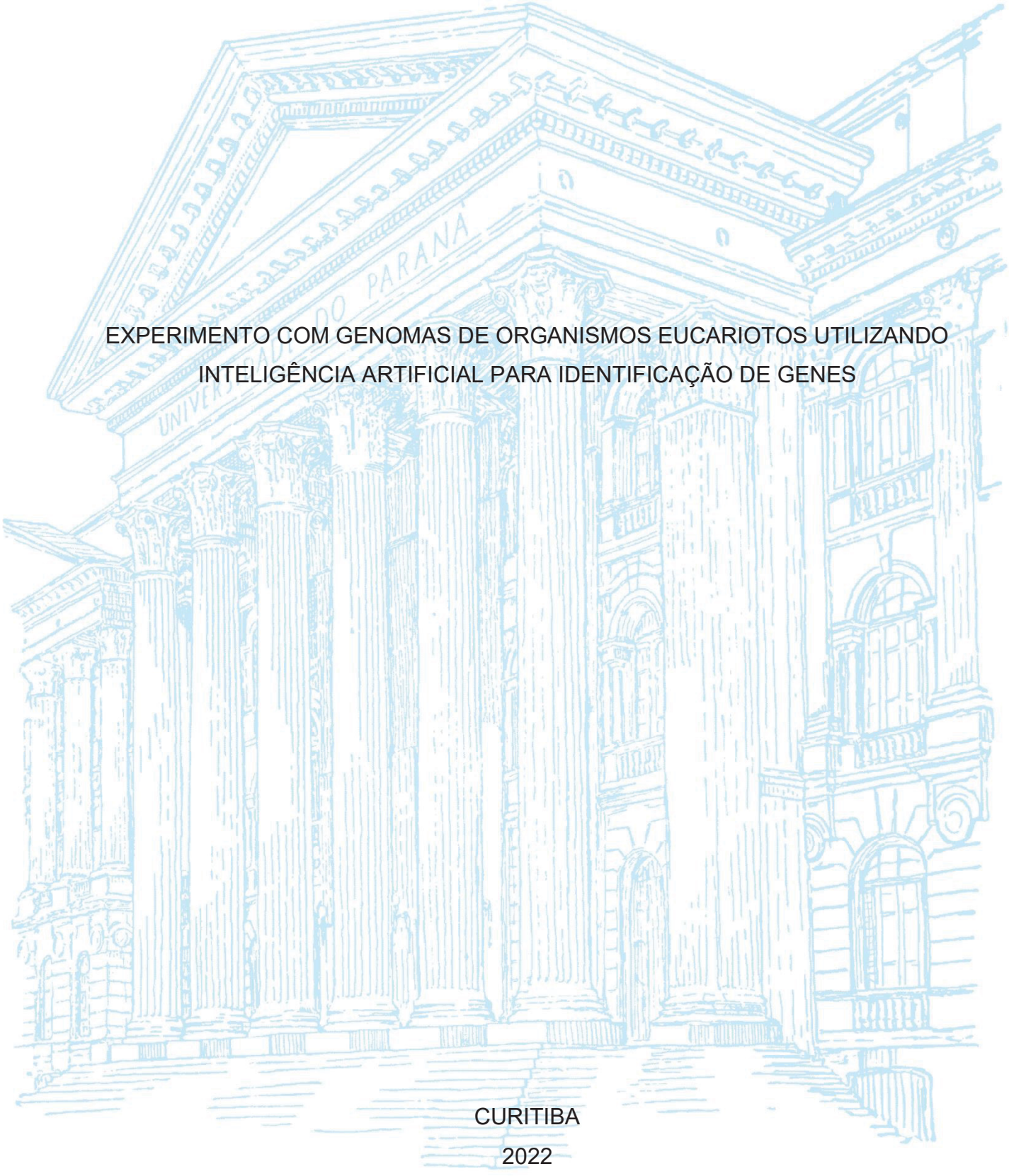
UNIVERSIDADE FEDERAL DO PARANÁ

LUÍS SÉRGIO DA SILVA JOPPERT



EXPERIMENTO COM GENOMAS DE ORGANISMOS EUCARIOTOS UTILIZANDO INTELIGÊNCIA ARTIFICIAL PARA IDENTIFICAÇÃO DE GENES

CURITIBA

2022

LUÍS SÉRGIO DA SILVA JOPPERT

EXPERIMENTO COM GENOMAS DE ORGANISMOS EUCARIOTOS UTILIZANDO INTELIGÊNCIA ARTIFICIAL PARA IDENTIFICAÇÃO DE GENES

Trabalho de Conclusão de Curso apresentado ao curso de Especialização em Inteligência Artificial Aplicada, Setor de Educação Profissional e Tecnológica, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Especialista em Inteligência Artificial Aplicada.

Orientador: Prof. Dr. Jaime Wojciechowski

CURITIBA

2022

MINISTÉRIO DA EDUCAÇÃO
SETOR DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
CURSO DE PÓS-GRADUAÇÃO INTELIGÊNCIA ARTIFICIAL APLICADA - 40001016348E1

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação INTELIGÊNCIA ARTIFICIAL APLICADA da Universidade Federal do Paraná foram convocados para realizar a arguição da Monografia de Especialização de **LUIS SERGIO DA SILVA JOPPERT** intitulada: **EXPERIMENTO COM GENOMAS DE ORGANISMOS EUCARIOTOS UTILIZANDO INTELIGENCIA ARTIFICIAL PARA IDENTIFICACAO DE GENES**, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua aprovação no rito de defesa.

A outorga do título de especialista está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 09 de Novembro de 2022.

JAIME WOJCIECHOWSKI
Presidente da Banca Examinadora

DIEVAL GUIZELINI
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Experimento com Genomas de Organismos Eucariotos Utilizando Inteligência Artificial para Identificação de Genes

1st Luís Sérgio da Silva Joppert
Setor de Educação Profissional e Tecnológica
Universidade Federal do Paraná
Curitiba, Brasil
luisjoppert@ufpr.br

2nd Jaime Wojciechowski
Setor de Educação Profissional e Tecnológica
Universidade Federal do Paraná
Curitiba, Brasil
jaimewo@ufpr.br

***Abstract*—Dentro da bioinformática até o momento, a maior parte dos esforços nas extrações e comparações de anotações de genomas são aplicadas em organismos procariotos, visto que estes possuem uma estrutura menor e mais simples em relação aos eucariotos. Além da complexidade, o processo de anotação é muito custoso computacionalmente, principalmente se utilizado técnicas de força bruta, sendo que posteriormente essas anotações ainda precisarão ser comparadas com outras para validação. Todavia, atualmente há abordagens amplamente utilizadas que podem obter ótimos resultados utilizando a inteligência artificial, a qual pode gerar bons modelos que irão auxiliar na futura diminuição do custo computacional e na melhora de resultados. Esta pesquisa utiliza-se de ferramentas de reconhecimento de padrões aliadas a mineração de dados para classificar se sequencias de DNA de organismos eucariotos representam ou não regiões gênicas, visando auxiliar na anotação genômica.**

***Index Terms*—predição, gene, bioinformática, genoma, dna**

***Abstract*—Within bioinformatics so far, most of the efforts in extracting and comparing genome annotations are applied to prokaryotic organisms, since they have a smaller and simpler structure compared to eukaryotes. In addition to the complexity, the annotation process is very computationally expensive, especially if brute force techniques are used, and later these annotations will still need to be compared with others for validation. However, currently there are widely used approaches that can obtain great results using artificial intelligence, which can generate good models that will help in the future decrease of the computational cost and in the improvement of results. This research uses pattern recognition tools combined with data mining to classify whether or not a DNA sequences from eukaryotic organisms are gene regions, in order to assist in genomic annotation.**

***Index Terms*—prediction, gene, bioinformatics, genome, dna**

I. Introdução

A bioinformática engloba inúmeros temas tanto dentro da computação quanto dentro da biologia e afins. Segundo Verli [1] ela pode ser dividida em duas vertentes. A primeira sendo mais clássica, que tem como foco o estudo e nos problemas relacionados a sequências de nucleotídeos e aminoácidos, já a segunda é uma ramificação que estuda as questões biológicas de um ponto de vista tridimensional. Independente da vertente, as duas tiveram seu início na década de 1950, sendo uma área do conhecimento ainda muito jovem em relação a outras mais fundamentais como a biologia e a química.

A bioinformática é uma área que utiliza da computação para auxiliar na resposta de questões biológicas, sendo que um pesquisador terá uma grande vantagem em utilizar grandes bases de dados como apoio para as conclusões biológicas [2].

O genoma é o todo o conjunto de DNA que um ser vivo possui, sendo que o DNA é formado por um sequência de moléculas chamadas de nucleotídeos [3]. Uma das linhas de estudo dos genomas é a identificação ou anotação genômica. A anotação é um conjunto de ações que tem como objetivo delimitar dentro da sequência genômica possíveis genes e predizer sua função baseando-se em outras sequências [1].

Para que se possa desenvolver estudos tanto funcionais quanto estruturais do DNA, é necessária uma técnica que permita revelar as suas sequências de suas bases nucleicas, esta que se chama sequenciamento [4]. Entretanto apenas 13% dos sequenciamentos de genoma de organismos procarióticos são concluídos, sendo que dentro dos bancos de dados públicos são depositados rascunhos do genoma na forma fragmentada de contigs, estes que por sua vez podem ter perda de informações [5].

Alguns softwares auxiliam na identificação de áreas para fazer anotações, sendo um exemplo o Artemis. Este programa possibilita a visualização tanto do genoma completo como de possíveis anotações permitindo que seja possível analisar as sequências [5].

O reconhecimento de padrões, diz respeito à classificação ou descrição de dados por meio de padrões observados nos mesmos [6], sendo que este artigo tem como objetivo utilizar o reconhecimento de padrões para que as análises manuais para não serem muito custosas.

A. Objetivos Específicos

Como objetivo principal este trabalho visa desenvolver um software que auxilie na identificação de genes para facilitar a etapa das anotações do genoma de organismos eucariotos, utilizando reconhecimento de padrões.

Os objetivos específicos se dão em:

- Analisar sequências obtidas de bancos de dados de genes públicos e confiáveis como por exemplo, o NCBI GenBank RefSeq;
- Desenvolver um método de extração de características
- Levantar técnicas que auxiliem na identificação dos genes;
- Comparar técnicas e modelos aplicados ao genoma;
- Criar de reconhecimento de padrões que classifique se um trecho de DNA é ou não uma região gênica de um organismo eucarioto;
- Validar o modelo baseado em anotações previamente feitas das bases utilizada para análise;
- Analisar a performance dos resultados gerados ao compará-los com outros programas.

B. *Justificativa*

Tradicionalmente a anotação do genoma é alcançada de forma manual, e embora a curadoria manual possa atingir altos graus de precisão, ela não pode acompanhar o volume de sequenciamentos sendo produzido na atualidade. Com isso, as soluções de bioinformática são cada vez mais necessárias para desenvolver técnicas de anotação automática para apoiar e complementar o processo de curadoria manual [7].

A escala do sequenciamento genômico de eucariotos é impressionante, e embora os genes codificadores de proteínas seja de suma importância, esse objetivo ainda representa um grande desafio [8].

Levando em consideração a importância da anotação genômica e o desafio de se trabalhar com os genes de organismos eucariotos devido a grande quantidade de dados, esta pesquisa visa facilitar esse processo por meio do reconhecimento de padrões. Em auxílio as pesquisas desse tema, um dos principais desenvolvimentos nos últimos anos foi a implementação de anotações de genomas automáticas [7].

As técnicas de reconhecimento de padrões e aprendizado de máquina podem ser muito úteis na área da bioinformática. Os mais recentes algoritmos de predição de genes usam técnicas de aprendizado de máquina, principalmente modelos de ganho probabilísticos (como modelos ocultos de Markov) para obter uma melhor previsão de resultados, sendo o GRAIL, GeneScan, Glim-mer, GeneMark.hmm, MZEF, GeneFinder alguns exemplos de softwares que utilizam dessas técnicas [9]. Alguns dos softwares já possuem uma grande taxa de acertos na predição de genes, todavia utilizam-se de um grande custo computacional [10].

Portanto dada a grande utilização das técnicas citadas e os bons resultados gerados por elas, esta pesquisa visa melhoras os processos de anotação automática de genomas em organismos eucariotos.

II. Fundamentação Teórica

Neste capítulo serão apresentados os conceitos utilizados ao longo desse trabalho.

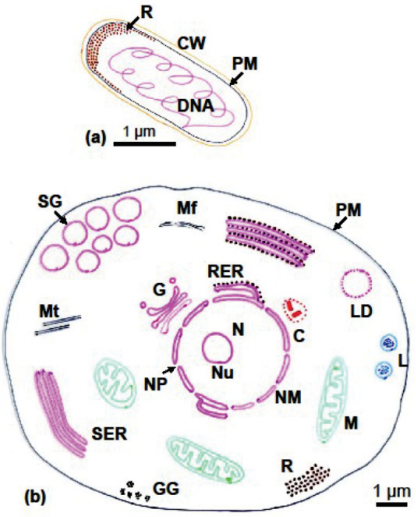


Fig. 1. A. Diagramas esquemáticos de uma célula procariótica (colon bacillus). B. Uma célula eucariótica (pâncreas de rato) [12]

A. *Diferença entre células eucariontes e procariontes*

As unidades funcionais e estruturais dos organismos são denominados células e todas elas compartilham algumas características em comum [11]. Segundo Yamaguchi existem apenas dois tipos conhecidos de célula viva: a procariótica e a eucariótica. As células procariontes fazem parte das bactérias e arqueias e são geralmente apenas alguns micrômetros em tamanho, tendo estruturas celulares simples, incluindo citoplasma, ribossomos, um membrana plasmática e uma parede celular. Já as células eucariontes são muito mais complexas e estão tanto em organismos unicelulares quanto multicelulares, como por exemplo, seres humanos, plantas, fungos e protistas. As células eucarióticas também têm um núcleo fechado por uma membrana dupla, retícula endoplasmática, aparelhos de Golgi, peroxissomos, lisossomos endossomos e inúmeros tipos de vacúolos. Além disso, as células eucarióticas têm ou dois tipos distintos de organelas que contêm seu próprio DNA: mitocôndrias e cloroplastos. Por fim essas células também têm vários tipos de estruturas do citoesqueleto como centríolos, microtúbulos e microfilamentos como mostra a Figura 1 [12].

B. *Ácido Desoxirribonucleico - DNA*

O DNA molécula responsável por codificar a informação genética. Essa molécula possui dois filamentos compostos de 4 bases nitrogenadas que são responsáveis por armazenar a informação genética, semelhante a um computador que armazena as informações de forma binária. O DNA se encontra na forma de uma dupla-hélice, onde os dois filamentos compostos grupos de açúcar e fosfato são enrolados entre si produzindo um espiral. Este espiral pode ser comparado a uma escada torcida. Os degraus dessa escada são compostos por 4 bases: adenina (A), timina (T), guanina (G) e citosina (C) como está na Figura 2. Essas bases são voltadas para o centro da hélice e são ligadas entre si por uma ponte de hidrogênio. A

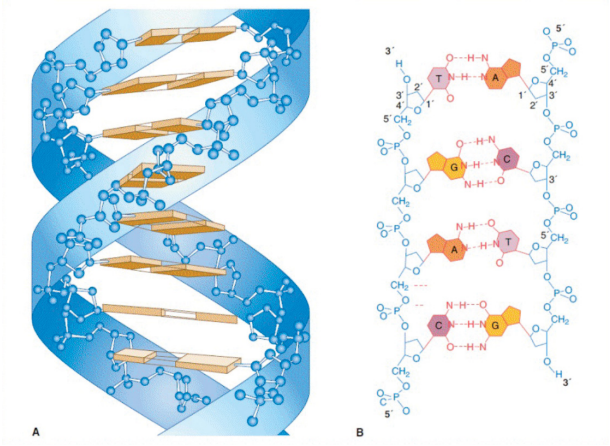


Fig. 2. A. Modelo dupla-hélice do DNA representando os filamentos de açúcar-fosfato em azul ao lado e os pares de base em marrom ao centro. B. Modelo plano do DNA mostrando as ligações adenina e timina (AT) e guanina citosina (GC). As fileiras em vermelho ao centro representam as pontes de hidrogênio das ligações [13]

ligação ocorrerá por uma ligação dupla de hidrogênio entre a adenina e timina (AT) e uma ligação tripla entre a guanina e a citosina (CG). Algumas espécies podem ter genomas grandes e outras genomas pequenos. [13].

C. Ácido Ribonucleico - RNA

Além do DNA outra estrutura presente nos organismos vivos é o RNA. Este é uma macromolécula que possui como função da manutenção e transmissão das informações do genoma (Figura 3), além da função de catálise durante a transcrição e tradução gênica. O RNA e o DNA possuem algumas semelhanças. Por exemplo dos dois são formados pela ligação de hidrogênio nas bases químicas, contudo o RNA possui uma base diferente do DNA. Como o açúcar na cadeia do RNA é a ribose, a base que seria uma timina, se torna uma uracila, ficando da seguinte forma: adenina (A), uracila (U), guanina (G) e citosina (C), sendo que as ligações prevalecem com as devidas modificações (adenina liga com uracila e guanina liga com citosina). Outra diferença é que o RNA comumente tem apenas uma cadeia, apesar de poder formar cadeias duplas por pareamentos internamente [14].

D. Tipos de RNA

Existem diversas classes de RNA nas células de um organismo, esses que são responsáveis por algumas funções como síntese de proteínas, no processamento dos RNAs, que geralmente são gerados como percursores não funcionais e na regulação da expressão gênica. As três classes principais de RNA são o RNA Mensageiro (mRNA), o RNA ribossômico (rRNA) e RNA transportador (tRNA) [14].

1) RNA Mensageiro: O papel do mRNA é passar as informações gênicas provinda do DNA para o ribossomo, onde as proteínas são sintetizadas. Sua ocorrência é de 1 a 5% do total de RNAs na célula [14].

2) RNA Ribossômico: É a estrutura principal dentro dos ribossomos. Representa em torno de 75% do total de RNAs das células formando fitas duplas por pareamento internamente [14].

3) RNA Transportador: É responsável por levar os resíduos de aminoácidos para os ribossomos para a sintetização das proteínas. Representa de 10 a 15% da quantidade de RNAs nas células [14].

E. Genes

Segundo Noble, existem várias controvérsias em relação ao o que é o gene e seu significado. Todavia para a biologia molecular, após a descoberta que o DNA codifica as proteínas, a definição mudou para intervalos no DNA que possuam começo e fim identificáveis (Figura 4) [15].

F. Aminoácidos

Proteínas são polímeros de aminoácidos. Esses aminoácidos são unidos um a ao outro por um tipo específico de ligação covalente. As proteínas podem ser degradadas em seus próprios aminoácidos de algumas formas. Naturalmente os primeiros estudos de proteínas se iniciaram por seus aminoácidos, visto que dentro das proteínas são comumente encontrados 20 tipos de aminoácidos, sendo que grande parte deles tem o seu nome baseado ou no organismo onde foi descoberto, ou por uma característica sua. Eles também geralmente são representados por uma sigla com 3 letras ou por uma letra. [11].

G. Códons

Como citado anteriormente o DNA possui duas fitas enroladas, com quatro pares de bases possíveis, ligados com o par na fita oposta. Também é dito que o DNA codifica as proteínas, essas que são processadas a partir tradução das bases do DNA pelo RNA transportador. Todavia comumente observa-se 20 proteínas dentro de um organismo celular. Contudo a forma como as proteínas são definidas é diferente, pois elas são definidas por três pares de base, podendo gerar até 64 combinações diferentes (4 x 4 x 4 = 64), muito além do número necessário. Essa trinca de bases é chamada de códon [13]. É possível ver o mapeamento de códons para aminoácidos na Tabela I.

H. ORF

Como nos genes também existe mais de uma definição para as fases de leitura aberta, ou também chamadas de Open Read Frame (ORF). Segundo Sieber existem três definições para uma ORF como mostra a Figura 5. A primeira delas diz que uma ORF é uma sequência de tamanho divisível por três, que inicia com um códon de início tradutor (ATG) e que termina com um códon de parada. A segunda definição é semelhante a primeira, a diferença está em que nessa definição tanto o início quanto o fim da sequência são dados por códons de parada. Por fim, a terceira definição fala que uma ORF é uma sequência delimitada por um sítio de splice aceitador e um doador. Dessa forma, refere-se a um éxon interno eucariótico potencialmente traduzido. Os éxons terminais 5' e 3' de um

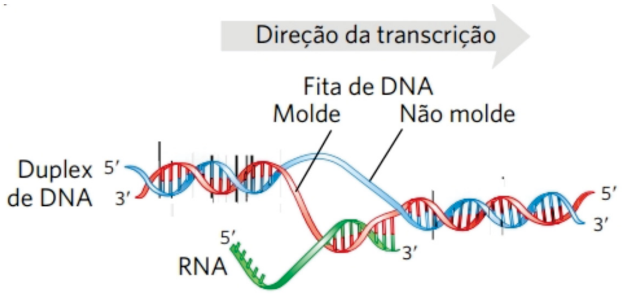


Fig. 3. Modelo demonstrando como um RNA é gerado a partir do DNA [14]

.

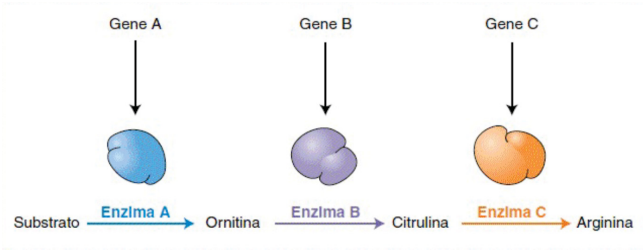


Fig. 4. Modelo gene-uma enzima. Esse modelo prega que os genes codificam enzimas, estas que desempenham diversos papéis bioquímicos no organismo do ser vivo. Foi proposto por Tatum e Beadle após estudarem a síntese da arginina no fungo do pão Neurospora crassa. [13]

TABELA I
MAPEAMENTO DE CÓDONS

Aminoácido	Códon(s)
Ile	ATT ATC ATA
Leu	TTA, TTG, CTT, CTC, CTA, CTG
Val	GTT, GTC, GTA, GTG
Phe	TTT, TTC
Met	ATG
Cys	TGT, TGC
Ala	GCT, GCC, GCA, GCG
Gly	GGT, GGC, GGA, GGG
Pro	CCT, CCC, CCA, CCG
Thr	ACT, ACC, ACA, ACG
Ser	TCT, TCC, TCA, TCG, AGT, AGC
Tyr	TAT, TAC
Trp	TGG
Gln	CAA, CAG
Asn	AAT, AAC
His	CAT, CAC
Glu	GAA, GAG
Asp	GAT, GAC
Lys	AAA, AAG
Arg	CGT, CGC, CGA, CGG, AGA, AGG
Parada	TAG, TAA, TGA

Tabela de mapeamento de códons para proteínas [16]

gene putativo são determinados ao fim do processo de predição do gene e não são considerados para a detecção de ORF real. [17] Para esse artigo é usada a segunda definição para trabalhar com os dados.

I. Regiões Codificadoras

Um dos passos importantes na análise das informações do genoma é decifrar o potencial completo de codificação ou a região codificadora da proteína (CDS) de cada gene. A CDS é uma sequência de nucleotídeos que correspondem com uma sequência de aminoácidos em uma proteína. Um CDS típico começa com as bases ATG e termina com um códon de parada. Ela também pode ser um subconjunto de um fase de leitura aberta. Em organismos eucariotos, predição de regiões codificadoras na sequencia do genoma é difícil, visto que há uma baixa porcentagem do genoma dedicado as CDS e também há inúmeras interrupções das regiões do CDS por introns. Não é possível, no momento, prever a partir da sequência do genoma a distribuição correta das regiões codificadoras que aparecem nas proteínas expressas por genoma [18].

J. Anotação Genômica

O processo de anotação genômica se dá pela identificação de padrões e de genes no DNA obtido através do sequenciamento. Existem diversos algoritmos que buscam fases de leitura aberta, representando que este intervalo pode ser uma região codificadora [19].

As anotações genômicas costumam ser separadas entre dois tipos manual ou automática. A anotação manual acontece quando o pesquisador cria e identifica manualmente as anotações. Já a automática é feita por programas que geram as geram automaticamente ou são baseadas a partir de dados de bases públicas. Além disso, as anotações também dependem da experiência dos pesquisadores em observar os resultados e

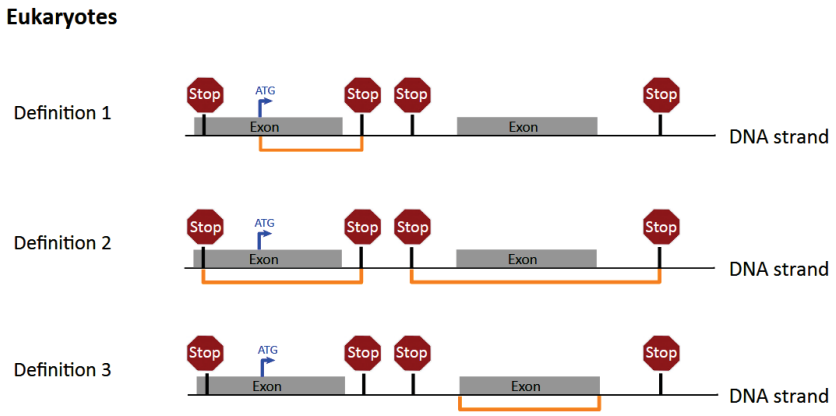


Fig. 5. Diferenciação das três definições de fase aberta de leitura em um organismo eucarioto [17].

interpretá-los. Após as descobertas os pesquisadores submetem os resultados de suas pesquisas bem como a anotação feita para para bases de dados públicas, tornando o acesso ao conhecimento mais acessível para outros pesquisadores usarem [20].

Apesar das anotações estarem melhorando constantemente, ainda ocorrem vários erros de anotações nos dados depositados em bancos de dados públicos. Entretanto, anotar uma montagem de genoma eucarioto ainda é desafiador, visto que um genoma eucarioto apresenta um nível de complexidade elevado, quando comparado aos procariotos. [21]

K. Etapas da anotação genômica

Segundo Yandell, a anotação pode ser divida em duas etapas. Uma é denominada anotação estrutural e é responsável por definir os genes, regiões repetitivas, diferentes tipos de RNA, bem como a composição de sequência e as regiões de início e fim, sendo esses exemplos considerados características genômicas. Já a outra etapa chama-se anotação funcional e tem como objetivo compreender e atribuir função a um gene ou elemento do genoma [22].

Já Beloze separa em três etapas. Na primeira etapa é utilizado softwares de anotação e comparação de bases públicas para destacar possíveis anotações. Após a utilização da anotação automática os especialista precisam analisar as anotações geradas para confirmar e encontrar os melhores resultados. Por fim é feita uma descrição pelo anotador sobre as regiões encontradas (Figura 6). Tanto a segunda quanto a terceira etapa são englobadas pela anotação manual [20].

L. Erros na anotação

Como citado anteriormente nas bases públicas podem conter erros de anotação, sendo que esses podem ser de diversos tipos. Alguns erros que podem ocorrer são: genes incompletos, duplicados, divididos, dentre outros. Ademais a para que a anotação funcional esteja correta a estrutural é de suma importância. No geral os erros da parte funcional são na maioria causados por falhas na estrutura [21].

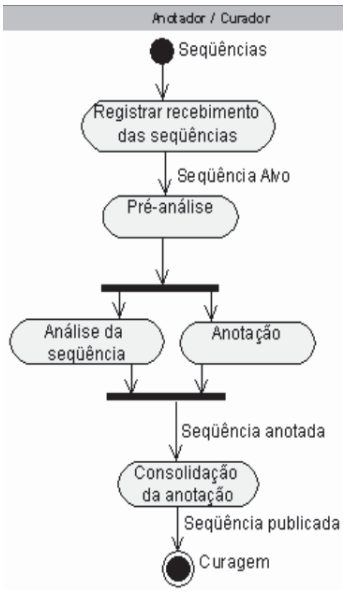


Fig. 6. Processo do estudo das sequencias genômicas [17]

M. Descoberta de Conhecimento em Bases de Dados

Atualmente muito se houve falar em mineração de dados em grandes empresas. Entretanto a parte de mineração de dados é apenas uma das etapas de um processo maior chamado de Knowledge Discovery in Databases (KDD). De forma concisa o termo KDD é pode ser referido como ”o processo geral de descoberta de conhecimento útil a partir de dados”. Para que este processo aconteça exitem algumas etapas propostas a se seguir. Para o dado ser transformado em conhecimento deve-se passar primeiro pela seleção, pré-processamento, transformação, mineração de dados e por fim a interpretação (Figura 7) [23].

N. Inteligência Artificial

A inteligência artificial (IA) usa computadores para simular comportamentos humanos inteligentes e treina-os para apren-

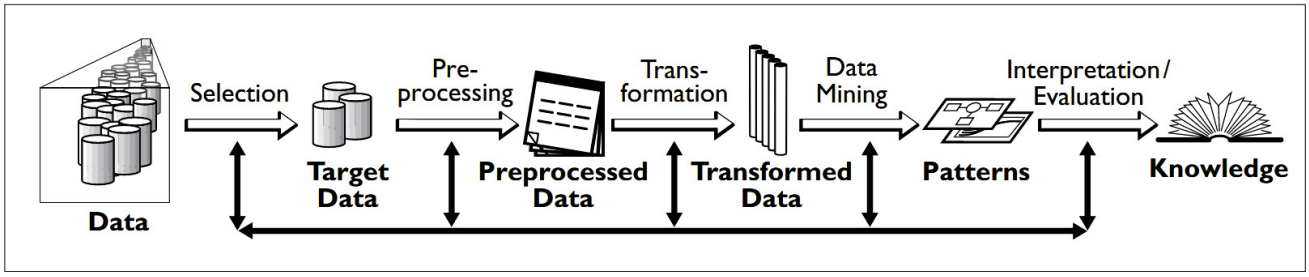


Fig. 7. Processo de KDD demonstrando dados que são selecionados, então pré-processados, para serem transformados, para que possam ser obtidos padrões, sendo que a interpretação desses padrões resultad em conhecimento [23].

der comportamentos humanos, como aprendizado, julgamento e tomada de decisão. Essa área de estudo toma o conhecimento como o objeto, adquire novos conhecimentos, analisa e estuda a expressão métodos desses conhecimentos, e emprega essas abordagens para alcançar o efeito de simular atividades intelectuais humanas [24].

Nos últimos cinquenta anos, o tema da inteligência artificial recebeu uma atenção renovada no meio acadêmico. O Projeto de Pesquisa Dartmouth definiu a IA como o problema de fazer uma máquina se comportar de maneiras que humanos diriam ser inteligentes. Portanto, a IA tem pode ser entendida como a capacidade de um sistema de agir de forma inteligente e em regiões cada vez mais amplas, interpretando corretamente dados externos e usando esses ensinamentos para atingir objetivos e atividades específicas por um configuração flexível [25].

Já nas décadas de 1980 e 1990 iniciou-se um aumento no interesse pela inteligência artificial. Técnicas de inteligência artificial, como sistemas especialistas fuzzy, redes bayesianas, redes neurais artificiais e sistemas inteligentes híbridos, foram usados em diferentes ambientes clínicos em assistência médica. Em 2016, a maior fatia dos investimentos, em comparação com outros setores, dessa área de pesquisa foram em aplicações de saúde [26].

Todavia, não é apenas no meio acadêmico que se interessa pela IA. As organizações que a implementam esperam que os aplicativos de IA obtenham ganhos em termos de valor de negócios, como aumento de receita, redução de custos e melhoria da eficiência dos negócios. Um estudo recente feito pelo MIT Sloan Management Review descobriu que mais de 80% das organizações veem a IA como uma oportunidade estratégica, e quase 85% veem a IA como uma forma de alcançar vantagem competitiva. Na busca por vantagem competitiva, muitas organizações estão investindo em tecnologias de IA. No entanto, apesar do crescente interesse pela tecnologia, muitas empresas lutam para obter valor da IA [27].

Mesmo já tendo passado anos de estudo da IA e com uma abundância de artigos em revistas de tecnologia e de outros temas cobrindo os tópicos de aprendizado de máquina (machine learning ou ML), aprendizado profundo (deep learning ou DP) e IA, ainda há confusão em torno da diferença desses tópicos. Os termos são altamente associados, mas não

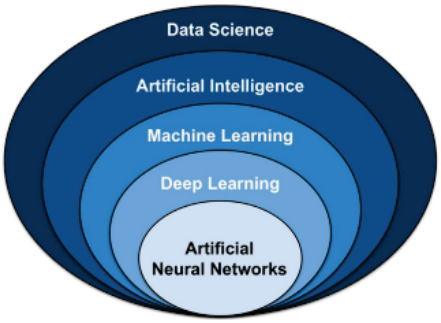


Fig. 8. Agrupamento de técnicas selecionadas de ciência de dados. Inteligência artificial (IA) se enquadra no domínio da ciência de dados e inclui programação clássica e aprendizado de máquina (ML). ML contém muitos modelos e métodos, incluindo aprendizado profundo (DL) e redes neurais artificiais (ANN) [28]

são intercambiáveis. Em 1956, um grupo de cientistas da computação propôs que os computadores podem ser programados para pensar, "que todo aspecto da aprendizagem ou qualquer outro característica da inteligência poderia, em princípio, ser tão precisamente descrito, que uma máquina poderia ser feita para simulá-la. Eles descreveram esse princípio como inteligência artificial. Simplificando, a IA é um campo focado em automatizar tarefas intelectuais normalmente executadas por humanos, e ML e DL são métodos específicos de alcançar este objetivo. Ou seja, os dois estão dentro do área da IA (Figura 8) [28].

O. Reconhecimento de padrões

Tanto as redes neurais e o reconhecimento de padrões muitas vezes são complementares, visto que muitas das técnicas de redes neurais derivam do que é o reconhecimento de padrões. Exemplificando, o reconhecimento de padrões envolve principalmente a classificação de estímulos em categorias mutuamente exclusivas (Figura 9 [29]. A área de reconhecimento de padrões esta preocupado com a descoberta automática de dados semelhantes por meio do uso de algoritmos em computadores, para então utilizar essas semelhanças para categorizar os dados (Figura 10) [30]. Existem muitas boas técnicas de classificação na literatura incluindo classificador

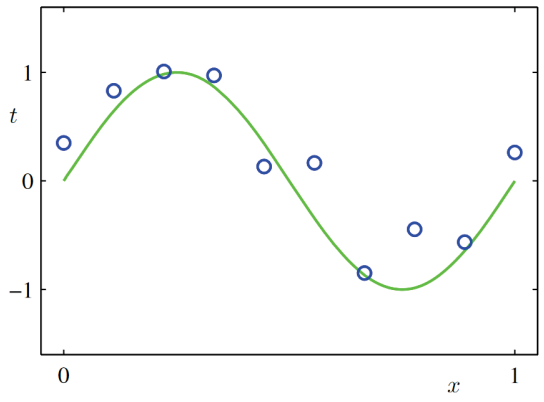


Fig. 9. Gráfico de um conjunto de dados de treinamento de N = 10 pontos, representados por círculos azuis, cada um compreendendo uma observação da variável de entrada x junto com a variável de almejada t. A curva verde mostra a função $sin(2\Pi x)$ usada para gerar os dados. O objetivo do reconhecimento de padrões é predizer o valor de t para algum novo valor de x, sem conhecimento de a curva verde. [30]

k-nearest-neighbor, redes Bayesianas, redes neurais artificiais, árvores de decisão e máquinas de vetores de suporte [31].

P. Aprendizado de Máquina

O aprendizado de máquina é um campo que se concentra no aspecto de aprendizagem de inteligência artificial, desenvolvendo algoritmos que melhor representam um conjunto de dados. Ao contrário da programação clássica, em que um algoritmo pode ser explicitamente codificado usando recursos conhecidos, o aprendizado de máquina usa subconjuntos de dados para gerar um modelo que pode usar combinações novas ou diferentes de características e pesos que podem ser derivados do primeiro princípios (Figura 12) [28].

O aprendizado de máquina pode ser dividido de acordo com a natureza da rotulagem de dados, podendo ser supervisionado, não supervisionado e semi-supervisionado como é apresentado na Figura 11. No Supervisionado o aprendizado é usado para estimar um mapeamento desconhecido (entrada, saída) para um mapeamento conhecido de amostras, onde a saída é rotulada (por exemplo, classificação e regressão). No aprendizado não supervisionado, apenas amostras de entrada são fornecidas ao sistema de aprendizado (por exemplo, agrupamento e estimativa da função de densidade de probabilidade). No Semi-supervisionado a aprendizagem é uma combinação de ambos, supervisionado e não supervisionado, onde parte do os dados são parcialmente rotulados e a parte rotulada é usada para inferir a parte não rotulada (por exemplo, sistemas de recuperação de texto/imagem) [32].

De uma perspectiva probabilística, os algoritmos de aprendizado de máquina podem ser divididos em modelos discriminantes ou gerais. Um modelo discriminante mede a probabilidade condicional de uma saída dada entradas tipicamente determinísticas, como redes neurais ou uma máquina de vetores de suporte. Um modelo geral é totalmente probabilístico quando está usando uma técnica de modelagem de grafos, como redes Bayesianas [32].

Q. Máquinas de Vetores de Suporte

As máquinas de vetores de suporte (Support Vector Machines ou SVMs) possui vários exemplos de sua eficácia em diversas áreas, como na categorização de textos, processamento de imagens e em na bioinformática [33].

Uma SVM é um algoritmo de computador que aprende a atribuir rótulos a objetos. Por exemplo, uma SVM pode aprender a reconhecer a atividade de um possível cartão de crédito fraudulento examinando relatórios de centenas ou milhares de cartões de crédito fraudulentos e não fraudulentos. Outra aplicação é poder aprender a reconhecer dígitos manuscritos examinando uma grande coleção de imagens digitalizadas de algarismos escritos. SVMs têm também já foram implementadas com sucesso a um variedade de aplicações biológicas. Teoricamente, uma SVM pode examinar o gene perfil de expressão gênica de uma amostra de tumor ou de fluido periférico e chegar a um diagnóstico ou prognóstico. Outras aplicações de SVMs na biologia envolvem a classificação de objetos tão diversas quanto sequências de proteínas e DNA, micro- perfis de expressão de matriz e espectros de massa [34].

O principal objetivo da SVM é separar várias classes no conjunto de treinamento com uma superfície que maximiza a margem entre eles (Figura 13). Em outras palavras, o SVM permite maximizar a capacidade de generalização de um modelo. Este é o objetivo do princípio da minimização de risco estrutural (Structural Risk Minimization ou SRM) que permite a minimização de um grupo no erro de generalização de um modelo, em vez de minimizar o erro quadrático médio no conjunto de dados de treinamento, sendo uma das técnicas usadas pelos métodos de minimização de risco empírico [31].

R. K Vizinhos Mais Próximos

O algoritmo de k vizinhos mais próximos (K-Nearest Neighbors ou KNN) tradicional é um dos métodos mais antigos e mais simples para reconhecimento de padrões. Mesmo assim, muitas vezes produz resultados competitivos e, em certos domínios, quando habilmente combinado com o conhecimento prévio, tem significativamente avançado o estado da arte da IA. A regra implementada pelo KNN classifica cada exemplo não rotulado pelo rótulo majoritário entre seus vizinhos mais próximos (K) no conjunto de treinamento. Seu desempenho depende crucialmente da métrica de distância usada para identificar vizinhos mais próximos. Na ausência de conhecimento prévio, a maioria dos classificadores KNN usam métrica euclidiana simples para medir a diferenças entre exemplos representados como entradas vetoriais [35].

A escolha de K (número de vizinhos) define a localidade do KNN. Para K = 1, pequenas vizinhanças surgem em regiões onde padrões de diferentes classes estão dispersos. Para vizinhança maiores, por exemplo K = 20, padrões com rótulos em minoria são ignorados. A Figura 14 ilustra a diferença na classificação entre KNN com K = 1 e K = 20 em um conjunto de dados bidimensional simples que consiste de duas nuvens de dados sobrepostas amostras gaussianas como 50 amostras cada, exibidas em pontos vermelhos e azuis. As localizações do espaço de dados que seriam classificadas

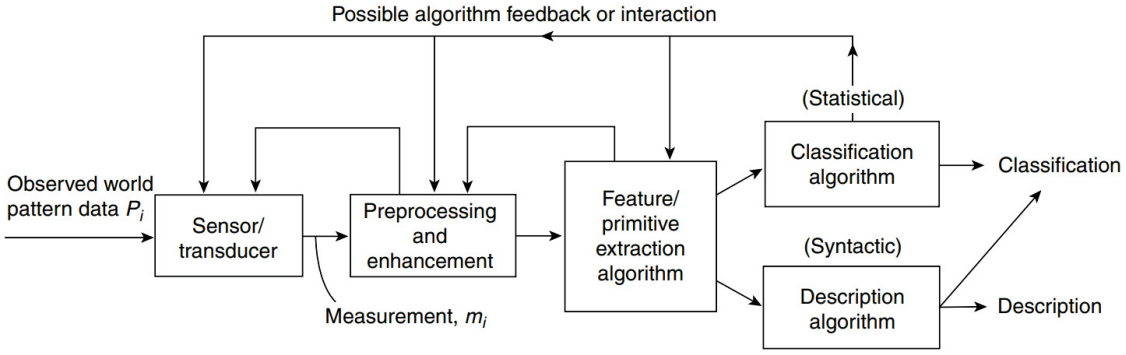


Fig. 10. Sistema genérico de reconhecimento de padrões. Um padrão observado no mundo entra por um sensor, para então ser pré-processado e melhorado, dirigindo-se para a extração de atributos, que são enviados ou para uma algoritmo de classificação ou para um de descrição. Todas as etapas podem possuir retroalimentação e/ou interações [6]

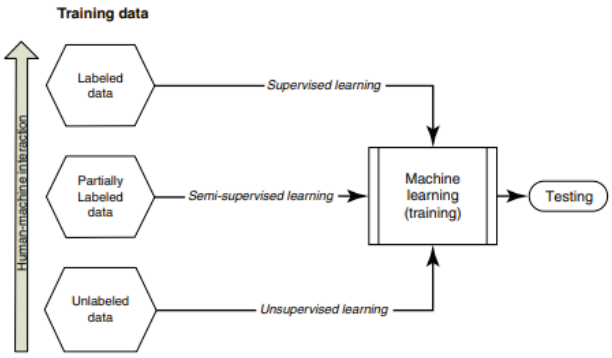


Fig. 11. Categorização dos tipos de aprendizado de máquina [28]

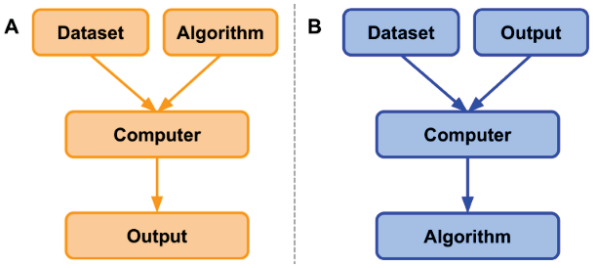


Fig. 12. Diferença entre a programação clássica e o paradigma de aprendizado de máquina. (A) Na programação clássica, um computador é fornecido com um conjunto de dados e um algoritmo. O algoritmo informa ao computador como operar no conjunto de dados para criar saídas. (B) No aprendizado de máquina, um computador é fornecido com um conjunto de dados e insumos associados. O computador aprende e gera um algoritmo que descreve a relação entre os dois. Esse algoritmo pode ser usado para inferência em conjuntos de dados futuros [28].

como azuis são mostradas em azul brilhante, enquanto as áreas classificadas como vermelhas são mostradas em branco. Para K = 1, a previsão é local. Por exemplo, um ponto azul que é um outlier (ponto destoante dos demais) da classe azul está localizada no centro da nuvem vermelha. Para um K maior, o classificador generaliza ignorando pequenas aglomerações

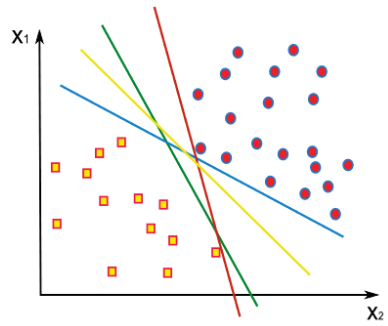


Fig. 13. Separação de duas classes dentro de um conjunto de dados utilizando vários hiperplanos usando SVM [31]

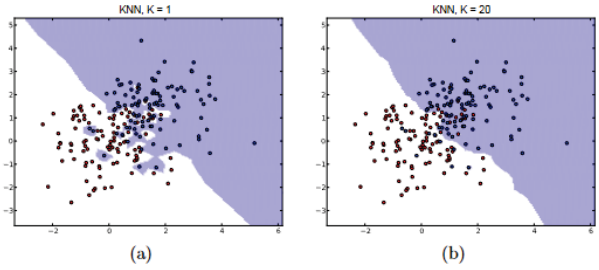


Fig. 14. Uma comparação da classificação KNN em duas nuvens de dados baseadas em Gauss para dois tamanhos de vizinhança ((a) K = 1 e (b) K = 20). Para vizinhanças pequenos, KNN tende a overfit (decorar a base de dados) tornando-se local, enquanto KNN generaliza o resultado para um K maior ignorando pequenas aglomerações de padrões [36]

de padrões. No caso de grandes conjuntos de dados, o KNN deve procurar os padrões K-mais próximos em todo o espaço, mas já pode produzir um boa aproximação com base nos K vizinhos mais próximos em um subconjunto escaneado [36].

S. Árvores de Decisão

As árvores de decisão (Decision Trees do inglês, ou DT) são um dos métodos mais poderosos comumente usado em vários campos, como aprendizado de máquina, imagem processa-

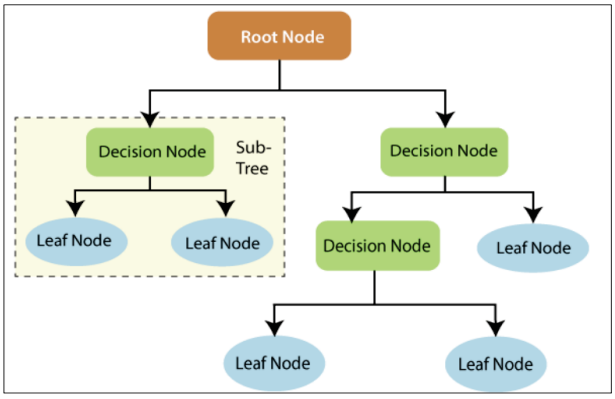


Fig. 15. Exemplo de funcionamento de uma árvore de decisão [38].

mento e reconhecimento de padrões. A DT é um modelo sucessivo que une uma série de testes básicos de forma eficiente e de forma coesa onde um recurso numérico é comparado a um valor limite em cada teste. As regras conceituais são muito mais fáceis de construir do que os pesos numéricos na rede neural na conexões entre nós. A DT é geralmente um modelo de classificação utilizado em mineração de dados. Por conta de sua análise simples e de sua precisão em vários conjuntos de dados, árvores de decisão encontraram muitos campos de implementação [37].

Uma árvore normal inclui raiz, galhos e folhas. A mesma estrutura é seguida na árvore de decisão. Ela contém o nó raiz, ramos e nós folha. O ato de testar um atributo está em cada nó interno. O resultado do teste está no ramo e no rótulo da classe que está no nó folha. Um nó raiz é pai de todos os nós e como o nome sugere é o nó mais alto dentro da árvore. Uma árvore de decisão é uma árvore onde cada nó mostra atributo, cada ligação mostra uma decisão, e cada folha mostra um resultado (categórico ou segue com o valor) como mostra a Figura 15. O fato das árvores de decisão imitam o pensamento do nível humano torna tão simples pegar os dados e fazer algumas interpretações [38].

T. Redes Neurais Artificiais

O estudo das redes neurais artificiais (Artificial Neural Networks ou ANNs) é motivado fortemente pela forma de como o cérebro humano processa as informações ao seu redor, sendo muito diferente da forma como um computador processa os dados. O cérebro humano é como um computador complexo não linear e paralelo, tendo a capacidade de organizar seus neurônios para realizar alguns processamentos mais rápido que um computador comum, como por exemplo, o reconhecimento de padrões [39].

As redes neurais podem ser definidas como modelos computacionais com propriedades particulares como a habilidade de se adaptar ou aprender, de generalizar, ou de agrupar ou organizar dados, onde sua operação é baseada em processamento paralelo (Figura 16). Contudo muitas dessas habilidades estão presentes em outras técnicas, porém as redes neurais provam ser mais adequadas para algumas tarefas [40].

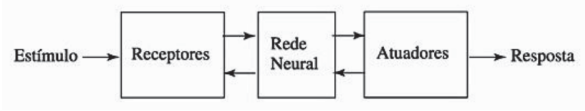


Fig. 16. Representação básica em diagrama de um sistema nervoso [39].

$$y = \Theta\left(\sum_{j=1}^{D} w_j x_j - \mu\right)$$

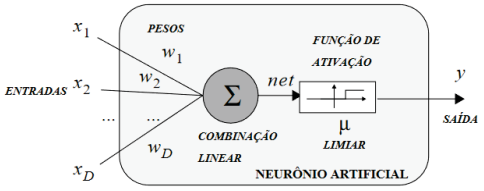


Fig. 17. Neurônio de McCulloch e Pitts [42]

O poder da rede neural provém de dois fatores. O primeiro é sua estrutura maciçamente e paralelamente distribuída. O segundo fator se dá pela sua capacidade de aprender ao longo do treinamento obtendo uma generalização. Esta generalização diz respeito a fato da rede conseguir expandir o número de saídas em relação ao início do treinamento. Com essas duas características as redes neurais conseguem resolver problemas de alta escala antes não tratáveis [39].

U. Computação Neural

A computação neural surgiu como uma tecnologia prática, com aplicações bem-sucedidas em várias áreas. As aplicações que usam dessa técnica podem fazer uso de arquiteturas de rede feed-forward, como o multi-layer perceptron e a rede de função de base radial. Porém para o reconhecimento de padrões, as redes neurais podem ser consideradas uma extensão das muitas técnicas convencionais que foram desenvolvidas ao longo de várias décadas [41].

V. Neurônio Artificial

Dentro das redes neurais artificias existem os neurônios artificiais. Os neurônios são responsáveis por processar um dado e retornar uma saída. Um exemplo de neurônio foi proposto por McCulloch e Pitts em 1943, este que a partir das entradas irá aplicar uma combinação linear em cima de pesos aplicados para cada entrada, para então passar esse valor para uma função de ativação que por fim resultará na saída do neurônio. Os pesos refletem na importância da entrada, esta que precisa passar por um limiar para ser direcionada a saída (Figura 17) [42].

W. Topologia de Redes de Neurônios Artificiais

Tendo definido um neurônio artificial é possível criar uma rede com vários. É a partir disso que vem o potencial a flexibilidade do cálculo em uma ANN, onde um neurônio processa uma entrada e passa para um ou mais neurônios que processaram a saída do anterior e assim por diante. Esse

paralelismo cria a ”inteligência” global da rede como apresenta a Figura 18 [42].

X. Propriedades e Capacidades das Redes Neurais

1) Não Linearidade: Um neurônio artificial pode ser linear ou não linear, sendo que uma rede neural composta por neurônios não lineares é uma rede não linear por si só. Esta é um propriedade muito importante principalmente se o dado de entrada também for não linear, como a voz por exemplo [39].

2) Mapeamento de Entrada-Saída: Um dos paradigmas relacionados a redes neurais é o aprendizado supervisionado. Esse aprendizado se dá por fornecer a rede exemplos pré-definidos a fim de que a rede possa usá-los e ajustar seus pesos sinápticos para melhor se adaptar as entradas. Dessa forma a rede consegue gerar um mapeamento entrada-saida a partir do treino proposto [39].

3) Adaptabilidade: Uma rede neural tem a capacidade de adaptar seus seus pesos sinápticos para cada situação, inclusive podendo ser configurada para mudar os pesos em tempo real caso necessário, fazendo dela uma ferramenta muito útil para tarefas adaptativas [39].

4) Resposta e Evidência: A rede ao trabalhar com a busca por padrões pode além de escolher o a categoria que o dado mais se encaixa dado, também pode se basear em graus de confiança para escolher essa categoria, podendo rejeitar padrões ambíguos e melhorar o desempenho da rede [39].

5) Informação Contextual: Como a rede está inteiramente ligada, cada mudança em um neurônio artificial afetará os outros de alguma forma, fazendo com que a rede consiga perceber informações dentro de um contexto [39].

6) Tolerância a Falhas: Pela informação da rede neural ser espalhada, uma falha que o ocorra em um neurônio irá afetar suavemente a rede, invés de provocar um falha fatal a todo o sistema, sendo que para danificar a rede seriamente é necessário uma falha muito extensa [39].

7) Implementação em VSLI: Por ser maciçamente paralela uma ANN tem vantagem para trabalhar com VSLI (very large scale integration), pois a rede pode se adaptar a uma escala de integração muito grande [39].

8) Uniformidade de Análise e de Projeto: Uma rede neural artificial possui a característica da universalidade da como processadores de informação. Isso significa que um mesmo padrão pode ser aplicado em diferentes redes. Por exemplo, todas as redes possuem neurônios, sendo possível compartilhar teorias e algoritmos em diferentes redes e torna possível construir redes modulares através de uma integração homogênea de módulos (Figura 19) [39].

9) Analogia Neurobiológica: Ao ponto que as ANNs se baseiam na neurobiologia para definir a forma como os dados serão processados visando replicar como o cérebro humano funciona, os neurobiólogos se espelham nas ANNs para estudar problemas mais complexos [39].

Y. Python

Python é uma linguagem de programação poderosa e fácil de aprender. Possui estruturas de dados de alto nível eficientes e uma abordagem simples, mas eficaz para programação orientada a objetos. A sintaxe elegante e a tipagem dinâmica do Python, juntamente com sua natureza interpretada, o tornam uma linguagem ideal para scripts e desenvolvimento rápido de aplicativos em muitas áreas na maioria das plataformas. O interpretador Python e a extensa biblioteca padrão estão disponíveis gratuitamente em formato fonte ou binário para todas as principais plataformas e podem ser distribuídos gratuitamente. Dentro das ferramentas que a linguagem oferece, também há distribuições e ponteiros para muitos módulos, programas e ferramentas de terceiros gratuitas do Python e documentação adicional. O interpretador Python é facilmente estendido com novas funções e tipos de dados implementados em C ou C++ (ou outras linguagens que podem ser integradas com C). O Python também pode ser usado como linguagem de extensão para aplicativos personalizáveis [43].

Existem muitas outras linguagens de alto nível poderosas e expressivas, incluindo Perl, Lisp, Scheme, Ruby, CAML e Haskell. De todos esses, o Python é um dos poucos que se baseia em pesquisas sobre usabilidade e os fatores que tornam uma linguagem de programação fácil de aprender e usar. No entanto, o Python também foi projetado para resolver problemas do mundo real enfrentados por um programador especialista. O resultado é uma linguagem que se adapta bem desde pequenos scripts escritos por um químico até grandes pacotes escritos por um desenvolvedor de software [43].

Z. Scikit-learn

Dentro das ferramentas para se trabalhar com inteligência artificial, uma delas é a scikit-learn. O Scikit-learn é uma biblioteca implementada na linguagem de programação Python que fornece uma interface padrão para implementação algoritmos de aprendizado de máquina. Inclui outras funções auxiliares que são importantes para os processos de aprendizado de máquina, como etapas de pré-processamento de dados, reamostragem de dados técnicas, parâmetros de avaliação e interfaces de pesquisa para ajustar/otimizar um desempenho do algoritmo [44].

III. Materiais e Métodos

Neste capítulo será apresentado como o trabalho foi desenvolvido e pensado.

A. Origem dos Dados

Para padronizar os dados e ao mesmo tempo visar uma fonte confiável de sequências e anotações é escolhida base a NCBI Genomes, e os bancos de dados RefSeq. Dessa forma, inicialmente são utilizadas sequências de organismos já bem consolidadas para análise prévia. A base do NCBI de referência de sequência (RefSeq) é uma coleção não redundante de sequências de genomas, transcritos e proteínas. A base contém dados de 3774 organismo divididos entre procariotos, eucariotos e vírus tendo dados de 2879860 proteínas na versão 19 (Figura 20) [45]. Os organismos escolhidos foram em sua maioria organismos recém anotados no ano de 2022 na NCBI Genomes RefSeq, sendo os seguintes organismos:

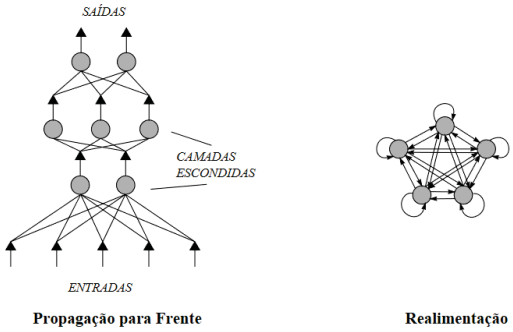


Fig. 18. Dois exemplos de topologias para redes neurais [42].

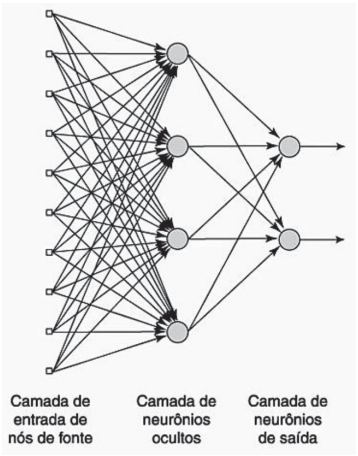


Fig. 19. Exemplo de rede acíclica totalmente conectada com uma com uma camada oculta e uma camada de saída [39].

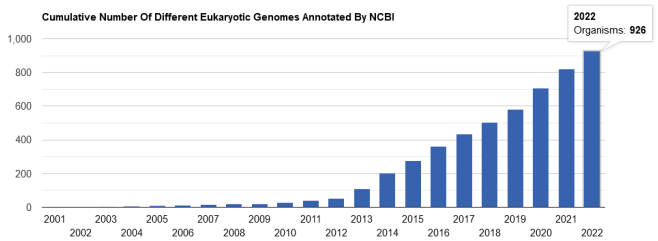


Fig. 20. Tabela no site do NCBI mostrando o número cumulativo de organismo eucariotos anotados até o ano de 2022 [16].

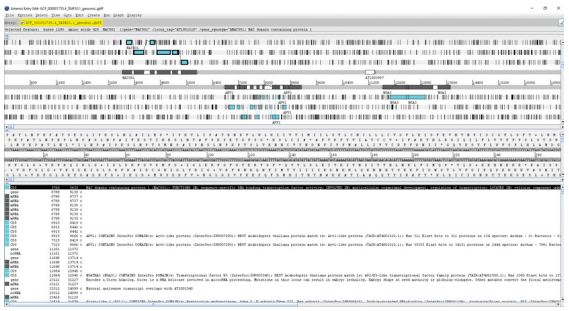


Fig. 21. Tela inicial do Artemis ao carregar o arquivo GBFF da Arabidopsis thaliana

Arabidopsis thaliana, (pequena planta com flor da família da mostarda e a primeira a ser completamente sequenciada), Drosophila biarmipes (mosca que se alimenta de frutas), Pisum sativum (espécie de ervilha), Vespula vulgaris (espécie da família das vespas), Serinus canaria (canário comum), Gopherus flavomarginatus (tartaruga gigante mexicana), Eriocheir sinensis (caranguejo-peludo-chinês), Gymnogyps californianus (condor-da-califórnia), Cygnus atratus (cisne-negro) e Cataglyphis hispanica (espécie de formiga) [16]. Os arquivos a serem baixados contendo o genoma dessas espécies são os arquivos do tipo GBBF (Genomic GenBank format). Esse é o formato de arquivo simples do GenBank para a(s) sequência(s) genômica(s) na montagem. Este arquivo inclui tanto a sequência genômica quanto a descrição dos contigs (pedaços do genoma) [16]. Em outras palavras esses arquivos possuem não só a sequência genômica anotada, mas também a descrição das funções de cada parte da sequência, como por exemplo, onde estão os genes na sequência.

B. Análise dos dados

Para análise dos dados é utilizado o software Artemis, a fim de analisar a quantidade de genes encontrados para a decisão de como separar o processamento dos arquivos. Dentro do software é possível encontrar uma listagem de todos os genes, como exibe a Figura 22, podendo assim mensurar o tamanho da carga gerada por cada espécie. O Artemis permite que o usuário tenha uma visualização interativa entre genomas completos e as anotações genômicas associadas (Figura 21) [46].

C. Configuração dos dados

Dadas as dez espécies escolhidas para a extração dos dados, seus arquivos se encontram com as seguintes configurações: o tamanho em mega bytes provém do sistema operacional Windows 11, o número de mega bases e o número de genes provém do NCBI [16]. A Tabela II mostra em quantidades algumas características dos dados.

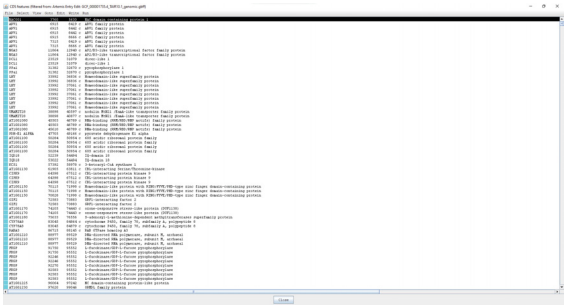


Fig. 22. Listagem dos genes no Artemis com arquivo GBFF da Arabidopsis thaliana

TABELA II
DADOS DOS ARQUIVOS DE CADA ESPÉCIE

Espécie	Mega Bases	Mega Bytes	Genes
Arabidopsis thaliana	120.087	374.764	38.309
Drosophila biarmipes	185.319	309.441	15.802
Pisum sativum	3796.07	4.943.051	61.482
Vespula vulgaris	185.444	307.974	11.750
Serinus canaria	1071.66	1.430.685	16.487
Gopherus flavomarginatus	2522.62	3.277.232	26.261
Eriocheir sinensis	1410.66	2.396.803	8.357
Gymnogyps californianus	1240.25	1.641.520	16.016
Cygnus atratus	10911.5	1.525.702	15.190
Cataglyphis hispanica	282.589	457.707	11.428

D. Separação dos dados

Tendo os dados escolhidos, a nível de código, os mesmos são separados em dois conjuntos a parte. O primeiro conjunto é o de treino, representado por 76,4% da população dos dados, e é usado para treinar o modelo de reconhecimento de padrões. O segundo conjunto representa 33,3% dos dados obtidos, é usado para validar o treino feito em cima do outro conjunto. Todavia para a validação final do modelo a proporção pode ser diferente. Tendo em vista que o organismo Pisum sativum é o maior em tamanho do genoma e número de genes, é decidido que ele será o organismo que testará o modelo final. Essa decisão visa possibilitar um treinamento com maior diversidade de organismos, visto que a espécie de ervilha escolhida ocupa uma grande porcentagem da base (38,55%) como mostra a Tabela III. Entretanto essa porcentagem ainda é próxima do valor estabelecido para testes.

E. Tecnologias

Com a base estabelecida, já é possível inciar a discussão das tecnologias empregadas. O ambiente de implementação escolhido é em cima da plataforma Python em sua verão 3.9,

TABELA III
DISPOSIÇÃO DOS DADOS PARA O MODELO FINAL

Tipo	Quantidade de Genes	Porcentagem do Total
Treino	159.500	61,45%
Teste	61.482	38,55%

TABELA IV
BIBLIOTECAS EXTERNAS PYTHON USADAS

Nome	Tipo	Versão
matplotlib	Externa	3.4.0
joblib	Externa	1.2.0
sci-kit-learn	Externa	1.1.2
pandas	Externa	1.5.0
re	Interna	-
json	Interna	-
sys	Interna	-
uuid	Interna	-
time	Interna	-
sys	Interna	-
random	Interna	-

Externas = terceiros. Internas = Python [43].

utilizando algumas bibliotecas (lista disponível na Tabela IV). Esta plataforma foi escolhida pois, com seu foco principal na legibilidade, o Python é amplamente reconhecido por ser fácil de aprender, porém consegue ser tão potente quanto outras linguagens do mesmo tipo. Além dos benefícios da própria linguagem, a comunidade em torno das ferramentas e bibliotecas disponíveis tornam o Python particularmente atraente para trabalhos nas áreas de ciência de dados, aprendizado de máquina e computação científica [47].

Como ambiente de desenvolvimento foi utilizado o software PyCharm 2022.2. Já como plataforma para todas as aplicações foi utilizado um Notebook Acer Nitro 5 ANS517-51, tendo um processador Intel(R) Core(TM) i5-9300H 2.40 GHz, uma placa de vídeo GTX 1650 com 4GB de VRAM, 32GB de memória RAM e 512GB de SSD NVME M2.

F. Extração dos Dados

Tendo os arquivos GBFF, é necessário lê-los para extrair apenas os genes e montar uma base com apenas os dados necessários. Esse tipo de arquivo possui formatações específicas para delimitar as diversas informações contidas nele. Por exemplo, o intervalo da sequência de DNA que contém um gene, é disposta da seguinte forma:

```
gene            3631..5899
```

Essa linha informa que um gene se encontra a partir da base nitrogenada na posição 3631, até a base na posição 5899 dentro da sequência de DNA. Baseado nesse padrão é que a implementação se inicia, guardando os intervalos de todos os índices dentro de um arquivo.

Feita a identificação das posições dos genes, o próximo passo é extrair a sequência que o representa de dentro do genoma. Para isso é usado outro padrão contido no arquivo GBFF. A partir de certos pontos começa a leitura de toda a sequência de DNA do arquivo. Esses pontos se chama ORIGIN, e partir deles começam as sequências de bases nitrogenadas, separadas de dez em dez, contendo 60 em cada linha. Além dessa marcação, a primeira informação de cada linha representa o posição que a primeira base se encontra, padronizada como no exemplo a seguir:

```
ORIGIN
```

```
     1 ccctaaaccc taaaccctaa accctaaacc tctgaatcct taatccctaa atccctaaat
    61 ctttaaatcc tacatccatg aatccctaaa tacctaattc cctaaacccg aaaccggttt
   121 ctctggttga aaatcattgt gtatataatg ataattttat cgtttttatg taattgctta
   181 ttgttgtgtg tagatttttt aaaaatatca tttgaggtca atacaaatcc tatttcttgt
```

Esse exemplo mostra que iniciou-se a transcrição do DNA com a marcação ORIGIN, e também exibe as 4 primeiras linhas do DNA no arquivo GBFF, onde o número no início da linha identifica qual a posição da primeira base da linha dentro da sequência completa. Com essa padronização, o algoritmo de extração olha sempre para a primeira informação da linha e compara com o primeiro índice dentro da lista de genes. Caso o início do gene esteja dentro da linha a ser lida, o algoritmo começa a guardar a sequência até que se chegue ao índice final do gene em questão. Após isso o gene é removido da lista de índices e o processo segue até que não haja mais nenhum gene. Essa abordagem se torna possível pois as informações genes são listados em ordem dentro do arquivo, do início da sequência, até o fim.

Por fim, a última etapa é o algoritmo saber o que representa o fim da sequência. No mesmo padrão da marcação de início, também existem uma que indica o fim, dada por duas barras como no trecho a seguir:

```
 30427441 tctacaaaca cagtcaatcc tgcagaaagg tacatccaga taatctcagt cgacgaccat
 30427501 gagttttggt tcatgtgttt cttaaactac tgtgtgatat aatattggtg gattcttgat
 30427561 tagaataaca ttattttggt tgcaaactaa attatttgat tcttattctt aattagttac
 30427621 catgtcttga ttagggttta gggtttaggg tttagggttt agggtttagg g
//
```

O exemplo apresenta as quarto últimas linhas da sequência, e com as duas barras sinalizando o fim, o algoritmo sabe que pode parar de guardar os genes e aguardar o próximo ORIGIN que apresentará outra sequência.

Entretanto esse algoritmo não extraia apenas genes, mas também extrai as sequências que não são genes para que mais tarde o modelo possa ter os dois exemplos. A extração dos trechos que não são genes ocorre juntamente com a extração dos genes, porém nos intervalos onde o que é gene não está sendo gravado, o código que lê não genes começa a atuar e a guardar as sequências. O maior diferencial é que, a parte que guarda o fator positivo começa a atuar ao encontrar o primeiro índice da lista da lista de genes, parando de atuar quando a lista de índices acaba, e a parte que armazena os fatores negativos começa atuar ao início da sequência e finaliza apenas quando encontra o fim da sequência. Apesar dos arquivos seguirem um padrão bem definido, foram encontrados genes que estavam na ordem de aparição dentro do genoma, dificultando a extração e comprometendo alguns dados da base gerada.

Visando facilitar o uso do algoritmo de extração também foram implementadas três funcionalidades no mesmo. A primeira é a possibilidade de poder passar quanto arquivos necessários para o algoritmo que o mesmo irá executar um a um, salvando sempre entre cada arquivo de genoma, sempre no mesmo arquivo final. Já a segunda serve para que não seja necessário o usuário ter que juntar manualmente mais de uma base gerada pela aplicação ao rodar várias vezes distintas. Um identificador único é gerado a cada execução e gravado no nome do arquivo da base gerada. Com esse identificador o usuário pode passá-lo para o algoritmo, para que então, os novos genomas a serem processados possam ser gravados dentro do mesmo arquivo sempre. Com os identificadores é possível também distinguir mais de um execução. Por fim a última funcionalidade é um mapeamento de todas as execuções. Esse mapeamento permite avisar o usuário caso o mesmo esteja tentando processar o mesmo arquivo mais de uma vez para a mesma execução. Apesar de existir o aviso, o próprio usuário que decide em processar ou não o arquivo possivelmente repetido. Todavia é possível desabilitar essa checagem e executar a ferramenta sempre no automático.

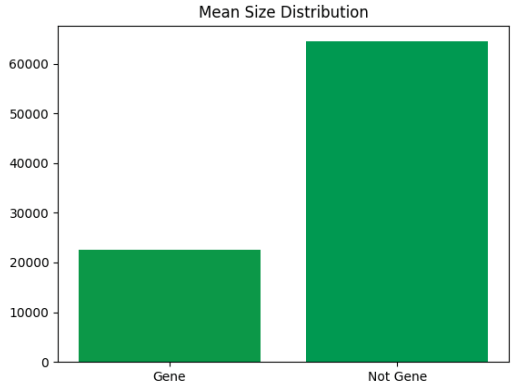


Fig. 23. Tamanho médio de sequência entre as classes.

G. Extração de Novas Informações

Com a base de genomas extraída, a primeira vista tudo o que há dentro de cada sequência é uma grande linha de texto com variações entre quatro letras apenas. Com apenas isso os algorítimos do scikit-learn não conseguem fazer sua classificação, dada documentação do mesmo [48]. O gráfico de diferença de tamanho médio das sequência mostra que a diferença é bem discrepante entre as classes, sendo que os não genes são cerca de seis vezes maiores como mostra a Figura 23. A partir desse atributo que começa a busca por outros que possam delimitar o que é e não é gene, e o tamanho consegue mostrar uma possível boa propriedade.

Partindo do mesmo princípio de contagem de elementos, outro atributo é obtido é a quantidade média de bases nitrogenadas nas duas classes. Se houvesse uma discrepância grande entre as ocorrências de bases, poderiam ser até 5 atributos fortes, todavia como o gráfico mostra uma curva ”U” semelhante na contagem média das bases, o que acaba por não diferir muito as classes. A única diferença apresentada está nas quantidades, entretanto essas informações já são explicadas pela diferença de tamanho de sequência de cada rótulo (Figura 24) .

Outro atributo extraído da coleção é a quantidade média de ORFs para cada tipo de item. Entretanto a extração dessa informação se provou cair no mesmo enviesamento da anterior, onde o dado extraído é também explicado pela diferença de tamanho dos contigs (Figura 25).

Por fim a última informação extraída em busca de novos atributos é a contagem média de aminoácidos das sequências.

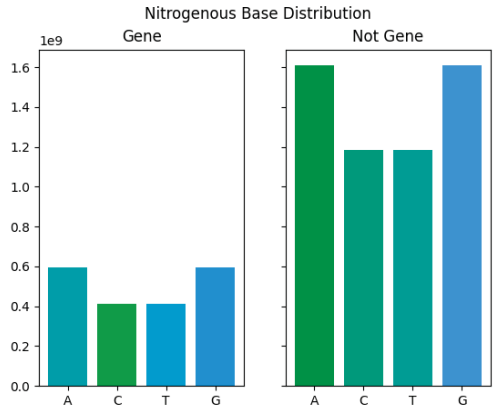


Fig. 24. Diferença de ocorrência média de bases entre as classes.

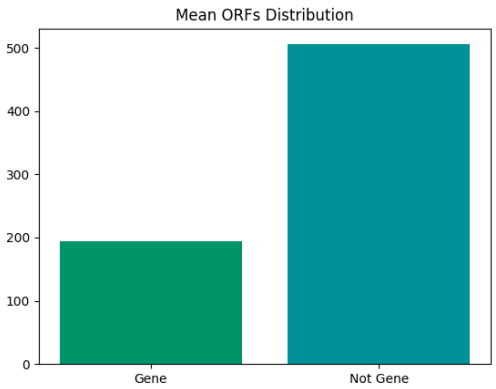


Fig. 25. Diferença de ocorrência média de ORFs entre as classes.

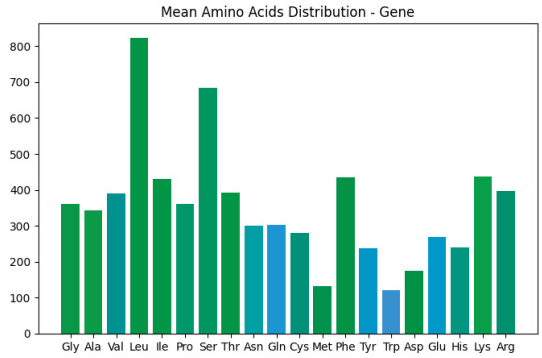


Fig. 26. Diferença de ocorrência média de aminoácidos para genes.

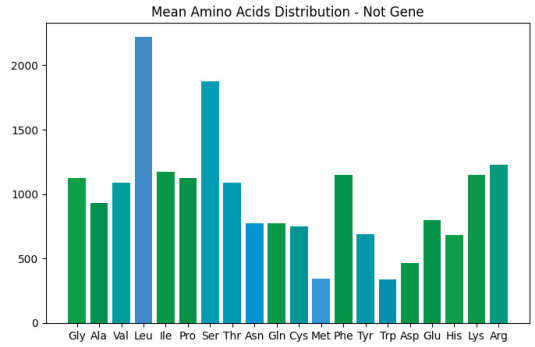


Fig. 27. Diferença de ocorrência média de aminoácidos para não genes.

Com essas ocorrências são gerados 20 atributos, um para cada aminoácido, aumentando as chances de achar uma propriedade descritiva, mas correndo o risco de cair no enviesamento por estar utilizando da contagem novamente. Todavia essa abordagem mostrou ter algumas informações que explicam um pouco mais cada classe, apesar de ainda nada chegar perto da diferença de tamanho. Com as informações do gráfico, por demonstrarem uma maior variação entre classes, demonstrando diferença na curva entre os gráficos, foi observado que a glicina, valina, prolina, glutamina, tirosina, arginina e o glutamato tem um maior potencial de auxiliar na explicação da diferença entre as informações genômicas como desenhado nas Figuras 26 e 27.

Os objetivos visados na procura por atributos explicatórios são fornecer para os algoritmos dados assertivos com o intuito de melhorar os resultados, e também para diminuir o custo computacional na execução das ferramentas. Reduzir o número de atributos para um problema de classificação é uma área muito avaliada. A abordagem da força bruta para encontrar a melhor combinação de atributos para classificação requer a tentativa de todas as combinações possíveis de todos os atributos disponíveis. Ou seja, considera-se um atributo de cada vez, então é investigada todas as combinações de dois atributos, três atributos, e assim por diante [49].

H. Métricas

Para a avaliação dos modelos de inteligência artificial que posteriormente gerados, foram escolhidas as seguintes métricas:

1) Acurácia e Precisão: A acurácia pode ser definida como grau de proximidade de uma estimativa com valor verdadeiro. Já a precisão representa o grau de consistência da grandeza medida com sua média (Figura 28). Ou seja, a acurácia determina o quão perto uma grandeza estatística está de seu valor real e a precisão está lida com a dispersão da distribuição das observações [50].

2) Coeficiente de Determinação: Dentro de um modelo linear simples, onde a variável preditora é x a variável de resposta é y, o coeficiente de determinação (R2) demonstra a porcentagem das variações de variáveis y serem explicadas pelas variações das variáveis x. Dessa forma, quando o valor obtido é próximo a 1, os pontos dentro o diagrama de dispersão estão próximos a reta da regressão. Já quando esse valor é próximo de 0, indica-se que a regressão não é linear [51].

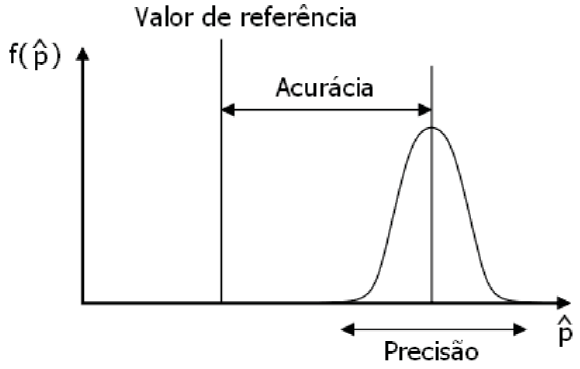


Fig. 28. Diferença entre acurácia e precisão [50].

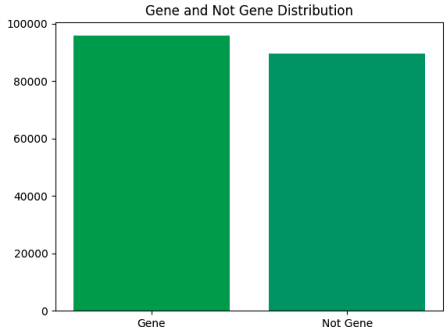


Fig. 29. Diferença da quantidade de sequências em cada classe.

3) Curva ROC: O ROC (Relative Operating Characteristic) é um método quantitativo para comparar uma variável booleana de referência contra um índice. A palavra índice é usada pois índice é geral e tem significado em termos de classificação. Um índice exibe valores mais altos para observações que são consideradas mais prováveis para a presença de um recurso booleano. A variável booleana de referência mostra presença em relação a ausência de um recurso, onde cada observação é geralmente codificada como 1 para presença e 0 para ausência. Uma curva ROC é obtida comparando a variável booleana de referência sucessivos limiares do índice. Dada essas informações, uma métrica bastante usada é a área sob a curva ROC, denominada AUC, que pode variar de 0 a 1, onde AUCs maiores indicam uma associação positiva mais forte [52].

4) Erro Quadrático Médio: Para avaliar o desempenho de um método de aprendizado estatístico em um determinado conjunto de dados, é necessário medir o quão bem suas previsões realmente correspondem aos dados observados. Ou seja, é preciso quantificar até que ponto o valor de resposta previsto para uma determinada observação está próximo do valor de resposta verdadeiro para essa observação. Na regressão, a medida mais comumente usada é o erro quadrático médio (MSE) dada pela seguinte fórmula: $\sum_{i=1}^{D}(x_i - y_i)^2$, onde o resultado será pequeno para casos onde as predições estão perto dos valores corretos ou será menor para caso estejam distantes [53].

5) Matriz de Confusão: Dentro das métricas são úteis gerar não apenas a acurácia média ou erro médio de tal de um modelo, mas também o que é chamado de matriz de confusão. Ela é uma matriz que mostra quais classes foram confundidas entre si durante o teste. Tal matriz fornece informações muito mais detalhes sobre os resultados do teste além da mera precisão ou erro. Exibe quais classes foram classificadas corretamente ou quase corretamente e quais foram classificadas incorretamente/confundidas com outras classes e em que grau. A análise de matrizes de confusão geralmente se mostra muito útil. Infelizmente, torna-se mais complicada a interpretação à medida que o tamanho da matriz cresce [54].

I. Construção do modelo

A construção do modelo se inicia como um experimento entre classificadores diferentes. Inicialmente são escolhidos um algoritmo de árvore de decisão, um de k vizinhos mais próximos, uma rede neural de multicamadas e um de máquinas de vetores de suporte. A ideia inicial é analisar como cada um se comporta, e conforme os testes forem ocorrendo, sair de um em um até sobrar apenas o que se sair melhor. Para os primeiros testes a utilização desses métodos será a padrão fornecida pela biblioteca ou próxima a ela, visto que por exemplo, a média de tamanho das sequências que não são genes está na casa dos 60.000 bases, ou seja, linha de texto com pelo menos 60.000 caracteres. No total sequências gênicas e não gênicas, a base para treinar e testar o modelo ficou com 179.126 sequências de DNA. A mudança de parametrização de quatro algoritmos para testes teria uma penalidade muito alta computacionalmente como os experimentos mostrarão mais a frente. A parametrização completa dos algoritmos se encontra na Tabela V

Com os dados os atributos extraídos anteriormente, já é possível treinar os modelos, entretanto foi decidido que seria feito balanceamento nos dados. O primeiro balanceamento feito é o de quantidade de classes da base total, já que como mostra a Figura 29, em quantidade de contigs, existem mais genes. O objetivo é tentar diminuir o escopo de execução dos modelos e também não entregar muitas informações desbalanceadas para evitar o enviesamento.

A estratégia usada para alinhar a quantidade de contigs é simples, a classe que possui o maior número deve ser reduzida para se equiparar a quantidade da classe que possui menos. Dessa forma a quantidade de genes foi diminuída para ficar igual a de não genes. É possível ver o resultado na Figura 30.

IV. Resultados

O próximo passo no balanceamento das classes é a normalização dos atributos extraídos. As quantidades entre cada atributo não são proporcionas, pois algumas informações estão na casa dos milhares enquanto outras informações estão na cada das dezenas. Para deixá-los uniformes foi aplicada uma normalização linear. O intuito dessa transformação

TABELA V
PARAMETRIZAÇÃO DOS MODELOS CLASSIFICADORES

Modelo	Parâmetros
Árvore de Decisão	`criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=1, max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0`
K Vizinhos Próximos	`n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None`
Rede Neural de Multicamadas	`hidden_layer_sizes=(100,), activation='relu', solver='adam', alpha=0.0001, batch_size='auto', learning_rate='constant', learning_rate_init=0.001, power_t=0.5, max_iter=1000, shuffle=True, random_state=1, tol=0.0001, verbose=False, warm_start=False, momentum=0.9, nesterovs_momentum=True, early_stopping=False, validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08, n_iter_no_change=10, max_fun=15000`
Máquinas de Vetores de Suporte	`nu=0.5, kernel='poly', degree=3, gamma='auto', coef0=0.0, shrinking=True, probability=True, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None`

também é evitar o enviesamento dos resultados e diminuir o escopo que modelos trabalharão. Com os dados prontos então os quatro modelos são executados, utilizando-se de 33% da base para teste sempre embaralhando a base.

O primeiro resultado é baseado na base utilizando todos os atributos extraídos (tamanho da sequência, contagem de bases nitrogenadas, contagem de aminoácidos e contagem de ORFs) e tem um resultado mediano em três dos modelos (Tabela VI). Como principal métrica fica estabelecida a acurácia, visando gerar modelos que tenham uma maior chande de fazer uma predição. Visto isso ao comparar os resultados do modelo

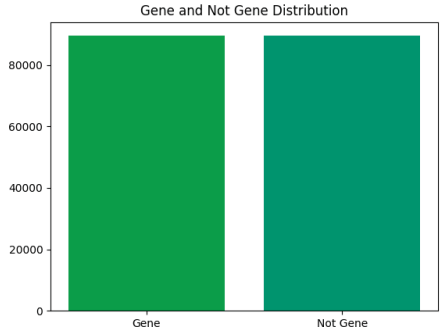


Fig. 30. Balanceamento da quantidade de sequências em cada classe.

TABELA VI
RESULTADOS DA VALIDAÇÃO DO PRIMEIRO TESTE

Modelo	AUC	R2	MSE	Acurácia	Tempo
Árvore de Decisão	0,69	-0,23	0,30	0,69	3,5s
K Vizinhos Próximos	0,83	0,03	0,24	0,73	59s
Rede Neural de Multicamadas	0,72	-0,34	0,33	0,66	266,48s
Máquinas de Vetores de Suporte	0,49	-1,40	0,60	0,40	5830,99s

baseado em SVM, o mesmo apresentou uma acurácia de 40%, ou seja, o modelo não acerta nem metade das predições, o que é um valor baixo levando em consideração o objetivo do trabalho. Além disso o tempo de execução do mesmo foi mais de cem vezes maior que os outros algoritmos. Dessa forma foi optado por não utilizar mais esse modelo nos próximos testes.

Dada as métricas outra informação relevante é o resultado do coeficiente de determinação, que esteve sem muito próximo a 0. Isso indica que não há uma regressão linear sendo utilizada, e por conta disso essa métrica não será mais utilizada para validação. Baseado nos resultados já há um indicador de qual modelo é mais apropriado para a predição desejada. O algoritmo baseado em KNN conseguiu valores melhores em todos os aspectos, exceto o tempo de execução, que os outros modelos.

Com o algoritmo de SVM fora dos testes, o próximo passo na decisão do algoritmo foi reduzir o escopo de variáveis e observar se algum algoritmo se beneficia da mudança e ultrapassa o KNN nas métricas. O escopo de atributos ficou em passar para os modelos o tamanho das sequências, e as quantidades de alguns aminoácidos (Gly, Val, Pro, Gln, Tyr, Glu e Arg) que se mostraram ter mais variações dentro do histograma gerado da base. Com essa mudança os algoritmos tiveram uma queda nas

TABELA VII
RESULTADOS DA VALIDAÇÃO DO SEGUNDO TESTE

Modelo	AUC	MSE	Acurácia	Tempo
Árvore de Decisão	0,65	0,34	0,65	1,38s
K Vizinhos Próximos	0,76	0,29	0,71	27s
Rede Neural de Multicamadas	0,61	0,48	0,51	752,25s

TABELA VIII
RESULTADOS DA VALIDAÇÃO DO TERCEIRO TESTE

Modelo	AUC	MSE	Acurácia	Tempo
Árvore de Decisão	0,66	0,33	0,67	1,76s
K Vizinhos Próximos	0,79	0,27	0,72	75,13s

métricas, porém ganharam em velocidade de processamento como mostra a Tabela VII.

Dentro das métricas obtidas, o modelo de ANN se saiu pior, além de ter demonstrado ter sofrido mais com a diminuição do escopo, o que faz ele não ir para a próxima etapa de testes. Todavia no quesito performance ele teve um destaque muito maior que os outros, executando mais de 20 vezes menos tempo do que no teste anterior.

Por fim o último teste de mudança de escopo é feito adicionando mais aminoácidos (Gly, Val, Pro, Asn, Gln, Tyr, Glu, His, Lys e Arg) que demonstraram não seguir a mesma proporção nas curvas apresentadas pelo gráfico, visando recuperar os valores perdidos nas métricas entre os testes. O ganho nessa execução foi pequeno, todavia serviu para demonstrar que o algoritmo a ser escolhido é o KNN, que obteve mais ganhos percentuais, exceto em tempo de execução, que a árvore de decisão (Tabela VIII).

Outros testes com o aumento de escopo foram feitos, todavia, apesar de ter ganhos nas métricas, o tempo de execução aumentou muito, e a adição de itens no escopo tendeu a ficar muito semelhante com o teste feito com todos os atributos. As Figuras 31 e 32 demonstram o estado do modelo após a redução de escopo.

Tendo um algoritmo escolhido para o modelo, então iniciou-se a mudança de parâmetros para começar a otimizar o modelo. A primeira mudança consistiu em mudar qual o algoritmo interno do modelo de KNN, entre Ball Tree, KD Tree e Brute Force. Foram idênticos em todos os aspectos, exceto que para o Brute Force a área sob a curva ROC foi muito pouco pior e o tempo de execução foi bem mais rápido como apresenta a Tabela IX. Por conta do ganho em execução, então esse último algoritmo foi deixado como padrão para os próximos testes.

Escolhido o algoritmo interno, a próxima padronização foi em relação a função de pesos, entre uniforme e baseada em distância. No peso uniforme todos os pontos são distribuídos

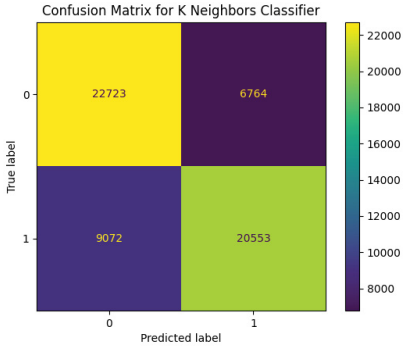


Fig. 31. Matriz de confusão do modelo de KNN após a diminuição do escopo de atributos.

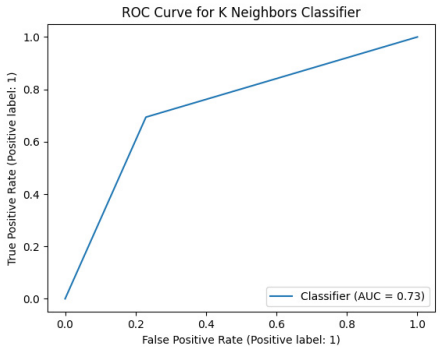


Fig. 32. Área da curva ROC do modelo de KNN após a diminuição do escopo de atributos.

uniformemente entre as vizinhanças. Já no peso baseado em distância os pontos de peso são inversamente proporcionais a distância, ou seja, as vizinhanças mais próximas da consulta em questão terão mais influência do que as mais distantes [48]. Para esse teste o peso baseado em distância obteve resultados um pouco melhores (Tabela X). Entretanto a acurácia ainda permanece igual. Com isso a função de peso baseada em distância permanece para os próximos testes.

O próximo parâmetro selecionado para análise foi o

TABELA IX
RESULTADOS DA VALIDAÇÃO ENTRE ALGORITMOS DE KNN

Algoritmo	AUC	MSE	Acurácia	Tempo
Ball Tree	0,801085	0,26	0,73	621,67s
KD Tree	0,801085	0,26	0,73	121,09s
Brute Force	0,801082	0,26	0,73	58,71s

TABELA X
RESULTADOS DA VALIDAÇÃO ENTRE FUNÇÕES DE PESO DE KNN

Peso	AUC	MSE	Acurácia	Tempo
Uniforme	0,8022	0,2665	0,73	51,41s
Distante	0,8056	0,2659	0,73	44,98s

TABELA XI
RESULTADOS DA VALIDAÇÃO ENTRE DISTÂNCIAS PARA KNN

Distância	AUC	MSE	Acurácia	Tempo
Manhattan	0,8099	0,2637	0,74	192,78s
Euclidiana	0,8094	0,2632	0,74	47,28s

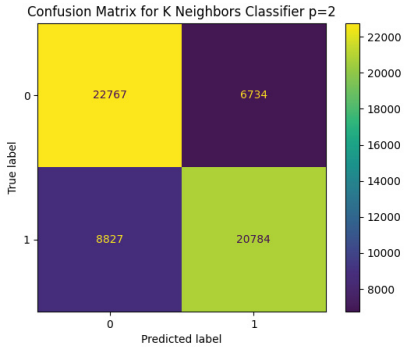


Fig. 33. Matriz de confusão do modelo de KNN utilizando distância euclideana.

parâmetro p. Esse parâmetro é o passado para uma função de distância de Minkowski. Caso $p = 1$, então é usada distância Manhattan, caso $p = 2$ é usada a distancia euclideana. Já para outros valores é usada a função de distância Suprema [48]. Foram executados valores de p de 1 até 3, onde 1 e 2 apesar de terem resultados diferentes, tiveram muito pouca diferença. A distância Manhattan teve um AUC maior, porém a distância euclideana teve um MSE menor. Como em questão de acurácia os dois permaneceram iguais, então foi escolhido o que teve menos falsos positivos (diferença pequena de apenas 4 resultados). Essa escolha se deve ao fato de que o objetivo principal é sinalizar se uma sequencia de bases nitrogenada corresponde a um gene, ou seja, a quem for usar é mais importante que se o contig for categorizado como gene, ele tenha mais chance de realmente ser, com o intuito de não atrapalhar uma possível pesquisa baseada nessa informação. É possível ver a matriz gerada na Figura 33. Em relação ao valor de p = 3, o mesmo não foi executado em tempo hábil e foi descartado para os próximos testes (Tabela XI)

Existe também um parâmetro que diz repeito ao tamanho de folhas da árvore, todavia como o algoritmo escolhido foi o de força bruta, então esse parâmetro não surte nenhum efeito dentro do modelo. Com isso o próximo parâmetro diz respeito a métrica utilizada pelo modelo para validar as distâncias. Essa métrica é similar com a anterior de valor p, porém com essa é possível passar cálculos de distâncias diferentes das citadas anteriormente. Para esse teste são usadas então a distância euclideana (a mesma que já vinha sendo usada), a distância de Chebyshev e a distância de Mahalanobis (as únicas diferentes listadas dentro da documentação). Dependendo do resultado desse teste um novo leque de parâmetros pode ser aberto, pois essas novas distâncias também aceitam parâmetros [48]. Apesar das possíveis possibilidades, o resultado usando outras distâncias como métrica não foi efetivo. A distância de Chebyshev obteve um resultado inferior ao modelo que já estava sendo usado antes, como mostra a Tabela XII, e não foi possível executar o modelo utilizando a distância de Mahalanobis. Ao tentar utilizá-la como métrica foi necessário usar um parâmetro específico para a mesma que pedia um vetor de covariância dos valores para teste. Todavia o framework (Numpy) usado para fazer essa conversão não conseguiu alocar memória suficiente para tal ação, resultando em erro. O ocorrido mostra que para esse problema, utilizar essa distância como métrica exigirá maior poder computacional. Outro ponto a se destacar é que a acurácia do modelo que já estava sendo usado anteriormente diminuiu. Isso se deve a base sempre se embaralhar a cada teste, o que pode impactar aleatoriamente os resultados esporadicamente.

TABELA XII
RESULTADOS DA VALIDAÇÃO ENTRE DISTÂNCIAS PARA KNN

Métrica	AUC	MSE	Acurácia	Tempo
Chebyshev	0,7921	0,2777	0,72	410,238s
Euclideana	0,8045	0,2686	0,73	48,58s

TABELA XIII
RESULTADOS DA VALIDAÇÃO ENTRE NÚMEROS DE VIZINHOS PARA KNN

Vizinhos	AUC	MSE	Acurácia	Tempo
3	0,7809	0,2789	0,72	49,72s
5	0,8066	0,2671	0,73	48,23s
8	0,8207	0,2600	0,74	47,78s
11	0,8272	0,2553	0,74	50,04s
50	0,8329	0,2527	0,75	55,88s

Feita toda a mudança até o momento, o próximo parâmetro diz a respeito de quantos vizinhos próximos o algoritmo irá analisar para reconhecer os padrões. Esse valor, dado como k, teve os seguintes valores passados: 3, 5, 8, 11 e 50. Essa distribuição visa uma tentativa de abordagem local da vizinhança e de uma abordagem genérica. Entretanto a diferença nos resultados não se mostra muito discrepante. Todavia ao utilizar de 50 vizinhos, foi obtido um resultado superior, sendo esse o modelo final. Foram feitos testes com mais vizinhos tentando alcançar resultados melhores, porém o limiar está nos 50 vizinhos, e a partir disso as métricas começam a cair.

O último parâmetro utilizado recebe um número que irá determinar quantos processos paralelos no processador irão executar simultaneamente. caso receba -1 irá executar todos os possíveis. Colocando esse parâmetro como 1 ou -1 não teve efeito em resultados, apenas uma leve vantagem em desempenho quando usando vários processos. De qualquer forma manteve-se como -1 visando utilizar de todo o potencial computacional disponível.

Outra tentativa de melhorar o modelo estabelecido se deu por utilizar formas diferentes de normalizar os dados utilizando outras funções fornecidas pelo scikit-learn. Entretanto todas tiveram resultados muito semelhantes sendo que

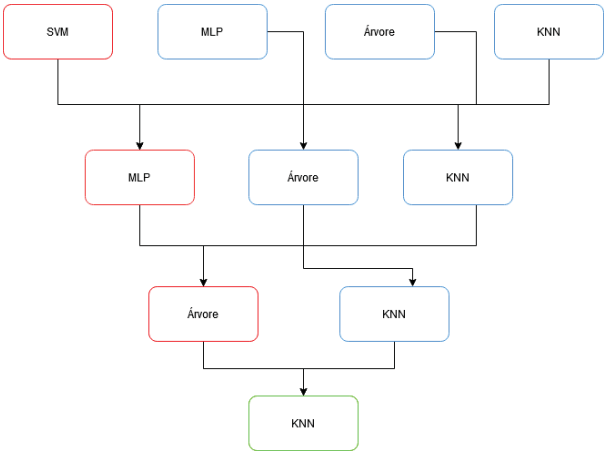


Fig. 34. Disputa entre os modelos na predição de gene.

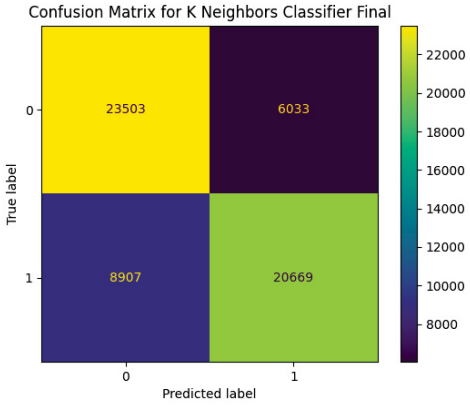


Fig. 35. Matriz de confusão do modelo final gerado.

TABELA XIV
DIFERENÇA DE PARAMETRIZAÇÃO DO KNN

Momento	Parâmetros
Início do experimento	`n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None`
Fim do experimento	`n_neighbors=50, weights='distance', algorithm='brute', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=-1`

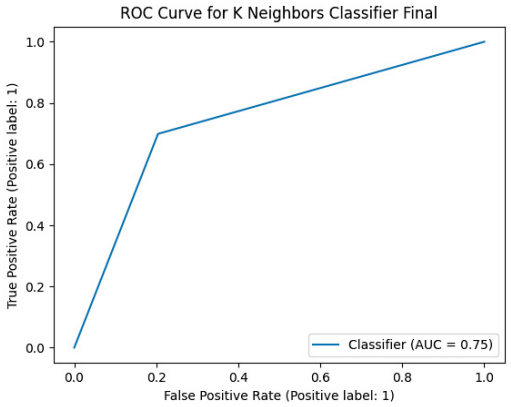


Fig. 36. AUC do modelo final gerado.

normalização entre 0 e 1 usada anteriormente ficou entre as melhores.

Dentro dos quatro algoritmos de IA escolhidos no experimento (SVM, MLP, Árvore de Decisão e KNN), o KNN teve maior aproveitamento da base de dados e em todos os procedimentos obteve métricas melhoras que seus concorrentes. A cada nível de teste um algoritmo saiu até sobrar apenas um, como mostra a Figura 34.

Todavia, mesmo o KNN tendo melhores resultados, com a base e com os atributos extraídos não conseguiu sair da estagnação nos resultados. Ao final dos testes o algoritmo terminou com uma parametrização diferente da qual iniciou. Foi possível diminuir os falsos positivos e aumentar um pouco a acurácia, bem como os outros parâmetros (Figuras 35 e 36)

Usando o arquivo final separado para teste que não foi usado nenhuma vez para treinos o modelo teve uma acurácia de 0,64, demonstrando que o modelo não se comporta tão bem com dados do mundo real. Todavia outros programas de predição gênica também tem dificuldades em identificar casos mais complexos, como sequências muito longas ou muito curtas por exemplo [55]. Isso pode sugerir que mesmo que um resultado geral aparentemente mediano, na verdade o preditor gerado nessa pesquisa não está performando tão longe de outros mais consolidados, visto que a base usada para testá-lo continha genes com graus variados de complexidade.

O resultado final consegue ser satisfatório, já que é possível que alguém consiga utilizar do modelo pra prever genes sem muito esforço e com uma confiança razoável. Também tem como vantagem ser 100% offline, necessitando apenas das bibliotecas do Python. As bibliotecas usadas são bem difundidas na comunidade da linguagem [48] ou são ferramentas já embutidas na própria plataforma [43].

Ademais o trabalho também gerou uma aplicação de KDD, onde os genes podem ser extraídos de maneira simples e rápida. Durante o desenvolvimento, esta aplicação conseguiu processar arquivos de mais de 5 milhões de linhas em torno de 11 a 12 segundos. Ela também conseguiu extrair grande parte dos genes contidos nos arquivos GBFF.

V. Considerações Finais

Este trabalho explorou como a extração de dados e conhecimento e técnicas de inteligência artificial andam em conjunto andam em conjunto. Partindo de um arquivo de texto com milhões de linhas foi possível mostrar que até mesmo os métodos mais simples de guardar dados, podem ter dados organizados e de fácil leitura.

Ademais também conseguiu demonstrar o quanto o genoma pode ser grande entre diferentes espécies, mas mesmo assim ainda ter um padrão que se mostra em todas elas, o que acabou por ser uma das dificuldades em se trabalhar com esse tipo de dado.

Foi possível entender que aonde não se tem boas variáveis explanatórias fica difícil de evoluir com um modelo de inteligência artificial, mas que mesmo assim é possível fazer progressos. Foi com esse progressos que pode-se considerar partir para uma abordagem utilizando KNN em outro trabalho em cima de sequências de DNA.

Com este trabalho fica evidenciado que a manipulação de sequências de DNA demanda de um alto processamento computacional e de um grande estudo das áreas que cercam esse tema. Apesar da pouca evolução em cima do modelo proposto, este trabalho consegue executar o que se propõe além de poder ser um alicerce para outros trabalhos futuramente.

Dentre trabalhos pensados para o futuro estão a melhora do extrator de genes, visto que o algoritmo não conseguiu recuperar todos os genes dos dados baixados devido a disposições diferentes das informações entre os arquivos. Além disso também melhorar a extração de características. Outras formas de extrair esses atributos tem potencial de gerar resultados melhores, inclusive formas que não necessariamente envolvam a bioquímica como feito anteriormente. Também é possível pesquisar a utilização de outras bibliotecas para efeito comparativo, ou mesmo outras plataformas. Por fim dada toda a pesquisa citada anteriormente, refazer o condicionamento dos parâmetros do modelo escolhido, olhando para os caminhos tomados nesse artigos e se aproveitando dos que funcionaram.

Outra alternativa pode ser treinar preditores de genes especialistas. Por exemplo, um preditor de gene apenas para mamíferos, ou ainda mais específico, apenas para cães. Dessa forma o escopo se reduz e pode ser possível explorar uma base mais homogênea.

References

[1] H. VERLI. (2014) Bioinformática: da biologia à flexibilidade moleculares.

[2] A. BAXEVANIS and B. OUELLETTE. (2001) Bioinformatics: A practical guide to the analysis of genes and proteins.

[3] A. C. de Souza Góes and B. V. X. de Oliveira. (2014) Projeto genoma humano: um retrato da construção do conhecimento científico sob a ótica da revista ciência hoje.

[4] E. AGUIAR. (2015) Sequenciamento, montagem e anotação do genoma de streptococcus agalactiae gbs85147: uma abordagem comparativa.

[5] D. GUIZELINI. (2016) G-finisher:uma nova estratégia para refinar e finalizar montagens de genomas bacterianos.

[6] R. SCHALKOFF. (2007) Pattern recognition.

[7] A. RUST, E. MONGIN, and E. BIRNEY. (2002) Genome annotation techniques:new approaches and challenges.

[8] A. LOMSADZE, V. TER-HOVHANNISYAN, Y. CHERNOFF, and M. BORODOVSKY. (2005) Gene identification in novel eukaryotic genomes by self-training algorithm.

[9] A. LIEW, H. YAN, and M. YANG. (2005) Pattern recognition techniques for the emerging field of bioinformatics: A review.

[10] R. GUIGÓ, P. AGARWAL, J. F. ABRIL, M. BURSET, and J. W. FICKETT. (2000) An assessment of gene prediction accuracy in large dna sequences.

[11] D. L. NELSON and M. M. COX. (2019) Princípios de bioquímica de lehninger.

[12] M. YAMAGUCHI and C. O. WORMAN. (2014) Deep-sea microorganisms and the origin of the eukaryotic cell.

[13] A. GRIFFITHS, S. WESSLER, S. CARROLL, and J. DOEBLEY. (2016) Introdução à genética.

[14] A. ZAHA, H. B. Ferreira, and L. M. P. PASSAGLIA. (2014) Biologia molecular básica.

[15] D. NOBLE. (2008) Genes and causation.

[16] (2022) The ncbi website. [Online]. Available: https://www.ncbi.nlm.nih.gov/

[17] P. SIEBER, M. PLATZER, and S. SCHUSTER. (2018) The definition of open reading frame revisited.

[18] M. FURUNO, T. KASUKAWA, R. SAITO, J. ADACHI, H. SUZUKI, R. BALDARELLI, Y. HAYASHIZAKI, and Y. OKAZAKI. (2003) Cds annotation in full-length cdna sequence.

[19] J. ORTEGA and F. SANTOS. (2003) Bioinformática aplicada à genômica.

[20] K. T. BELOZE, A. DAVILA, R. M. ARAUJO, and M. C. CAVALCANTI. (2007) Ampliando o uso colaborativo de ontologias em processos de anotação genômica.

[21] J. GERALDO. (2019) Integração de dados para avaliação da qualidade da anotação dos genes codificadores de proteínas em eucariotos.

[22] M. YANDELL and D. ENCE. (2012) A beginner's guide to eukaryotic genome annotation.

[23] U. FAYYAD, G. PIATETSKY-SHAPIRO, and P. SMYTH. (1996) The kdd process for extracting useful knowledge from volumes of data.

[24] C. ZHANGA and Y. LUB. (2021) Study on artificial intelligence: The state of the art and future prospects.

[25] A. D. VAIO, R. PALLADINO, R. HASSAN, and O. ESCOBAR. (2020) Artificial intelligence and business models in the sustainable development goals perspective: A systematic literature review.

[26] P. M. AMISHA, M. PETHANIA, and V. K. RATHAUR. (2019) Overview of artificial intelligence in medicine.

[27] I. M. ENHOLM, E. PAPAGIANNIDIS, P. MIKALEF, and J. KROGSTIE. (2021) Artificial intelligence and business value: a literature review.

[28] R. Y. CHOI, A. S. COYNER, J. KALPATHY-CRAMER, M. F. CHIANG, and J. P. CAMPBELL. (2020) Introduction to machine learning, neural networks, and deep learning.

[29] S. REED. (1972) Pattern recognition and categorization.

[30] C. M. BISHOP. (2006) Pattern recognition and machine learning.

[31] J. CERVANTES, F. GARCIA-LAMONT, L. RODRÍGUEZ-MAZAHUA, and A. LOPEZ. (2020) A comprehensive survey on support vector machine classification: Applications, challenges and trends.

[32] I. E. NAQA and M. J. MURPHY. (2015) What is machine learning?

[33] A. C. LORENA and A. C. P. L. F. de CARVALHO. (2007) Uma introdução às support vector machines.

[34] W. S. NOBLE. (2006) What is a support vector machine?

[35] S. SUN and R. HUANG. (2010) An adaptive k-nearest neighbor algorithm.

[36] O. KRAMER. (2013) K-nearest neighbors.

[37] B. T. JIJO and A. M. ABDULAZEEZ. (2021) Classification based on decision tree algorithm for machine learning.

[38] H. H. PATEL and P. PRAJAPATI. (2018) Study and analysis of decision tree based classification algorithms.

[39] S. S. HAYKIN. (2001) Redes neurais: Princípios e prática.

[40] B. KRÖSE and P. SMAGT. (1996) An introduction to neural networks.

[41] C. BISHOP. (1995) Neural networks for pattern recognition.

[42] T. W. RAUBER. (2005) Redes neurais artificiais.

[43] (2022) The python website. [Online]. Available: https://www.python.org/

[44] E. BISONG. (2019) Building machine learning and deep learning models on google cloud platform: A comprehensive guide for beginners.

[45] K. D. PRUITT, T. TATUSOVA, and D. R. MAGLOTT. (2007) Ncbi reference sequences (refseq): a curated non-redundant sequence database of genomes, transcripts and proteins.
[46] T. J. CARVER, K. M. RUTHERFORD, M. BERRIMAN, M.-A. RAJANDREAM, B. G. BARRELL, and J. PARKHILL. (2005) Act: the artemis comparison tool.
[47] S. RASCHKA, J. PATTERSON, and C. NOLET. (2020) Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence.
[48] (2022) The scikit-learn website. [Online]. Available: https://scikit-learn.org/
[49] S. VISA, B. RAMSAY, A. RALESCU, and E. van der KNAAP. (2011) Confusion matrix-based feature selection.
[50] M. G. M. C. d. S. L. C. d. O. João Francisco Galera MONICO, Aluir Porfírio Dal PÓZ. (2009) Acurácia e precisão: revendo os conceitos de forma acurada.
[51] M. E. G. MARTINS. (2018) Coeficiente de determinação.
[52] R. G. P. JR and B. PARMENTIER. (2013) Recommendations for using the relative operating characteristic (roc).
[53] G. James, D. Witten, T. Hastie, and R. Tibshirani. (2021) An introduction to statistical learning with applications in r.
[54] R. SUSMAGA. (2004) Confusion matrix visualization.
[55] N. Scalzitti, A. Jeannin-Girardon, P. Collet, O. Poch, and J. D. Thompson. (2020) A benchmark study of ab initio gene prediction methods in diverse eukaryotic organisms.