativa dos mesmos no *dataset* total e positivo, respetivamente.

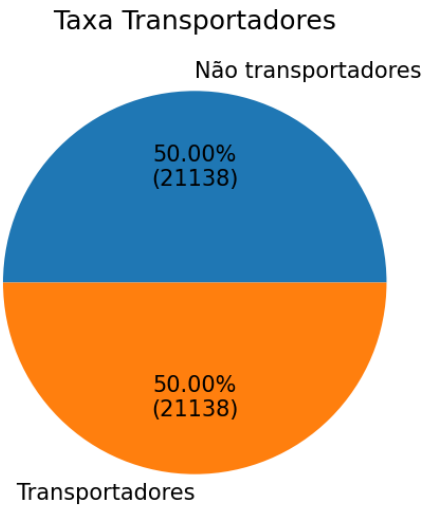


Figura 4.1: Relação transportador/não transportador.

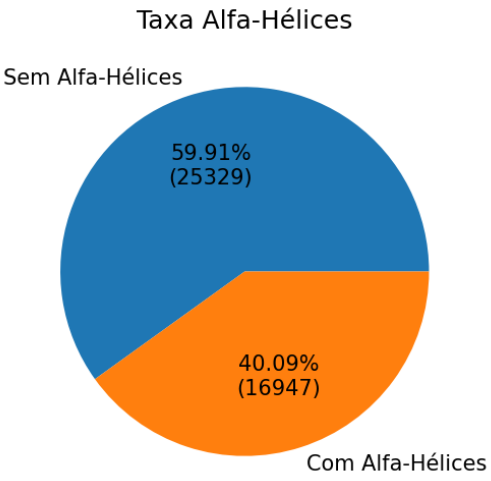


Figura 4.2: Relação α-hélice e não α-hélice, no *dataset*.

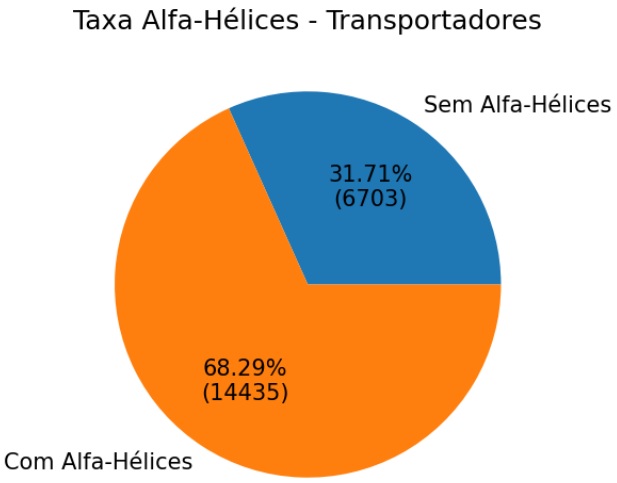


Figura 4.3: Relação α-hélice e não α-hélice, *dataset* positivo.

Dado que no presente *dataset* existem 400 dipéptidos distintos, procedeu-se, à criação de um histograma, para representar o 'top 10' de dipéptidos mais frequentes nos dados usados, quer no *dataset* completo, quer na porção positiva do mesmo (Figura 4.10, 4.11).

Finalmente, fez-se uma análise exploratória, pela criação de um histograma (Figura 4.12, 4.13), de modo a verificar o 'top 10' da localização celular das proteínas (transportadoras e não transportadoras) mais frequentes nos dados, quer no *dataset* completo, quer no *dataset* positivo.

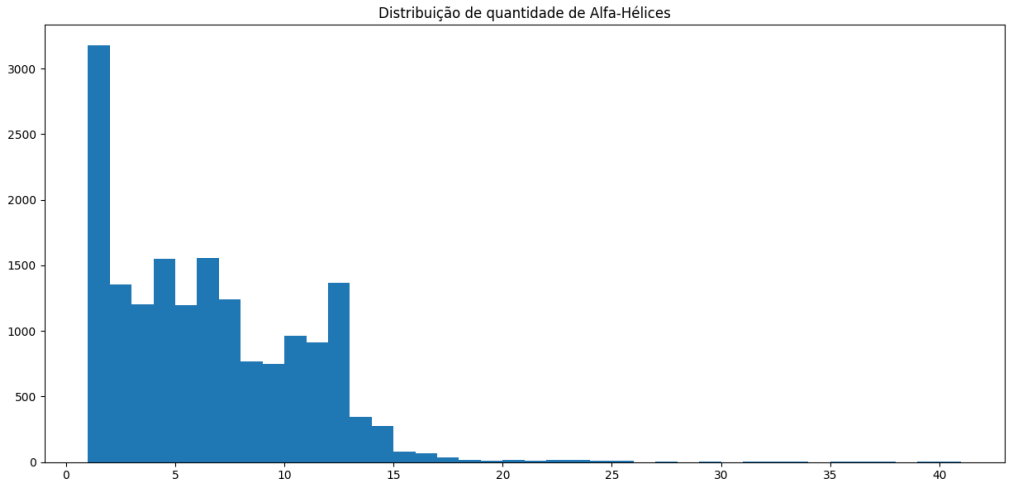


Figura 4.4: Distribuição de TMH's, no *dataset*.

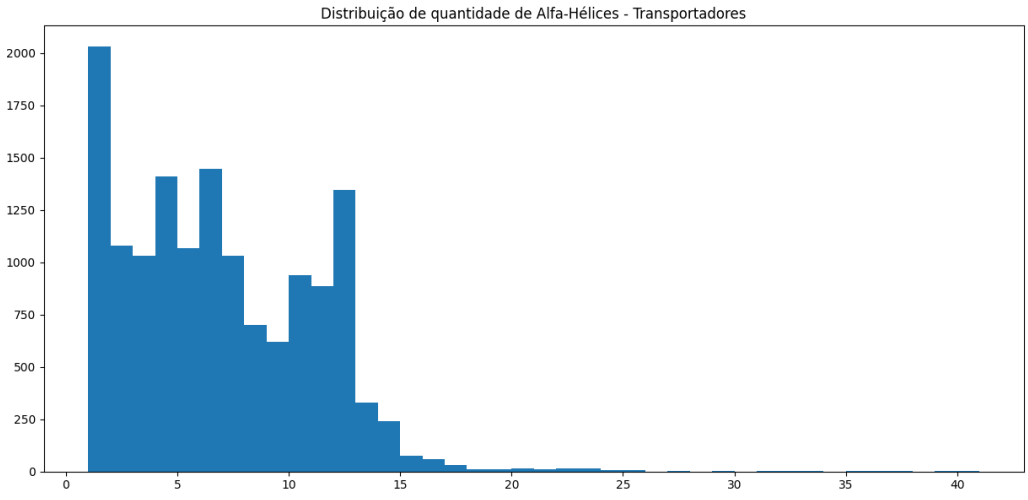


Figura 4.5: Distribuição de TMH's, no *dataset* positivo.

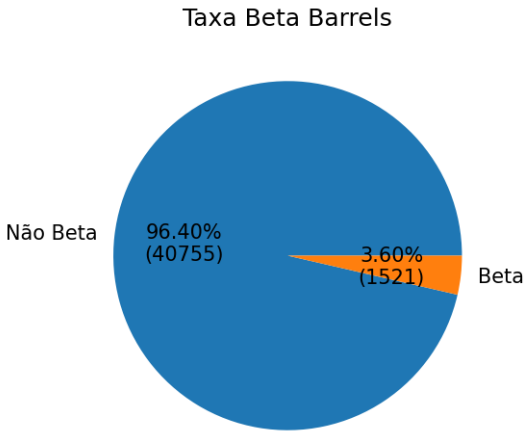


Figura 4.6: Relação β-folha e não β-folha, no *dataset*.

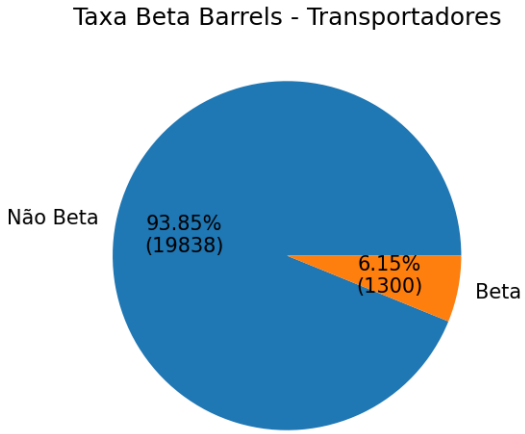


Figura 4.7: Relação β-folha e não β-folha, *dataset* positivo.

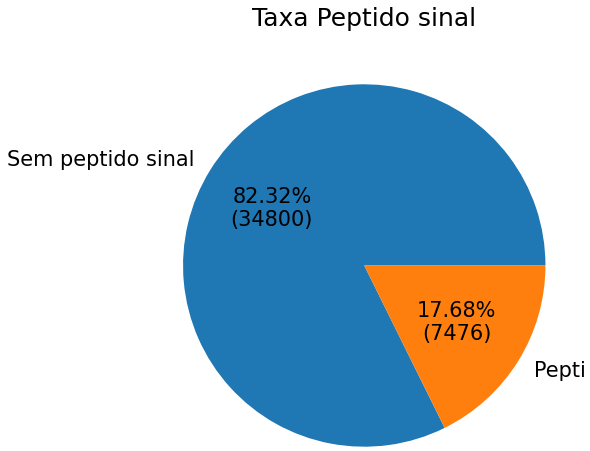


Figura 4.8: Relação da presença/ausência de péptidos sinal, no *dataset*.

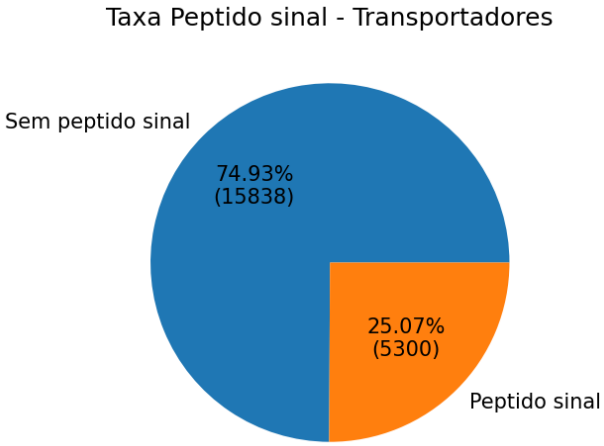


Figura 4.9: Relação da presença/ausência de péptidos sinal, no *dataset* positivo.

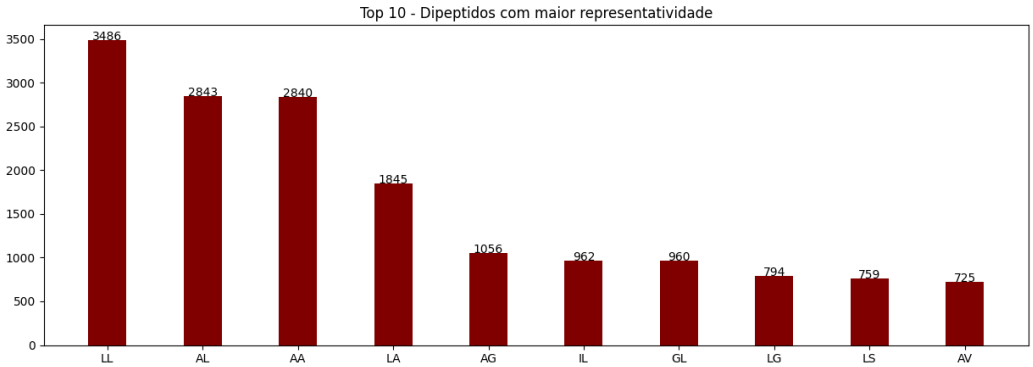


Figura 4.10: 'Top 10' de dipéptidos, no *dataset*.

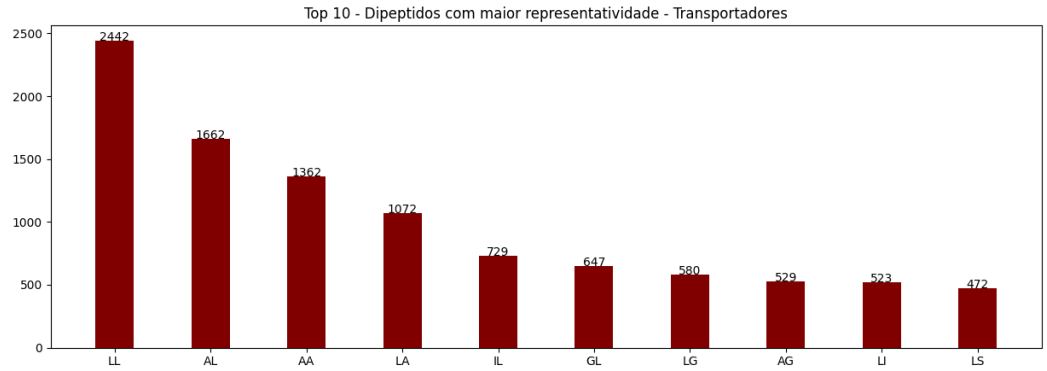


Figura 4.11: 'Top 10' de dipéptidos, no *dataset* positivo.

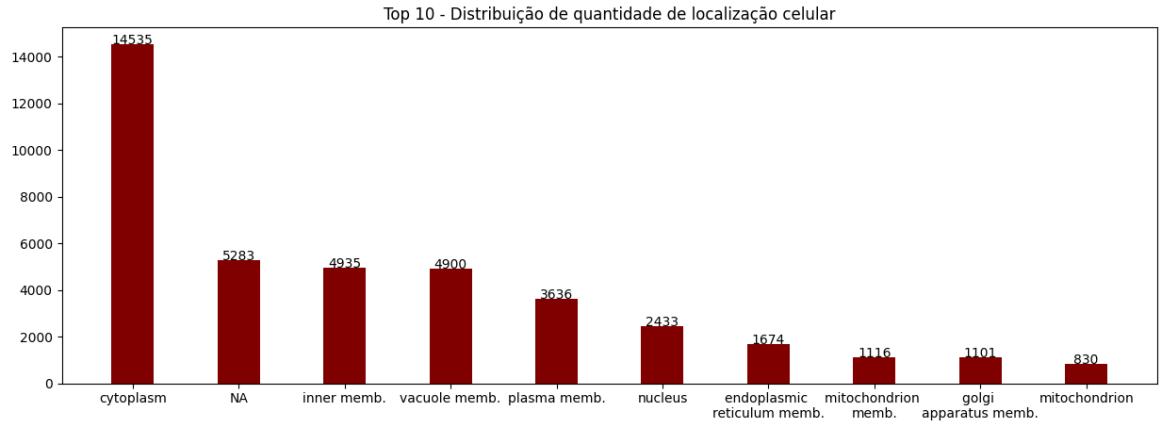


Figura 4.12: 'Top 10' da localização celular das proteínas, no *dataset*.

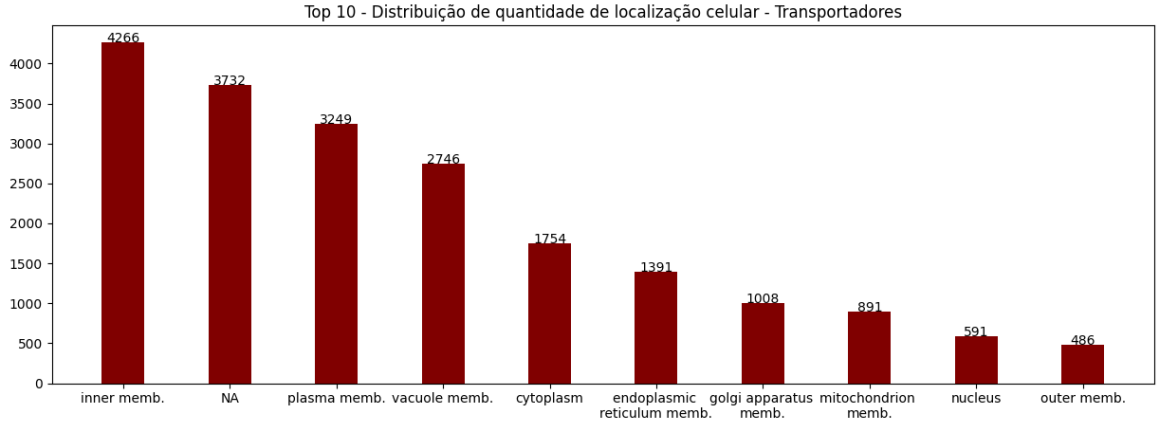


Figura 4.13: 'Top 10' da localização celular das proteínas, no *dataset* positivo.

4.2.2 *Avaliação dos modelos de ML e DL gerados*

Através da implementação dos métodos referidos na subsecção 3.2.4, foi possível obter, sumariamente, o desempenho dos modelos de ML (Tabela 4.2), pela visualização das métricas de estimativa de erro referidas, para cada modelo, respetivamente. Pela análise da mesma, é possivel verificar que o modelo que apresenta um melhor desempenho dos dados de teste é o modelo RF e o RF com a função *ExtraTreeClassifier*.

Tabela 4.2: Valores obtidos das métricas, relativas aos modelos de ML

	Exatidão	**Precisão**	***Recall***	**F1**	***ROC-AUC***	**CMM**
Naive Bayes	0.76	0.74	0.80	0.77	0.82	0.52
Extra Tree Classifier	**0.84**	**0.86**	**0.82**	**0.84**	**0.92**	**0.69**
K nearest neighbours	0.81	0.84	0.76	0.80	0.89	0.62
Logistic Regression	0.83	0.87	0.77	0.82	0.90	0.66
Gradient Boosting	0.76	0.74	0.80	0.77	0.82	0.51
Random Forest	**0.84**	**0.86**	**0.82**	**0.84**	**0.92**	**0.69**
Support Vector Machine	0.81	0.83	0.77	0.80	0.89	0.62

De igual modo, gerou-se um ”sumário”do desempenho dos modelos de DL (Tabela 4.3), nomeadamente das redes CNN, LSTM e híbridas CNN-LSTM, com a apresentação das métricas de estimativa de erro. Pela análise da Tabela 4.3, verifica-se que as redes CNN foram as que obtiveram os melhores resultados nos dados de treino, com resultados próximas as redes híbridas CNN-LSTM e, finalmente, com resultados muito insatisfatórios as redes LSTM.

Tabela 4.3: Matriz de confusão e estimativas de erro, das redes de DL

	Reais Positivos	Reais Negativos		Estimativas de erro
Preditivo Positivo	1128	429	**CNN**	CCM: 0,50 ACC: 0,77 PRE: 0,72 Recall: 0,63 F1: 0,67
Preditivo Negativo	660	2476		
Preditivo Positivo	0	1591	**LSTM**	CCM: 0,00 ACC: 0,66 PRE: 0,00 Recall: 0,00 F1: 0,00
Preditivo Negativo	0	3102		
Preditivo Positivo	881	663	**Híbrida CNN-LSTM**	CCM: 0,50 ACC: 0,79 PRE: 0,57 Recall: 0,73 F1: 0,64
Preditivo Negativo	322	2827		

4.2.3 *Obtenção de resultados pela previsão de transportadores no genoma de E. coli e S. cerevisiae, com modelos ML e DL*

Relativamente ao desempenho na previsão de transportadores, quer dos modelos de ML quer dos modelos DL, verificou-se que estes apresentam, consideravelmente, melhores resultados na previsão de transportadores no genoma da *E. coli* do que no da *S. cerevisiae*. Estes resultados podem ser consultados no https://github.com/ritaconde/Thesis_ML_DL/.

Deste modo, relativamente aos resultados obtidos pelos **modelos de ML** para a *E. coli*, o que apresentou melhores resultados foi o RF com uso da função *ExtraTreeClassifier*, como podemos concluir pelas métricas apresentadas na tabela 4.4, exatidão, precisão, *recall*, F1 e CCM. Por outro lado, pela análise das diversas estimativas de erro, o que apresentou o pior desempenho foi o modelo GB.

Tabela 4.4: Matriz de confusão, dos resultados dos modelos de ML da *E. coli*

	Reais Positivos	Reais Negativos		**Estimativas de erro**
Preditivo Positivo	712	1931	**GB**	CCM: 0,33 ACC: 0,54 PRE: 0,27 Recall: 0,97 F1: 0,42
Preditivo Negativo	24	1575		
Preditivo Positivo	657	1004	**KNN**	CCM: 0,47 ACC: 0,74 PRE: 0,40 Recall: 0,89 F1: 0,54
Preditivo Negativo	79	2502		
Preditivo Positivo	660	1556	**LR**	CCM: 0,34 ACC: 0,62 PRE: 0,30 Recall: 0,90 F1: 0,44
Preditivo Negativo	76	1950		
Preditivo Positivo	612	1399	**NB**	CCM: 0,33 ACC: 0,64 PRE: 0,30 Recall: 0,83 F1: 0,45
Preditivo Negativo	124	2107		
Preditivo Positivo	729	1290	**RF**	CCM: 0,47 ACC: 0,69 PRE: 0,36 Recall: 0,99 F1: 0,53
Preditivo Negativo	7	2216		
Preditivo Positivo	607	1217	**SVM**	CCM: 0,37 ACC: 0,68 PRE: 0,33 Recall: 0,82 F1: 0,47
Preditivo Negativo	129	2289		
Preditivo Positivo	726	985	**Tree**	CCM: 0,54 ACC: 0,77 PRE: 0,42 Recall: 0,99 F1: 0,60
Preditivo Negativo	10	2521		

Já no que diz respeito aos resultados dos modelos de ML para a *S. cerevisiae*, o modelo que mostrou uma melhor *performance* foi, novamente, o RF com uso da função *ExtraTreeClassifier*, como se pode verificar na tabela 4.5. Tal como o sucedido para a *E. coli*, o modelo com piores resultados foi o GB.

Tabela 4.5: Matriz de confusão e estimativas de erro, dos modelos de ML da *S. cerevisiae*

	Reais Positivos	Reais Negativos		**Estimativas de erro**
Preditivo Positivo	486	3599	**GB**	CCM: 0,16 ACC: 0,39 PRE: 0,12 Recall: 0,92 F1: 0,21
Preditivo Negativo	40	1877		
Preditivo Positivo	424	1854	**KNN**	CCM: 0,27 ACC: 0,67 PRE: 0,19 Recall: 0,81 F1: 0,30
Preditivo Negativo	102	3622		
Preditivo Positivo	464	32352	**LR**	CCM: 0,16 ACC: 0,43 PRE: 0,12 Recall: 0,88 F1: 0,21
Preditivo Negativo	62	2124		
Preditivo Positivo	409	2692	**NB**	CCM: 0,16 ACC: 0,53 PRE: 0,13 Recall: 0,77 F1: 0,23
Preditivo Negativo	117	2784		
Preditivo Positivo	517	2629	**RF**	CCM: 0,28 ACC: 0,56 PRE: 0,16 Recall: 0,98 F1: 0,28
Preditivo Negativo	9	2847		
Preditivo Positivo	416	2302	**SVM**	CCM: 0,21 ACC: 0,60 PRE: 0,15 Recall: 0,79 F1: 0,26
Preditivo Negativo	110	3174		
Preditivo Positivo	519	1922	**Tree**	CCM: 0,37 ACC: 0,68 PRE: 0,21 Recall: 0,99 F1: 0,35
Preditivo Negativo	7	3554		

Em relação aos **modelos de DL**, e de acordo com a avaliação das métricas calculadas nas redes LSTM, estas últimas não foram incluídas na geração de resultados para os modelos biológicos selecionados no presente projeto. Assim, apenas foram usadas as redes CNN e híbridas CNN-LSTM para a previsão de transportadores no genoma da *E. coli* e da *S. cerevisiae*.

Como é possível verificar nas Tabelas 4.6 e 4.7, os resultados foram mais promissores na *E. coli*, tal como se tem verificado em todas as abordagens. Adicionalmente, pode-se concluir, que ao contrário dos resultados observados nos dados de teste (Tabela 4.3), as redes que apresentaram desempenho na previsão de transportadores foram as redes híbridas CNN-LSTM, ainda que com resultados relativamente próximos entre as redes CNN e híbridas.

Tabela 4.6: Matriz de confusão e estimativas de erro, dos resultados das redes CNN da *E. coli* e *S. cerevisiae*

	Reais Positivos	Reais Negativos		**Estimativas de erro**
Preditivo Positivo	628	824	***E. coli***	CCM: 0,49 ACC: 0,78 PRE: 0,43 Recall: 0,85 F1: 0,57
Preditivo Negativo	108	2682		
Preditivo Positivo	401	1533	***S. cerevisiae***	CCM: 0,29 ACC: 0,72 PRE: 0,21 Recall: 0,76 F1: 0,33
Preditivo Negativo	125	3943		

Tabela 4.7: Matriz de confusão e estimativas de erro, dos resultados das redes híbridas CNN-LSTM da *E. coli* e *S. cerevisiae*

	Reais Positivos	Reais Negativos		**Estimativas de erro**
Preditivo Positivo	566	433	***E. coli***	CCM: 0,58 ACC: 0,86 PRE: 0,57 Recall: 0,77 F1: 0,65
Preditivo Negativo	170	3073		
Preditivo Positivo	349	837	***S. cerevisiae***	CCM: 0,36 ACC: 0,83 PRE: 0,29 Recall: 0,66 F1: 0,41
Preditivo Negativo	177	4639		

4.3 DISCUSSÃO

No que diz respeito ao ***TransPredict***, confrontando os resultados obtidos no programa construído e os obtidos nas plataformas, nomeadamente, na sequência peptídica do transportador *Accumulation of dyads protein 2* codificada pelo gene *ADY2*, verificou-se que no *output* gerado diretamente no TMHMM e no gerado com o *script* obteve-se, respetivamente, 5 e 6 domínios transmembranares. Para tentar explicar esta diferença, e uma vez que já foi determinada por cristalização da estrutura do seu homólogo bacteriano (SatP) [75], concluiu-se que o resultado obtido com a ferramenta é coincidente com o determinado biologicamente, disponível na bibliografia (6 domínios transmenbranares). Estes resultados distintos devem-se ao facto de o algoritmo base do TMHMM detetar os aminoácidos da proteína, nas diferentes partes da membrana celular - intracelular, extracelular ou transmembranar. Adicionalmente, uma porção da proteína só é considerada um domínio transmembranar, se pelo menos 20 aminoácidos constituírem essa mesma porção. Em suma, a ferramenta desenvolvida apresenta uma maior precisão neste sentido, pelo facto de ter sido restringida, logo à partida, a última condição apresentada.

No entanto, este tipo de algoritmo usado para prever estruturas α-hélice transmembranares, classifica, com relativa frequência, falsos péptidos sinal como α-hélices transmembranares. Por outro lado, os preditores de péptidos sinal têm uma tendência de classificar, erradamente, α-hélices transmembranares como péptidos sinal. Estas "falsas"classificações são uma consequência de ambas as previsões detetarem, principalmente, uma extensão de resíduos hidrofóbicos como principal padrão de reconhecimento. Assim, numa futura implementação da ferramenta, seria pláusivel resolver esta falta de discriminação, através da aplicação de uma topologia conjunta de deteção de α-hélice transmembranares com um preditor de péptidos sinal [35].

Já no que diz respeito às restantes plataformas - *Pred-TMBB*, *Prosite* e CDD -, não se verificaram discrepâncias nos resultados, uma vez que estes foram obtidos através das *API* dos serviços das plataformas, ao contrário do TMHMM que foi implementado com base numa biblioteca.

Verificou-se, pela análise exaustiva dos resultados extraídos em ficheiro *.txt*, que muitas das descrições de domínios conservados surgem de forma abreviada, em siglas, ou até mesmo em hiperligações, o que dificulta a captação de *keywords*. Isto constitui mais uma limitação desta ferramenta, dificultando a sua sensibilidade, sendo necessário a resolução da mesma numa versão definitiva do *Transpredict*.

Relativamente aos resultados obtidos na matriz de confusão, tanto no genoma da *E. coli* como no da *S. cerevisiae*, estes podem não ser os mais exatos. Isto prende-se com o facto, dos valores da matriz serem obtidos por comparação direta com todo o repositório da base de dados TCDB, ou seja, esta pode não estar completamente atualizada. Deste modo, um

transportador que não esteja inserido na base de dados utilizada, vai originar um falso positivo que pode, na realidade ser um verdadeiro positivo.

Com a reavaliação do RF da ferramenta, decidiu-se aplicar penalizações e/ou incrementos às condições que constituem um forte indicador da presença, ou não, de um transportador. As principais penalizações/incrementos aplicadas foi a presença de determinadas *keywords* na descrição extraídas do *Prosite* e CDD. Assim, a qualidade dos domínios conservados influencia em grande medida, as penalizações e incrementos aplicados.

Com a obtenção dos diferentes resultados e a análise da ferramenta, em conjunto com a comparação entre a mesma e as plataformas de base, detetaram-se algumas limitações quer a nível conceptual, quer a nível de usabilidade. Uma das limitações passa pela ferramenta estar intimamente dependente das plataformas, o que, no caso de indisponibilidade dos servidores, pode comprometer a execução do programa. Relativamente à *performance* do *TransPredict*, este tornar-se menos eficiente aquando do processamento do proteoma, constituindo um problema de escalabilidade da ferramenta. No que diz respeito à facilidade de instalação e utilização da ferramenta, esta pode ser pouco intuitiva e de difícil utilização por parte do utilizador comum.

Relativamente aos resultados obtidos nos **modelos de ML**, foi possível observar que os valores das estimativas de erro obtidos na avaliação dos modelos com os dados de teste, pioraram consideravelmente, quando estes foram aplicados nos genomas dos modelos biológicos selecionados. Esta constatação demonstra a importância de testar os modelos em contextos reais, de modo a aferir a real qualidade da abordagem.

Tal como no *TransPredict*, dos dois genomas testados o que obteve melhores resultados foi o genoma de *E. coli*. Neste sentido, e uma vez que os modelos de ML dependem em grande parte das *features* escolhidas, seria importante fazer uma reavaliação das seleção das mesmas e, até possivelmente, adotar uma "bateria"de *features* consoante os modelos biológicos, a sua complexicidade a nível biológico (unicelular, multicelular, procarionte, eucarionte) e as suas características específicas.

No que respeita a questões algorítmicas, os resultados obtidos comprovaram a importância e pertinência do uso de funções específicas de cada modelo de ML. Essa importância pode ser comprovada pela aplicação da função *ExtraTreeClassifier*, nos modelos de RF, que permitiram a melhoria do desempenho dos modelos quer nos dados de teste, quer nos genomas.

Tal como nos valores obtidos da matriz de confusão do *TransPredict*, os valores da matriz de confusão dos diferentes modelos de ML podem estar a ser influenciados negativamente, pela forma como são obtidos diversos valores. Isto porque, estes também foram gerados por comparação direta dos dados disponíveis na base de dados do TCDB.

Tal como nos modelos de ML, os resultados obtidos nos **modelos de DL**, os valores das estimativas de erro obtidos na avaliação dos modelos com os dados de teste, pioraram, no entanto em menor quantidade, relativamente aos valores obtidos na testagem dos genomas dos modelos biológicos da *E. coli* e *S. cerevisiae*. Mais uma vez, destaca-se a importância de testar os modelos em genomas, para verificar a real eficácia da diversas abordagens e, consequentemente, selecionar a mais precisa/exata/robusta.

Relativamente aos resultados obtidos com os diferentes genomas testados, o que teve melhor desempenho foi o genoma da *E. coli*. Neste tipo de modelos, um dos fatores que pode ter contribuido para estes resultados, é o tamanho das sequências.

Outro fator que pode ter contribuído para os resultados obtidos com os algoritmos de DL, pode ser o de os modelos conseguirem determinar mais facilmente padrões de entre os não transportadores face aos transportadores. Isto pelo facto do modelo "aprender"mais com não transportadores, uma vez que o *dataset* negativo tem o dobro do tamanho. Outro facto que pode condicionar os resultados obtidos nos dados de treino dos modelos DL, pode estar relacionado com o *bias*. Isto porque, uma vez que os dados são selecionados aleatoriamente e a escolha das sequências pode nem sempre ser a melhor.

Aquando do processo de treino, verificou-se, ainda, um fenómeno de *overfitting* nos modelos de DL, em que se constatou, a cada iteração, um *bias* com dados de treino. Isto fez com que a rede não conseguisse generalizar, quando recebe um *input* diferente daquilo que está "habituado"a receber.

Como foi possível percecionar ao longo da elaboração dos modelos de DL, os parâmetros, as *layers* e as seleção das diferentes redes nas diferentes camadas do algoritmo (este último, no caso das redes híbridas), determinam em grande escala a qualidade dos resultados a obter. Deste modo, seria vantajoso realizar mais tentativas de diferentes "conjugações", de modo a obter melhores resultados.

Em suma, como é possível concluir pela análise dos resultados apresentados nas subsecções 4.1 e 4.2, os modelos biológicos que melhores resultados apresentaram nas diferentes abordagens foi o modelo da *E. coli*. Estes resultados podem ser explicados pelo facto de esta ter um genoma menor, um menor grau de complexidade dos mecanismos biológicos e pelo facto de na *S. cerevisiae* existir uma maior variedade de N e C-terminais de maiores dimensões, bem como *loops* longos entre segmentos transmembranares, que dificultam a distinção dos transportadores.

Adicionalmente, após a análise dos resultados dos genomas da *E. coli* e *S. cerevisiae* no *TransPredict*, seria pertinente complementar o estudo com genomas de outros modelos biológicos, com maior diferenciação e mais complexos, bem como modelos não tão bem caracterizado, em todas as abordagens.

O *TransPredict* e as ferramentas à base de ML e DL, apresentadas nesta dissertação, são uma mais valia pois não existem, atualmente, nenhuma outra ferramenta robusta que permita fazer um estudo tão abrangente. Pelo facto dos transportadores, como o próprio nome indica, transportarem as mais variadas moléculas, desde fármacos a metabolitos biológicos [76], a criação de uma ferramenta deste tipo torna-se imperativa e indispensável, no âmbito das diversas áreas da investigação.

5

CONCLUSÕES E TRABALHO FUTURO

No capítulo final, serão apresentadas as conclusões finais das metodologias apresentadas, bem como algumas limitações detetadas ao longo do desenvolvimento do projeto.

Serão, também, apresentados alguns pontos a desenvolver no futuro, de modo a melhorar e a otimizar as ferramentas desenvolvidas e apresentadas no presente trabalho.

5.1 CONCLUSÕES

Este projeto surgiu na sequência da proposta da construção de uma ferramenta que permitisse obter um *output* capaz de incorporar a informação obtida das diferentes plataformas disponíveis *online* e correr um proteoma total, sem ter limite de tamanho de ficheiro. Apesar de serem detetadas algumas limitações e havendo melhorias a implementar, concluiu-se que a tarefa foi terminada com sucesso, uma vez que se verificou que a mesma deteta transportadores. Verificou-se, ainda, que este tipo de abordagem permite costumizar mais especificamente para diversos cenários os resultados obtidos e, dessa forma, consegue ser mais preciso para a situação em estudo.

O segundo ponto fulcral deste trabalho consistia na validação do *TransPredict*, pela implementação de ferramentas com base em modelos de ML e DL. Assim, verificou-se que a ferramenta principal, o *TransPredict*, é uma ferramenta mais robusta na previsão de transportadores em procariontes, com a qual poderemos obter resultados promissores. Atualmente, o *TransPredict* tem a vantagem de um utilizador não necessitar de tarefas intermédias para obter o resultado pretendido, ao contrário das outras duas abordagens. Apresenta, no entanto, algumas desvantagens como o tempo de geração de resultados (principalmente em relação aos modelos DL) e a estreita dependência com os servidores *online*.

Em suma, este projeto tem potencial para ser mais explorado e desenvolvido, podendo complementar, ainda mais, o conhecimento e compreensão dos transportadores.

5.2 TRABALHO FUTURO

De forma a complementar os dados e informações obtidas com o desenvolvimento da **ferramenta *TransPredict***, sugere-se a abordagem dos seguintes pontos:

- Verificar homologias, podendo mesmo integrá-las na ferramenta;
- Incorporar outras plataformas, tais como o TCDB e *HHpred*, para comparar homologias e tentar classificar o tipo de transportador;
- Fazer a comparação entre sequências dadas como *input*, diretamente no resultado final;
- Obter as matrizes de confusão, para os genomas em estudo, a partir de bases de dados mais abrangentes, uma vez que se pretende detetar possíveis novos transportadores com a ferramenta desenvolvida;
- Automatizar o processo de validação/verificação dos resultados obtidos, a partir da integração de uma base de dados de transportadores fidedigna;
- Melhorar a procura de *keywords*, essencialmente ao nível do CDD, incluindo a pesquisa ao nível de descrição das superfamílias de proteínas, do motivo detetado;
- Melhorar a interface para o utilizador, tornando-a mais apelativa e intuitiva.

Já no que diz respeito à ferramenta desenvolvida, tendo por base **modelos de ML**, existem outros pontos que podem ser melhorados e implementados:

- Utilizar um *dataset* não balanceado, isto é, criar um *dataset* com mais casos positivos que negativos, pois no contexto real a quantidade de transportadores que são codificados pelos genoma, constituem apenas 10% do mesmo;
- Aplicar métodos de *Emsemble*, de modo a melhorar o desempenho de todo o sistema;
- Testar a utilização de outras *features*, que possam ser mais adequadas e crucias no treino dos modelos de ML;
- Aplicar um modelo híbrido entre ML e DL, isto é, tentar incorporar as *features* geradas para processamento dos modelos de ML e aplicar sobre algoritmos de geração de modelos de DL;
- Melhorar a interface para o utilizador, tornando-a *user friendly* e utilizável por qualquer pessoa.

Em jeito de conclusão, relativamente à ferramenta desenvolvida, tendo por base **modelos de DL**, poderiam ser testadas outras redes neuronais, nomeadamente outras conjugações

para as redes híbridas, e até mesmo testar diferentes parâmetros. Outra melhoria poderia passar por testar diferentes parâmetros e conjugações dos mesmos, para a criação dos algoritmos de DL. Isto permitirá encontrar o *set* de parâmetros que melhor se ajuste para levar à otimização da predição de transportadores, objetivo final do desenvolvimento destas ferramentas. Finalmente, poder-se-á garantir a igual percentagem de casos positivos entre os *datasets* de treino e teste.

BIBLIOGRAFIA

[1] https://support.proteomesoftware.com/hc/en-us/articles/115002058583-Introduction-to-Proteomics#What%20is%20a%20Proteome? Acedido: 2021-01-31.

[2] M. Sousa-Silva, D. Vieira, P. Soares, M. Casal, and I. Soares-Silva, "Expanding the knowledge on the skillful yeast cyberlindnera jadinii," *Journal of Fungi*, vol. 7, no. 1, p. 36, 2021.

[3] I. Soares-Silva, D. Ribas, M. Sousa-Silva, J. Azevedo-Silva, T. Rendulić, and M. Casal, "Membrane transporters in the bioproduction of organic acids: state of the art and future perspectives for industrial applications," *FEMS Microbiology Letters*, vol. 367, no. 15, p. fnaa118, 2020.

[4] www.britannica.com/science/protein. Acedido: 2021-01-31.

[5] A. Krogh, B. Larsson, G. Von Heijne, and E. L. Sonnhammer, "Predicting transmembrane protein topology with a hidden markov model: application to complete genomes," *Journal of molecular biology*, vol. 305, no. 3, pp. 567–580, 2001.

[6] P. G. Bagos, T. D. Liakopoulos, I. C. Spyropoulos, and S. J. Hamodrakas, "Pred-tmbb: a web server for predicting the topology of β-barrel outer membrane proteins," *Nucleic acids research*, vol. 32, no. suppl_2, pp. W400–W404, 2004.

[7] N. Hulo, A. Bairoch, V. Bulliard, L. Cerutti, E. De Castro, P. S. Langendijk-Genevaux, M. Pagni, and C. J. Sigrist, "The prosite database," *Nucleic acids research*, vol. 34, no. suppl_1, pp. D227–D230, 2006.

[8] A. Marchler-Bauer, M. K. Derbyshire, N. R. Gonzales, S. Lu, F. Chitsaz, L. Y. Geer, R. C. Geer, J. He, M. Gwadz, D. I. Hurwitz, *et al.*, "Cdd: Ncbi's conserved domain database," *Nucleic acids research*, vol. 43, no. D1, pp. D222–D226, 2015.

[9] I. Korf, M. Yandell, and J. Bedell, *Blast*. "O'Reilly Media, Inc.", 2003.

[10] S. Kumar, M. Nei, J. Dudley, and K. Tamura, "MEGA: A biologist-centric software for evolutionary analysis of DNA and protein sequences," *Briefings in Bioinformatics*, vol. 9, pp. 299–306, 04 2008.

[11] K. Katoh, K. Misawa, K. Kuma, and T. Miyata, "MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform," *Nucleic Acids Research*, vol. 30, pp. 3059–3066, 07 2002.

[12] N. K. Mishra, J. Chang, and P. X. Zhao, "Prediction of membrane transport proteins and their substrate specificities using primary sequence information," *PloS one*, vol. 9, no. 6, p. e100278, 2014.

[13] H. Li, X. Dai, and X. Zhao, "A nearest neighbor approach for automated transporter prediction and categorization from protein sequences," *Bioinformatics*, vol. 24, no. 9, pp. 1129–1136, 2008.

[14] M. M. Gromiha and Y. Yabuki, "Functional discrimination of membrane proteins using machine learning techniques," *BMC bioinformatics*, vol. 9, no. 1, pp. 1–8, 2008.

[15] H. Li, V. A. Benedito, M. K. Udvardi, and P. X. Zhao, "Transporttp: a two-phase classification approach for membrane transporter prediction and characterization," *BMC bioinformatics*, vol. 10, no. 1, pp. 1–13, 2009.

[16] www.nature.com/scitable/topicpage/transcriptome-connecting-the-genome-to-gene-function-605/. Acedido: 2021-01-31.

[17] A. Quintas, A. P. Freire, and M. J. Halpern, "Bioquímica, organização molecular da vida," *Lisboa: Lidel*, 2008.

[18] https://medlineplus.gov/genetics/understanding/howgeneswork/protein/. Acedido: 2021-01-31.

[19] http://labs.icb.ufmg.br/lbcd/grupo1/pag3.html. Acedido: 2021-01-31.

[20] O. Vit and J. Petrak, "Integral membrane proteins in proteomics. how to break open the black box?," *Journal of proteomics*, vol. 153, pp. 8–20, 2017.

[21] M. P. Krebs, S. M. Noorwez, R. Malhotra, and S. Kaushal, "Quality control of integral membrane proteins," *Trends in biochemical sciences*, vol. 29, no. 12, pp. 648–655, 2004.

[22] A. Conde, G. Diallinas, F. Chaumont, M. Chaves, and H. Gerós, "Transporters, channels, or simple diffusion? dogmas, atypical roles and complexity in transport systems," *The international journal of biochemistry & cell biology*, vol. 42, no. 6, pp. 857–868, 2010.

[23] J. H. Pinho, M. A. G. Quaresma, and S. I. Martins, "Biologia molecular da célula," *FMV-UTL)*, 2013.

[24] M. Riley, T. Abe, M. B. Arnaud, M. K. Berlyn, F. R. Blattner, R. R. Chaudhuri, J. D. Glasner, T. Horiuchi, I. M. Keseler, T. Kosuge, *et al.*, "Escherichia coli k-12: a cooperatively developed annotation snapshot—2005," *Nucleic acids research*, vol. 34, no. 1, pp. 1–9, 2006.

[25] S. R. Engel, F. S. Dietrich, D. G. Fisk, G. Binkley, R. Balakrishnan, M. C. Costanzo, and e. a. Dwight, "The Reference Genome Sequence of Saccharomyces cerevisiae: Then and Now," *G3 Genes—Genomes—Genetics*, vol. 4, pp. 389–398, 03 2014.

[26] www.ncbi.nlm.nih.gov/bioproject/PRJNA225/. Acedido: 2021-02-14.

[27] F. R. Blattner, G. Plunkett, C. A. Bloch, N. T. Perna, and e. a. Burland, "The complete genome sequence of escherichia coli k-12," *Science*, vol. 277, no. 5331, pp. 1453–1462, 1997.

[28] www.ncbi.nlm.nih.gov/bioproject/PRJNA43747/. Acedido: 2021-02-14.

[29] J. Hou, L. Acharya, D. Zhu, and J. Cheng, "An overview of bioinformatics methods for modeling biological pathways in yeast," *Briefings in functional genomics*, vol. 15, no. 2, pp. 95–108, 2016.

[30] J. F. Nijkamp, M. van den Broek, E. Datema, and e. a. de Kok, "De novo sequencing, assembly and analysis of the genome of the laboratory strain saccharomyces cerevisiae cen. pk113-7d, a model for modern industrial biotechnology," *Microbial cell factories*, vol. 11, no. 1, pp. 1–17, 2012.

[31] E. A. Johnson and C. Echavarri-Erasun, "Yeast biotechnology," in *The yeasts*, pp. 21–44, Elsevier, 2011.

[32] M. H. Saier Jr, C. V. Tran, and R. D. Barabote, "Tcdb: the transporter classification database for membrane transport protein analyses and information," *Nucleic acids research*, vol. 34, no. suppl_1, pp. D181–D186, 2006.

[33] M. H. Saier Jr, "Families of transmembrane sugar transport proteins," *Molecular Microbiology*, vol. 35, no. 4, pp. 699–710, 2002.

[34] A. Bairoch and R. Apweiler, "The swiss-prot protein sequence database and its supplement trembl in 2000," *Nucleic acids research*, vol. 28, no. 1, pp. 45–48, 2000.

[35] L. Käll, A. Krogh, and E. L. Sonnhammer, "A combined transmembrane topology and signal peptide prediction method," *Journal of Molecular Biology*, vol. 338, no. 5, pp. 1027–1036, 2004.

[36] F. S. Berven, K. Flikka, H. B. Jensen, and I. Eidhammer, "Bomp: a program to predict integral β-barrel outer membrane proteins encoded within genomes of gram-negative bacteria," *Nucleic acids research*, vol. 32, no. suppl_2, pp. W394–W399, 2004.

[37] T. Goldberg, M. Hecht, T. Hamp, T. Karl, G. Yachdav, N. Ahmed, U. Altermann, P. Angerer, S. Ansorge, K. Balasz, *et al.*, "Loctree3 prediction of localization," *Nucleic acids research*, vol. 42, no. W1, pp. W350–W355, 2014.

[38] https://biopython.org/. Acedido: 2021-12-05.

[39] https://scikit-learn.org/. Acedido: 2021-12-05.

[40] G. Hackeling, *Mastering Machine Learning with scikit-learn*. Packt Publishing Ltd, 2017.

[41] www.tensorflow.org/. Acedido: 2021-12-08.

[42] G. Zaccone, M. R. Karim, and A. Menshawy, *Deep learning with TensorFlow*. Packt Publishing Ltd, 2017.

[43] A. M. Sequeira, D. Lousa, and M. Rocha, "Propythia: A python package for protein classification based on machine and deep learning," *Neurocomputing*, 2021.

[44] A. M. Sequeira, D. Lousa, and M. Rocha, "Propythia: A python automated platform for the classification of proteins using machine learning," in *International Conference on Practical Applications of Computational Biology & Bioinformatics*, pp. 32–41, Springer, 2020.

[45] G. Bonaccorso, *Machine learning algorithms*. Packt Publishing Ltd, 2017.

[46] E. Y. Lee, B. M. Fulan, G. C. L. Wong, and A. L. Ferguson, "Mapping membrane activity in undiscovered peptide sequence space using machine learning," *Proceedings of the National Academy of Sciences*, vol. 113, no. 48, pp. 13588–13593, 2016.

[47] X.-D. Zhang, *Machine Learning*, pp. 223–440. Singapore: Springer Singapore, 2020.

[48] Y. Qi, "Random forest for bioinformatics," in *Ensemble machine learning*, pp. 307–323, Springer, 2012.

[49] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html. Acedido: 2021-12-09.

[50] J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002. Nonlinear Methods and Data Mining.

[51] S. Dreiseitl and L. Ohno-Machado, "Logistic regression and artificial neural network classification models: a methodology review," *Journal of biomedical informatics*, vol. 35, no. 5-6, pp. 352–359, 2002.

[52] A. Dey, "Machine learning algorithms: a review," *International Journal of Computer Science and Information Technologies*, vol. 7, no. 3, pp. 1174–1179, 2016.

[53] W. Lou, X. Wang, F. Chen, Y. Chen, B. Jiang, and H. Zhang, "Sequence based prediction of dna-binding proteins based on hybrid feature selection using random forest and gaussian naive bayes," *PloS one*, vol. 9, no. 1, p. e86703, 2014.

[54] www.javatpoint.com. Acedido: 2021-12-19.

[55] C. Halimu, A. Kasem, and S. S. Newaz, "Empirical comparison of area under roc curve (auc) and mathew correlation coefficient (mcc) for evaluating machine learning algorithms on imbalanced datasets for binary classification," in *Proceedings of the 3rd international conference on machine learning and soft computing*, pp. 1–6, 2019.

[56] www.ibm.com/cloud/learn/deep-learning. Acedido: 2021-12-09.

[57] P. P. Shinde and S. Shah, "A review of machine learning and deep learning applications," in *2018 Fourth international conference on computing communication control and automation (ICCUBEA)*, pp. 1–6, IEEE, 2018.

[58] V. I. Jurtz, A. R. Johansen, M. Nielsen, J. J. Almagro Armenteros, H. Nielsen, C. K. Sønderby, O. Winther, and S. K. Sønderby, "An introduction to deep learning on biological sequence data: examples and solutions," *Bioinformatics*, vol. 33, pp. 3685–3690, 08 2017.

[59] M. L. Bileschi, D. Belanger, D. Bryant, T. Sanderson, B. Carter, D. Sculley, M. A. DePristo, and L. J. Colwell, "Using deep learning to annotate the protein universe," *BioRxiv*, p. 626507, 2019.

[60] A. Voulodimos, N. Doulamis, G. Bebis, and T. Stathaki, "Recent developments in deep learning for engineering applications," 2018.

[61] C. C. Aggarwal *et al.*, "Neural networks and deep learning," *Springer*, vol. 10, pp. 978–3, 2018.

[62] https://towardsdatascience.com/protein-sequence-classification-99c80d0ad2df. Acedido: 2021-12-09.

[63] https://towardsdatascience.com/deep-hybrid-learning-a-fusion-of-conventional-ml-with-state-of-the-art-dl-cb43887fe14. Acedido: 2022-01-26.

[64] M. Suganthi and J. G. R. Sathiaseelan, "An exploratory of hybrid techniques on deep learning for image classification," in *2020 4th International Conference on Computer, Communication and Signal Processing (ICCCSP)*, pp. 1–4, 2020.

[65] https://zhanglab.ccmb.med.umich.edu/FASTA/ (acedido a 8 de julho de 2020). Acedido: 2020-07-08.

[66] http://genetics.bwh.harvard.edu/pph/FASTA.html (acedido a 8 de julho de 2020). Acedido: 2020-07-08.

[67] https://biopython.org/DIST/docs/api/Bio.SeqIO-module.html (acedido a 8 de julho de 2020). Acedido: 2020-07-08.

[68] https://github.com/dansondergaard/tmhmm.py (acedido a 8 de julho de 2020). Acedido: 2020-07-08.

[69] B.-J. Yoon, "Hidden markov models and their applications in biological sequence analysis," *Current genomics*, vol. 10, no. 6, pp. 402–415, 2009.

[70] D. Jurafsky and J. Martin, "Speech and language processing-chapter 8 hidden markov models," *stanford*, 2014.

[71] https://requests.readthedocs.io/en/master/ (acedido a 8 de julho de 2020). Acedido: 2020-07-08.

[72] https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/. Acedido: 2021-01-15.

[73] www.uniprot.org/uniprot/P36035 (acedido a 18 de julho de 2020). Acedido: 2020-07-18.

[74] www.rcsb.org/structure/1F88 (acedido a 18 de julho de 2020). Acedido: 2020-07-18.

[75] B. Qiu, B. Xia, Q. Zhou, Y. Lu, M. He, K. Hasegawa, Z. Ma, F. Zhang, L. Gu, Q. Mao, *et al.*, "Succinate-acetate permease from citrobacter koseri is an anion channel that unidirectionally translocates acetate," *Cell research*, vol. 28, no. 6, pp. 644–654, 2018.

[76] L. M. Veenhoff, E. H. Heuberger, and B. Poolman, "Quaternary structure and function of transport proteins," *Trends in biochemical sciences*, vol. 27, no. 5, pp. 242–249, 2002.