UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE CIÊNCIAS EXATAS E DA TERRA
DEPARTAMENTO DE INFORMÁTICA E MATEMÁTICA APLICADA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Desenvolvimento de um aplicativo móvel para predição de mutações patogênicas

Daniel Henrique Ferreira Gomes

Natal-RN
Fevereiro de 2022

Daniel Henrique Ferreira Gomes

Desenvolvimento de um aplicativo móvel para predição de mutações patogênicas

Monografia de Graduação apresentada ao Departamento de Informática e Matemática Aplicada do Centro de Ciências Exatas e da Terra da Universidade Federal do Rio Grande do Norte como requisito parcial para a obtenção do grau de bacharel em Ciência da Computação.

Orientador

Prof. Dr. Jorge Estefano Santana de Souza

Universidade Federal do Rio Grande do Norte – UFRN

Departamento de Informática e Matemática Aplicada – DIMAp

Natal-RN

Fevereiro de 2022

Monografia de Graduação sob o título *Desenvolvimento de um aplicativo móvel para predição de mutações patogênicas* apresentada por Daniel Henrique Ferreira Gomes e aceita pelo Departamento de Informática e Matemática Aplicada do Centro de Ciências Exatas e da Terra da Universidade Federal do Rio Grande do Norte, sendo aprovada por todos os membros da banca examinadora abaixo especificada:

Prof. Dr. Jorge Estefano Santana de Souza

Orientador(a)

Instituto Metrópole Digital - IMD

Universidade Federal do Rio Grande do Norte - UFRN

Profa. Dra. Beatriz Stransky Ferreira

Departamento de Engenharia Biomédica - DEB

Universidade Federal do Rio Grande do Norte - UFRN

Profa. Dra. Silvia Maria Diniz Monteiro Maia

Departamento de Informática e Matemática Aplicada - DIMAp

Universidade Federal do Rio Grande do Norte - UFRN

Natal-RN, 4 de fevereiro de 2022.

Para meus familiares e amigos por apoiar minha caminhada acadêmica.

Agradecimentos

À toda a equipe docente que me guiou durante minha jornada acadêmica, em especial aos professores do Instituto Metrópole Digital(IMD) e do Departamento de Informática e Matemática Aplicada(DIMAp).

Ao Programa de Educação Tutorial do curso de Ciência da Computação(PET-CC), do qual faço parte, e ao meu tutor Umberto Costa por todo apoio e suporte dado.

Aos meus familiares e amigos que sempre estiveram ao meu lado me estimulando e apoiando durante toda a graduação.

Ao meu orientador Prof. Dr. Jorge Estefano Santana de Souza por se disponibilizar a realizar a orientação do presente trabalho.

Dans la vie, rien n'est à craindre, tout est à comprendre.

Marie Curie

Desenvolvimento de um aplicativo móvel para predição de mutações patogênicas

Autor: Daniel Henrique Ferreira Gomes
Orientador(a): Prof. Dr. Jorge Estefano Santana de Souza

Resumo

A identificação de mutações patogênicas é um desafio da medicina atual, dessa forma, existem diversos preditores no mercado que possuem precisões diferentes e apresentam resultados diferentes para a mesma mutação, o que pode causar confusão para o médico que busca identificar se uma mutação é ou não patogênica. Utilizar árvores de decisão e algoritmos de aprendizado de máquina supervisionados para realizar essa identificação se mostrou bastante eficiente, porém não existem aplicativos clínicos que utilizem essas técnicas para realizar predição quanto a patogenicidade de uma variante de significância desconhecida (VUS). Dessa forma, este trabalho apresenta o aplicativo DtreePred, um aplicativo multiplataforma, compilado nativamente para Android, iOS e web, que auxilia o usuário na identificação de mutações patogênicas na prática clínica. O aplicativo, permite realizar solicitações de predições de variantes genéticas de maneira intuitiva e rápida.

Palavras-chave: Árvore de decisão, aprendizado de máquina, mutação patogênica, aplicativo móvel, predição, bioinformática.

Development of a mobile application to predict pathogenic mutations

Author: Daniel Henrique Ferreira Gomes
Advisor: PhD. Jorge Estefano Santana de Souza

ABSTRACT

The identification of pathogenic mutations is a real challenge in medicine, therefore, there are several predictors on the market that have different precisions and present different results for the same mutation, which can cause confusion for the physician who seeks to identify whether a mutation is pathogenic or not. . Using decision trees and supervised machine learning algorithms to perform this identification proved to be quite efficient, but there are no clinical applications that use these techniques to predict the pathogenicity of a variant of unknown significance (VUS). Thus, this work presents the DtreePred application, a multiplatform application, natively compiled for Android, iOS and web, which helps the user to identify pathogenic mutations in clinical practice. The application allows you to make requests for predictions of genetic variants in an intuitive and fast way.

Keywords: Decision tree, machine learning, pathogenic mutation, mobile app, prediction, bioinformatics.

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

AM – Aprendizado de Máquina, p. 19

dbnSFP – Database for nonsynonymous SNPs' functional predictions, p.19

ExAC – Exome Aggregation Consortium , p.19

VEP – Variant Effect Predictor, p.23

Sumário

1 Introdução

Segundo Nascimento, et al. (2020) "Um desafio real na medicina de precisão é identificar precisamente quais mutações detectadas de um processo de sequenciamento têm um papel adequado no tratamento ou diagnóstico de uma doença". Embora existam diversos preditores no mercado, para cada mutação há uma discrepância de resultado entre eles, o que pode causar uma confusão na classificação da mutação.

Nascimento, et al. (2020) propuseram uma árvore de decisão com uma precisão maior que os preditores do mercado na identificação de mutações patogênicas. Porém, o processo descrito no artigo é basicamente manual no ponto de vista computacional, dificultando a utilização do processo para pessoas que não tenham conhecimento computacional. Com isso, a criação de um aplicativo que apresente o resultado dos algoritmos de predição atuais, bem como o resultado obtido através da árvore de decisão proposta, auxiliaria bastante os médicos uma vez que “os laboratórios frequentemente utilizam programas de previsão de patogenicidade on-line, juntamente com a literatura tradicional e consultas de banco de dados” (WALTERS-SEN et al., 2014) para a análise de variantes.

1.1 Impacto de variante genética desconhecida

“Uma variante de significância desconhecida (VUS) é uma forma variante de um gene que foi identificado através de testes genéticos, mas cujo significado para a função do organismo não é conhecido.” (NASCIMENTO et al., 2020). Dessa forma, é um verdadeiro desafio para a medicina atual prever o impacto que cada variante.

Com isso, os laboratórios diagnósticos clínicos constantemente encontram o desafio de estabelecer corretamente a ligação de patogenicidade ou benignidade de uma variante específica em genes associados à doenças. Mesmo que a criação de bancos de dados de mutações tenham ajudado a registrar variantes relatadas previamente, a interpretação de uma variante

verdadeiramente desconhecida ainda permanece desafiadora. Conforme o sequenciamento de exoma inteiro se torna algo cada vez mais corriqueiro para a prática clínica, existe o potencial para a identificação de inúmeras variantes de significância desconhecidas que precisariam ser analisadas para verificar o impacto potencial no fenótipo específico do indivíduo (WALTERS-SEN et al., 2014).

Diante do exposto, o presente trabalho visa desenvolver uma ferramenta que auxilie a comunidade médica a basear suas decisões em um laudo clínico/genético.

1.2 Objetivos

O objetivo geral do trabalho é desenvolver um aplicativo para dispositivos móveis multiplataforma que auxilie o médico na identificação de mutações patogênicas na prática clínica.

1.2.1 Objetivos Específicos

Os objetivos específicos do projeto são os seguintes:

1. Construir uma API REST (MASSE, 2011) para receber as requisições do aplicativo e realizar o processamento de forma eficiente;
2. Gerar o resultados dos preditores de mutação patogênica e o resultado da árvore de decisão proposta por Nascimento, et al. (2020);
3. Desenvolver um aplicativo que seja multiplataforma, isto é, funcione tanto no sistema operacional Android, quanto no IOS.
4. Fazer com que o aplicativo além de ser instalado de forma nativa nos smartphones, funcione em navegadores web.
5. Utilizar algoritmos de aprendizagem de máquina para complementar o resultado.

1.3 Metodologias

O presente projeto foi desenvolvido utilizando boas práticas de programação, como o padrão *Repository*, responsável por comunicar a

aplicação com o banco de dados, o padrão *Service*, responsável por conter todas as regras de negócios da aplicação (EVANS, 2003) e o padrão *Front Controller*, o qual centraliza o tratamento de todas as solicitações e que não limita o número de requisições no sistema (BOOCH et al., 2003).

2 Fundamentação teórica

Neste capítulo, serão apresentadas as tecnologias utilizadas para o desenvolvimento do aplicativo para predição de patogenicidade proposto no presente artigo.

Além disso, também serão apresentados princípios teóricos sobre a árvore de decisão proposta por Nascimento, et al. (2020). Também serão elencados os algoritmos de aprendizado de máquina utilizados no projeto.

2.1 Tecnologias utilizadas

Os dispositivos móveis oferecem uma gama de possibilidades para software comercial e pessoal porque são realmente as primeiras plataformas de computação móvel (IVERSEN; EIERMAN, 2013). Dessa forma, a utilização desses dispositivos se tornou mais comum em diversos ambientes de trabalho, como em clínicas médicas.

Portanto, esta seção contém a descrição das tecnologias utilizadas para o desenvolvimento do aplicativo proposto.

2.1.1 Java

Java é uma linguagem de programação de propósito geral com todos os recursos que pode ser usada para desenvolver aplicativos robustos de missão crítica. Atualmente, ela é utilizada não apenas para desenvolvimento *web*, mas também para o desenvolvimento de aplicativos independentes em plataformas em servidores, computadores desktop e dispositivos móveis (LIANG, 2012).

2.1.2 Spring Boot

Spring Boot é uma *framework* opinativo que ajuda os desenvolvedores a criar aplicativos baseados em Spring sem exigir que eles escrevam a mesma configuração padrão repetidas vezes (REDDY, 2017).

O Spring Boot é altamente personalizável e é altamente utilizado na indústria para construção de APIs REST e microsserviços (MASSE, 2011) .

2.1.3 Dart

Dart é uma linguagem de programação orientada a objetos de propósito geral voltada para desenvolvimento web e desenvolvimento para dispositivos móveis (BRACHA, 2015).

2.1.4 Flutter

Flutter é um kit de desenvolvimento de código aberto e amigável ao desenvolvedor criado pelo Google que pode ser utilizado para criar aplicativos para dispositivos móveis Android e iOS, web e desktop através da linguagem de programação Dart. A principal vantagem do flutter é que a partir de um único código fonte, é possível construir um aplicativo para dispositivos móveis, um web app e uma aplicação desktop (ALESSANDRIA; KAYFITZ, 2021).

2.1.5 Python

Python é uma linguagem de programação que pode ser usada para programar em estilo procedural, orientado a objetos e em estilo funcional. Python está sendo altamente utilizada na academia e na indústria por sua facilidade de aprendizagem e por conter inúmeras bibliotecas de terceiros que são totalmente e transparentemente multiplataforma e que possuem diversas aplicações como desenvolvimento web, desktop, jogos e sistemas embarcados (SUMMERFIELD, 2008).

2.1.6 Scikit-learn e Flask

Scikit-Learn é uma biblioteca para Python para aprendizado de máquina. O principal objetivo da biblioteca é fazer com que o desenvolvedor, ao invés de implementar as próprias versões de cada algoritmo, utilize estruturas Python reais prontas para produção (GÉRON, 2017). A versão

utilizada foi a 1.0.1

Já o flask é um *microframework* para Python que possui uma estrutura muito simples, mas altamente extensível. Com isso, os desenvolvedores têm o poder de escolher a configuração que desejam, facilitando a criação de pequenas aplicações, como pequenas API REST (RELAN, 2019). A versão utilizada foi a 2.0.2.

2.1.7 DbNSFP

O dbNSFP é um banco de dados desenvolvido para previsão funcional das variantes de nucleotídeo único não sinônimas no genoma humano (LIU et al., 2020).

Esse banco compila a predição de 37 algoritmos além de algumas informações relacionadas, como as frequências alélicas observadas no Projeto 1000 genomas e dados do consórcio ExAC.

2.2 Algoritmos de Aprendizado de máquina

Foram utilizados os algoritmos de aprendizado de máquina implementados pelo scikit-leaning e as bases de dados geradas por Nascimento, et al. (2020) com o intuito de treinar 15 modelos. Esses modelos são utilizados pela API REST desenvolvida para prever a patogenicidade das variantes genéticas passadas pelos usuários.

Os 15 algoritmos de AM utilizados pela API são: *AdaBoostClassifier*, *BaggingClassifier*, *ExtraTreesClassifier*, *RandomForestClassifier*, *LogisticRegression*, *BernoulliNB*, *GaussianNB*, *DecisionTreeClassifier*, *KNeighborsClassifier*, *MLPClassifier*, *LinearSVC*, *NuSVC*, *SVC*, *LinearDiscriminantAnalysis* e *QuadraticDiscriminantAnalysis*. O Anexo A apresenta uma tabela desenvolvida por Nascimento, et al. (2020) no qual apresenta a acurácia desses algoritmos para a identificação de mutações patogênicas.

2.3 Árvore de decisão para a identificação de mutações patogênicas

Nascimento, et al. (2020) propuseram um modelo de árvore de decisão para melhorar a identificação de mutações patogênicas visto que os preditores comumente utilizados pela comunidade científica apresentam resultados conflitantes. Para isso, foi selecionado um conjunto de 9 preditores, todos disponíveis no banco dbNSFP(LIU et al., 2020).

A utilização dessa árvore se mostrou bastante eficaz chegando a uma acurácia de 91%, além disso, ela serve como um guia decisório quando os preditores tradicionais apresentam dados conflitantes, isto é, quando não há um consenso quanto a variante é ou não patogênica.

No primeiro nível da árvore, ilustrada na Figura 1, caso os preditores SIFT (KUMAR; HENIKOFF; NG, 2009), Polyphen (ADZHUBEI et al., 2010) e PROVEAN (CHOI; CHAN, 2015) classifiquem a mutação como neutra, a classificação final da mutação é neutra. Caso contrário, no segundo nível da árvore, se a frequência alélica no ExAC for inferior a 0.0001, a mutação é classificada como patogênica. O terceiro nível da árvore classifica a mutação como neutra se pelo menos 3 dos 9 preditores apresentarem esse resultado (essa informação é representada pela variável NDAMAGE na Figura 1).

Por fim, no último nível da árvore, se a mutação tiver alguma frequência alélica no projeto 1000 genomas, a classificação a árvore classifica a mutação como neutra, caso contrário, a classifica como patogênica (essa informação é representada pela variável COMMON na Figura 1).

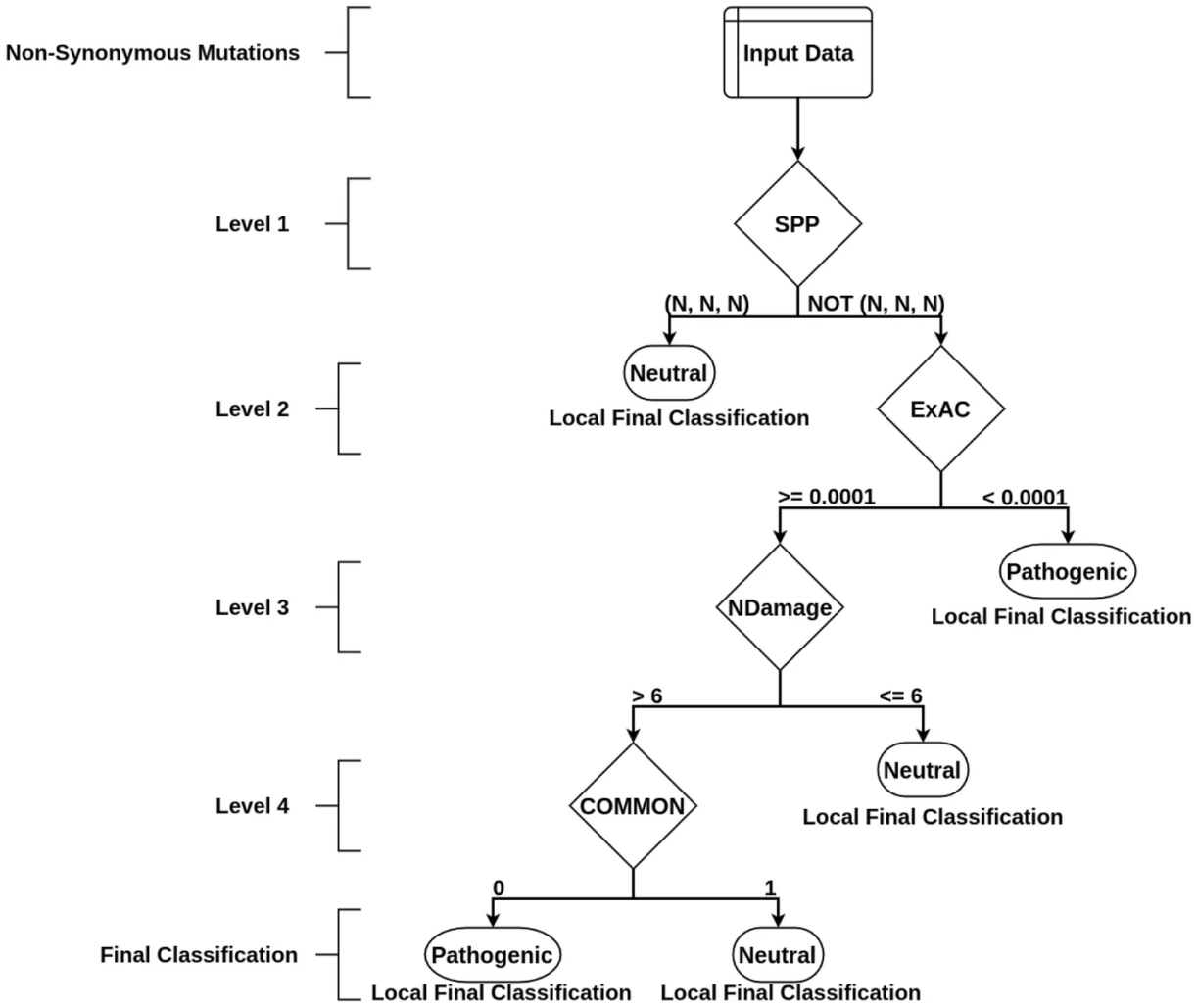


Figura 1 – Árvore de decisão para identificação de mutações patogênicas.

Fonte – Retirado de Nascimento, et al. (2020).

3 Revisão de sistemas de informação para a identificação de mutações patogênicas

O presente capítulo apresenta alguns sistemas de informações que possuem o intuito de prever o impacto das variantes genéticas.

3.1 dbNSFP *web service*

Liu, et al. (2020) desenvolveram uma página web na qual é possível a partir de “uma ou várias coordenadas de genoma (cromossomo, posição, alelo de referência e alelo alternativo), os usuários podem facilmente recuperar todas as colunas de anotação em dbNSFP”. Dessa forma, conforme apresentado nas Figuras 2 e 3, é possível ver os resultados dos preditores contidos no banco dbNSFP em uma página web.

Uma desvantagem desse visualizador é que as predições solicitadas não ficam armazenadas, sendo necessário o download do arquivo com os resultados. Além disso, o site não possui responsividade para ser utilizado em dispositivos móveis.

É válido ressaltar ainda que o visualizador não apresenta predições baseadas em algoritmos de AM e não fornece um guia decisório para dados conflitantes. Ou seja, caso os preditores apresentem resultados diferentes um dos outros, o visualizador não indica para o usuário uma decisão acerca da natureza da mutação.

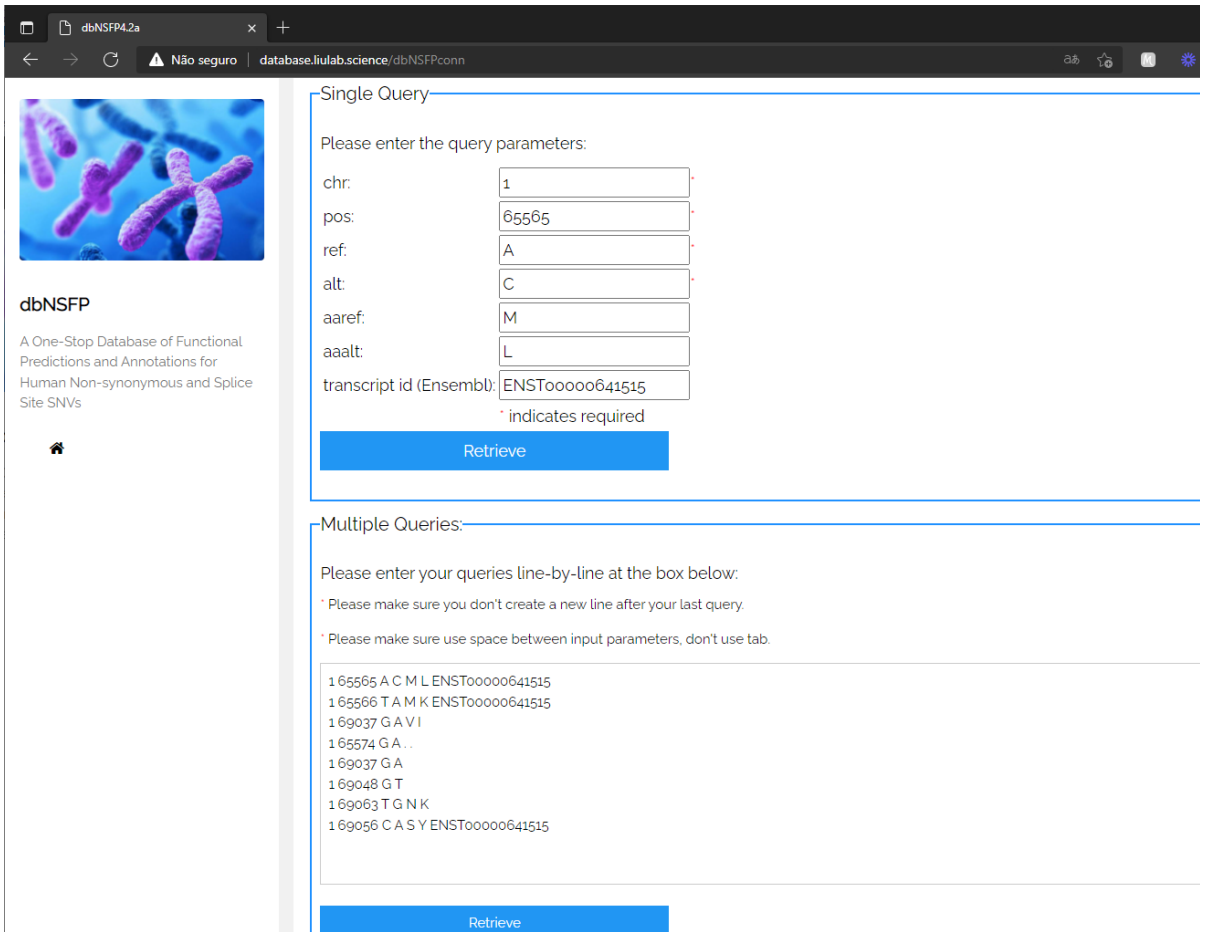


Figura 2 – Captura de tela do dbNSFP *web service*.

Na Figura é possível visualizar que os dados de entrada são: o cromossomo, a posição da variante, os nucleotídeos de referência e alternativos, aminoácidos referência e alternativos, ambos sendo opcionais, e o id da transcrição, também sendo opcional.

Fonte – Elaborado pelo autor.

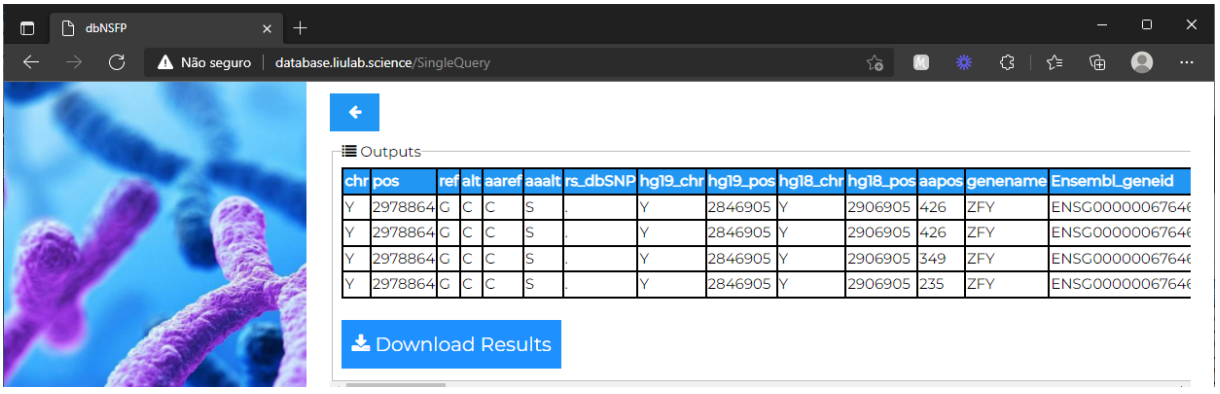


Figura 3 – Resultado obtido pelo *web service*.

No dbNSFP *web service* O resultado é disponibilizado em formato de tabela, podendo ainda ser baixado em formato vcf.

Fonte – Elaborado pelo autor.

3.2 *Ensembl Variant Effect Predictor*

O visualizador *Ensembl Variant Effect Predictor* (Vep Web) segundo McLaren, et al. (2016) “oferece uma interface simples de ponto e clique. Este é ideal para explorar a anotação de forma interativa. O portal é mais adequado para análises de uso pela primeira vez ou em pequena escala.”. Como ilustrado nas Figura 4 e 5, a partir de um arquivo vcf, o sistema retorna o resultado uma tabela de visualização contendo hiperlinks para genes, transcrições, recursos regulatórios e variantes a predição do impacto daquela variante na proteína com o resultado dos preditores contidos no dbnSFP.

Assim como o dbNSFP, o VEP Web também não apresenta predições baseadas em algoritmos de AM e não também não fornece um guia decisório para dados conflitantes. Além disso, o portal também não possui responsividade para visualização em dispositivos móveis.

Por outro lado, ao contrário do dbNSFP Web service, o VEP WEB armazena o resultado da predição do impacto das variantes solicitadas pelo usuário, porém o visualizador não apresenta o resultado de predição da árvore de decisão proposta por Nascimento, et al. (2020), que mostrou ter uma acurácia maior que os preditores contidos no banco dbNSFP.

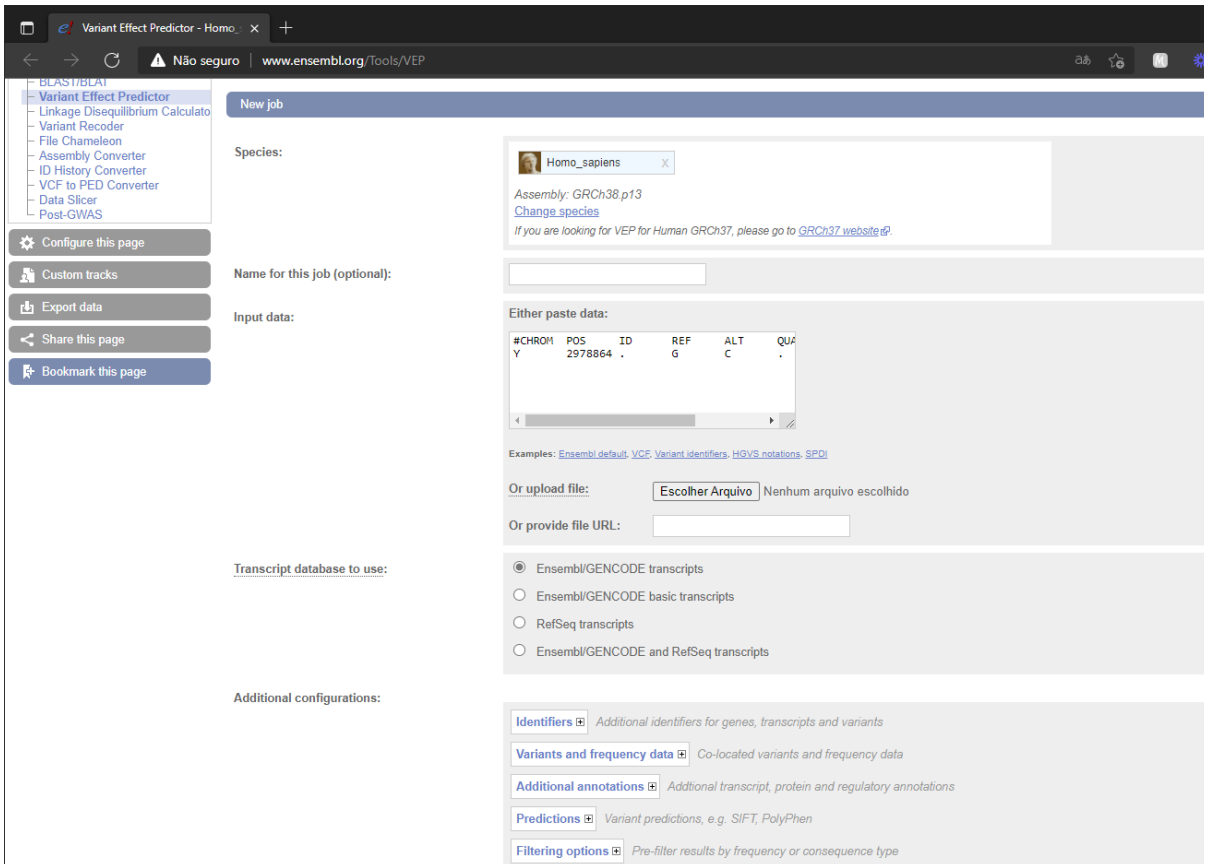
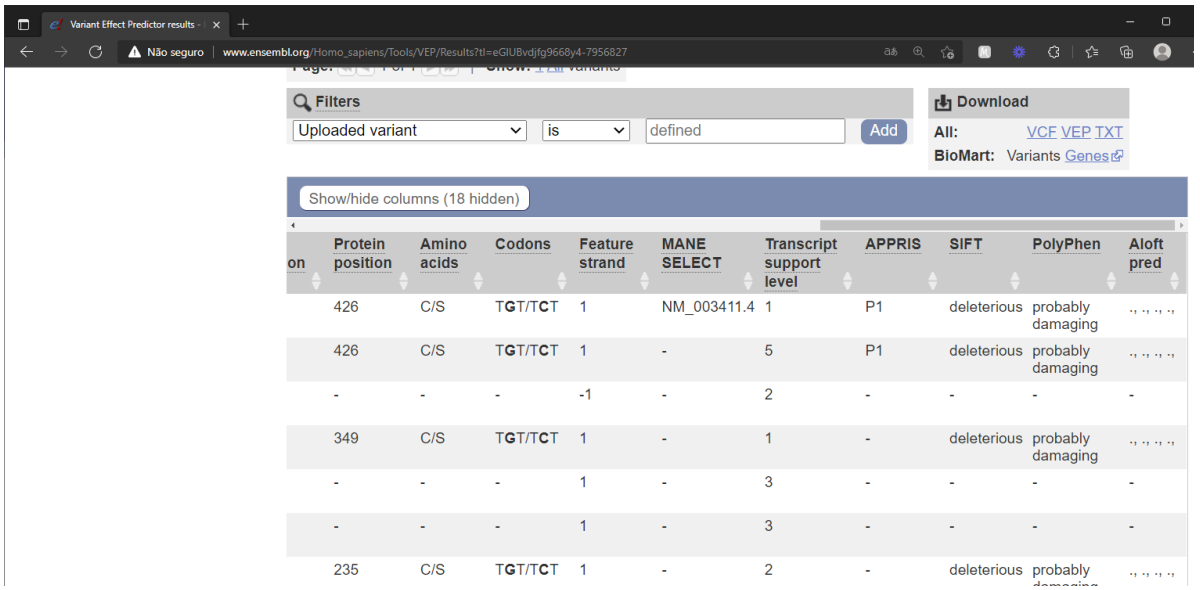


Figura 4 – Captura de tela do Vep Web.

No Vep Web o dado de entrada é um arquivo em formato vcf. Ele contém ainda um menu com opções para escolher os preditores a serem utilizados.

Fonte – Elaborado pelo autor.



on	Protein position	Amino acids	Codons	Feature strand	MANE SELECT	Transcript support level	APPRIS	SIFT	PolyPhen	Aloft pred
	426	C/S	TGT/TCT	1	NM_003411.4	1	P1	deleterious	probably damaging	., ., ., .,
	426	C/S	TGT/TCT	1	-	5	P1	deleterious	probably damaging	., ., ., .,
	-	-	-	-1	-	2	-	-	-	-
	349	C/S	TGT/TCT	1	-	1	-	deleterious	probably damaging	., ., ., .,
	-	-	-	1	-	3	-	-	-	-
	-	-	-	1	-	3	-	-	-	-
	235	C/S	TGT/TCT	1	-	2	-	deleterious	probably	., ., ., .,

Figura 5 – Resultado obtido pelo Vep Web.

No Vep Web, o resultado é disponibilizado em formato de tabela.

Fonte – Elaborado pelo autor.

3.3 PROVEAN *web server*

Choi e Chan (2015) apresentaram uma página web, conhecida como PROVEAN *web server* que “fornece uma análise rápida das variantes proteicas de qualquer organismo, e também suporta análises de alto rendimento para variantes humanas e de camundongos nos níveis genômico e proteico.”.

A Figura 6 apresenta a tela inicial do PROVEAN *web server*, nela é possível observar que para realizar a solicitação de predição, o usuário precisa preencher um formulário simples no seguinte padrão: <cromossomo>,<posição>,<alelo referência>,<alelo variante>,<comentário>.

Após a submissão, o usuário deve aguardar o término do processamento da sua predição, como mostra a Figura 7. Quando o servidor terminar de processar a predição, o usuário tem três maneiras de visualizar, conforme apresenta a Figura 8. No caso, é possível visualizar uma tabela detalhada do processamento, uma tabela condensada e um resumo geral do resultado por até 48 horas depois da solicitação.

O resultado do processamento pode ser visto na Figura 9, na informa se o preditor PROVEAN classificou a variante como neutra ou deletéria e se o preditor SIFT classificou a variante como tolerada ou prejudicial.

Assim como o dbNSFP e o VEP *Web*, o PROVEAN também não apresenta predições baseadas em algoritmos de AM, não também não fornece um guia decisório para dados conflitantes e não é responsivo para visualização em dispositivos móveis.

Além disso, o portal só apresenta os resultados dos preditores SIFT e PROVEAN, não contendo o restante dos preditores contidos no dbNSFP.

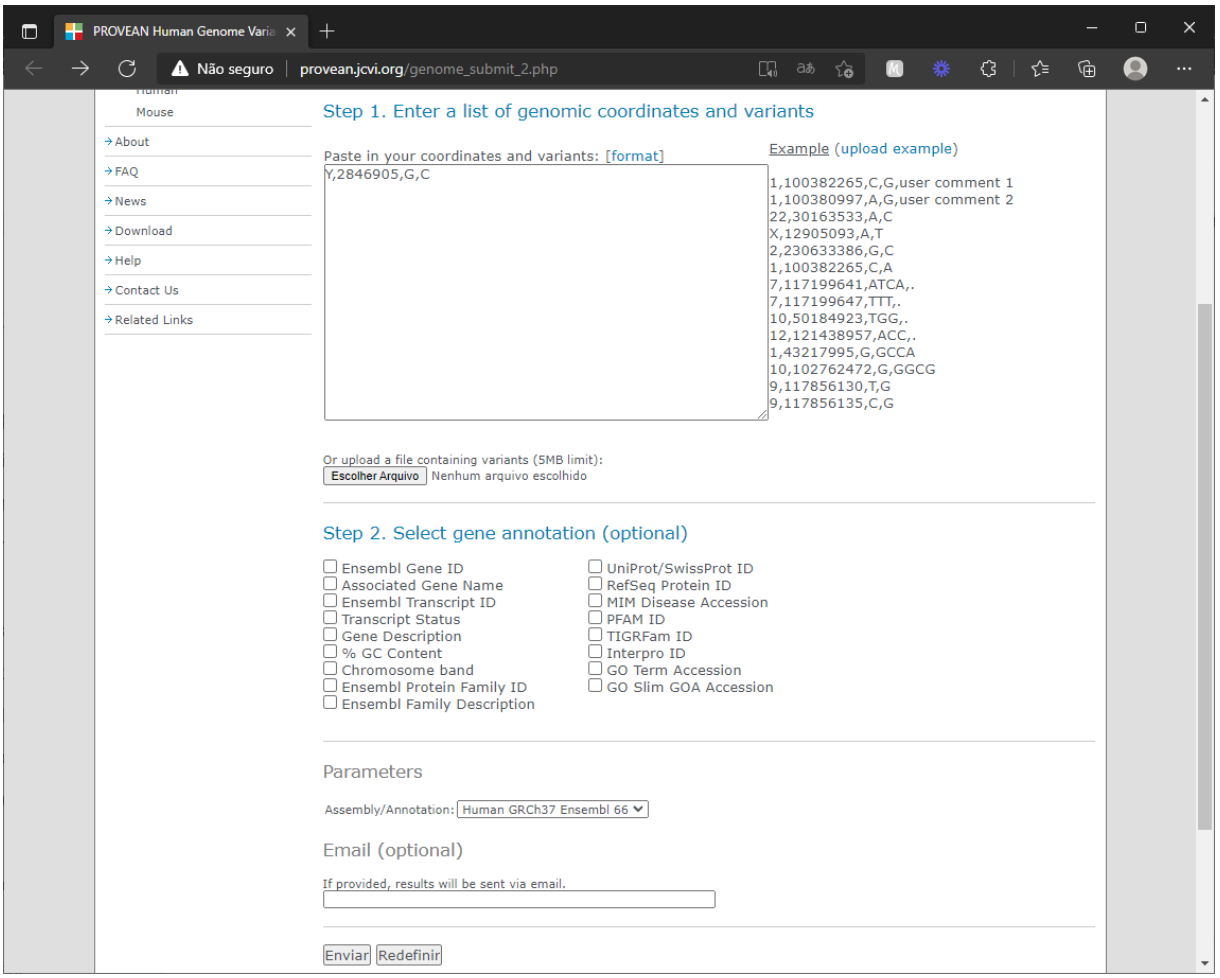


Figura 6 – Tela inicial do PROVEAN *web server*.

No PROVEAN web server a solicitação de predição deve ser feita utilizando o padrão <cromossomo>,<posição>,<alelo referência>,<alelo variante>,<comentário>.

Fonte – Elaborado pelo autor.

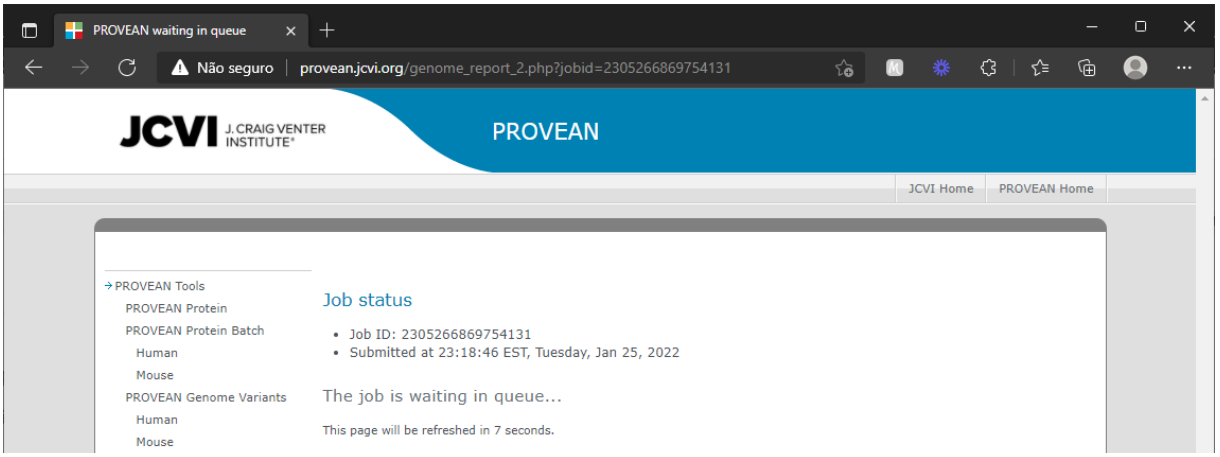


Figura 7 – Tela de espera após a solicitação no PROVEAN *web server*.

Fonte – Elaborado pelo autor.

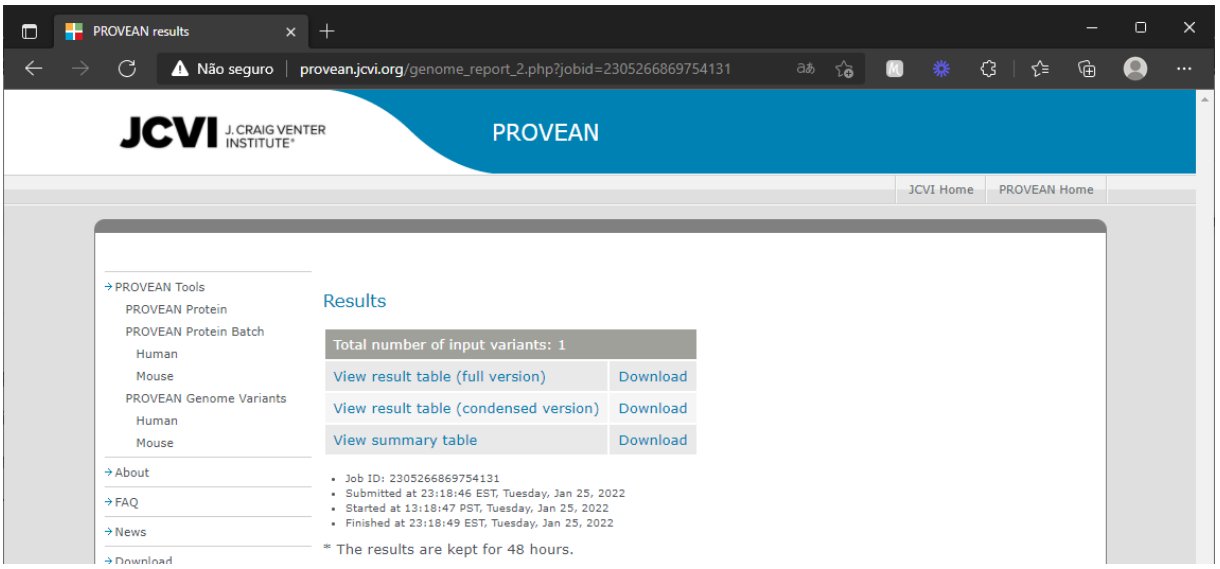


Figura 8 – Tela após a conclusão do processo pelo PROVEAN *web server*.

Fonte – Elaborado pelo autor.

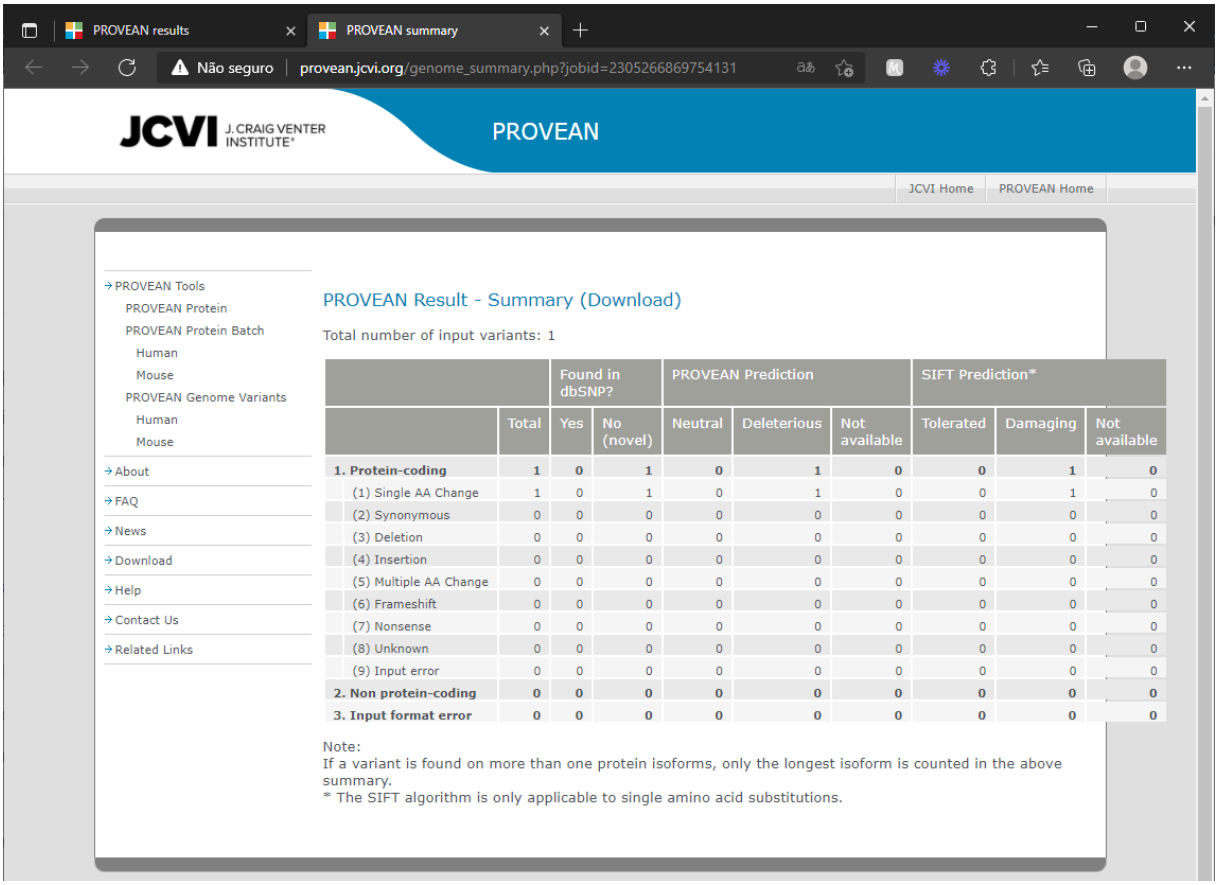


Figura 9 – Predição com os algoritmos PROVEAN e SIFT no PROVEAN *web server*.

No PROVEAN web server o resultado da predição vem em formato de tabela, mostrando lado a lado o resultado do preditor SIFT e do preditor PROVEAN.

Fonte – Elaborado pelo autor.

3.4 Comparação entre os sistemas

A Tabela 1 apresenta uma comparação entre as funcionalidades presentes e não presentes nos sistemas apresentados com o sistema proposto. Nela, é possível observar que as tecnologias atuais embora cumpram o papel proposto, elas possuem diversas limitações tecnológicas que vão desde a falta de adaptabilidade para dispositivos móveis, até a falta de novas técnicas de técnicas de predição como a árvore de decisão proposta por Nascimento, et al. (2020) e algoritmos supervisionados de aprendizados de máquina.

Além disso, também é válido ressaltar que nenhum dos sistemas apresentados apresenta um guia decisório para conflitantes, diferentemente do DtreePred, que apresenta o resultado da árvore de decisão para isso.

Tabela 1 – Comparação das funcionalidades dos sistemas.

Funcionalidades	dbNSFP Web Service	VEP Web	PROVEAN Web	DtreePred
Apresenta os resultados dos algoritmos de predição contidos nos banco dbNSFP.	X	X		X
Armazena as solicitações solicitadas pelos usuários.		X		X
Genoma de referência humano versão GRCh38 - hg38 (CONSORTIUM, 2013).	X	X		X
Genoma de referência humano versão GRCh37 - hg19 (CONSORTIUM, 2009).	X	X	X	
Responsivo para dispositivos móveis.				X
Dispõe de um aplicativo nativo para dispositivos móveis.				X
Apresentam predições baseadas em algoritmos de AM.				X
Fornece guia decisório para dados conflitantes?				X

Fonte – Elaborado pelo autor.

4 O aplicativo DtreePred

O seguinte capítulo apresentará os requisitos do aplicativo desenvolvido, tanto os funcionais quanto os não funcionais. Além disso, detalhes do aplicativo desenvolvido e das APIs Rest desenvolvidas também serão elencados.

4.1 Acesso eficiente ao dbNSFP e aos algoritmos de aprendizado de máquina

Por existirem diversos preditores comumente utilizados na prática clínica, se tornou necessário a criação de um banco de dados unificado que sumarize o resultados deles, o dbNSFP. Entretanto, isso culminou em uma base de dados enorme, fazendo que uma busca linear simples, por exemplo, se torne algo inviável.

Uma maneira para solucionar o problema de eficiência é inserir esse dado em um banco de dados tradicional, porém isso também não é o ideal, porque algoritmos genéricos de indexação de banco de dados não são especializados para dados biológicos. Dessa forma, a maneira mais eficiente encontrada para a foi a utilização da ferramenta Tabix, um sistema capaz de recuperar rapidamente recursos sobrepostos a regiões cromossômicas específicas (LI, 2011).

Portanto, com um simples comando de terminal, no caso, “tabix dbNSFP4.1a.txt.gz <chr>:<pos>-<pos>”, o sistema consegue rapidamente obter o resultado de predição de uma determinada variante, contida no banco dbNSFP em um formato delimitados por TAB, no caso o VCF(*Variant Call Format*). Dessa forma o DtreePred consegue rapidamente obter o resultados dos preditores clássicos e gerar a predição da árvore de decisão proposta por Nascimento, et al. (2020).

O desenvolvimento de técnicas computacionais de aprendizado de máquina especializadas em reconhecer e associar padrões de bases de

dados, torna capaz de proporcionar maiores taxas de correção ao analisar mutações (NASCIMENTO et al., 2020). Porém, se a cada nova variante encontrada fosse treinado um novo modelo fosse gerado, o custo computacional seria muito elevado. Para obter uma predição rápida e eficiente foi previamente treinado um modelo para cada algoritmo disponível no scikit-learn, fazendo com que a cada solicitação de predição, cada um desses modelos retornem sua predição de maneira imediata. Portanto, um dos diferenciais do aplicativo desenvolvido é que ele é capaz de apresentar os resultados de predição desses modelos de AM.

O Anexo C apresenta o algoritmo de treinamento dos modelos treinados e os modelos se encontram disponíveis na seguinte url: https://drive.google.com/drive/folders/1jbfeTNJosYRkyJo62rgM52DUpNED7VHK?usp=sharing

4.2 Requisitos funcionais

Segundo Sommervill (2011) requisitos funcionais “são declarações de serviços que o sistema deve fornecer, de como o sistema deve reagir a entradas específicas e de como o sistema deve se comportar em determinadas situações.”

Com isso, a Tabela 2 apresenta a lista com os requisitos funcionais do aplicativo DtreePred.

Tabela 2 – Tabela de requisitos funcionais.

Nome	Descrição
Consultar a predição da árvore de decisão	Será possível consultar a predição da patogenicidade de uma variante genética a partir da árvore decisão.
Consultar a predição dos algoritmos ML	Será possível consultar a predição da patogenicidade de uma variante genética a partir dos algoritmos de ML.
Excluir predições	Será possível excluir as solicitações de predições.
Gerenciamento de usuários	O aplicativo permitirá que os usuários se cadastrem e se conectem ao sistema.
Solicitação de predições	Será permitido que os usuários solicitem a patogenicidade de uma variante genética.

Visualização do NDAMAGE dos preditores	Visualização da quantidade de preditores contidos no banco dbNSFP que informaram que a variante é patogênica.
Visualização do NDAMAGE dos algoritmos de ML	Visualização da quantidade de algoritmos de ML que informaram que a variante é patogênica.

Fonte – Elaborado pelo autor.

4.3 Requisitos não funcionais

Sommervill (2011) afirmou que “os requisitos não funcionais, como o nome sugere, são requisitos que não estão diretamente relacionados com os serviços específicos oferecidos pelo sistema a seus usuários”.

Dessa forma, a Tabela 3 apresenta a lista de requisitos não funcionais contidos no aplicativo DtreePred.

Tabela 3 – Tabela de requisitos não funcionais.

Nome	Descrição
Eficiência	As predições da patogenicidade das mutações solicitadas pelos usuários serão executadas de forma eficiente e imediata pelo sistema.
Extensibilidade	O sistema será capaz de ser atualizado para novas versões do banco dbNSFP.
Facilidade de uso	O aplicativo será de fácil utilização para qualquer usuário.
Privacidade	Apenas os próprios usuários terão acesso a suas predições.
Segurança	Apenas os usuários aprovados obterão acesso ao aplicativo.

Fonte – Elaborado pelo autor.

4.4 DtreePred

O aplicativo DtreePred é um aplicativo desenvolvido para a identificação de mutações patogênicas para ser utilizado na prática clínica. Para isso foram desenvolvidas um aplicativo mobile multiplataforma e duas API REST que são consumidas pelo aplicativo, com o intuito processar de maneira eficiente a solicitação do usuário.

4.4.1 Arquitetura da comunicação do aplicativo com os serviços

A Figura 10 ilustra a arquitetura de comunicação do aplicativo com o

back-end. O servidor Spring boot é o back-end principal do sistema, ele é o único que se comunica diretamente com o aplicativo. Além disso, ele é responsável por se comunicar com o banco dbNSFP para obter o resultados dos vários preditores contidos nele. A API também é responsável por se comunicar com a API secundária, desenvolvida em Flask, com o intuito de obter o resultados da predição dos algoritmos de Aprendizado de Máquina.

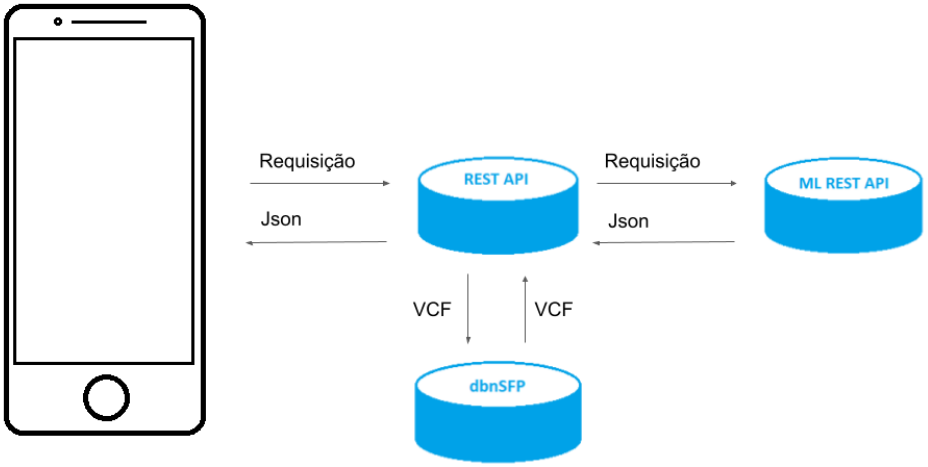


Figura 10 – Arquitetura de comunicação do app com as APIs.

Na arquitetura utilizada (1) o aplicativo envia as solicitações via http para o servidor Spring, (2) O servidor Spring se comunica com o banco dbnSFP para obter o resultado os preditores, (3) O servidor Spring se comunica com o servidor Flask para obter o resultado dos algoritmos de AM, (4) O servidor Spring responde o aplicativo por meio do protocolo http e no formato json.

Fonte – Elaborado pelo autor.

4.4.2 Spring boot API Rest

A API Rest desenvolvida em spring é responsável por se comunicar com o aplicativo, tendo ela a tarefa de realizar o gerenciamento de usuários e o gerenciamento das predições. A Tabela 4 apresenta todas as rotas desenvolvidas e que são utilizadas pelo aplicativo.

Tabela 4 – Tabela de rotas da API principal.

Tipo de requisição	Rota	Descrição
POST	sign-in/	Acesso de usuários ao sistema.
POST	sign-up/	Cadastro de usuários no sistema.
GET	user/list/	Lista todos os usuários do sistema (apenas para administradores).

GET	user/	Retorna os dados do usuário solicitante.
POST	user/activate/{id}	Ativa usuários do sistema (apenas para administradores).
GET	validation/	Valida o email do usuário.
POST	predict/process/	Processa a solicitação de predição.
GET	predict/results/	Obtém todas as predições ordenadas e paginadas.
GET	predict/results/list/	Obtém todas as predições ordenadas e em formato de lista.
GET	predict/allPretictiors/{id}	Obtém uma determinada predição.
DELETE	predict/delete/{id}	Excluir uma predição solicitada.

Fonte – Elaborado pelo autor.

A rota *predict/process/* se comunica com o banco dbNSFPv4.1 para obter o resultado de predição de todos os algoritmos de predição contidos nele, para aí poder gerar o resultado da árvore de decisão, conforme ilustra o Apêndice A.

4.4.3 Flask API Rest

A API Rest desenvolvida em Flask tem a única tarefa de retornar o resultado de predição de cada um dos 15 modelos de aprendizado de máquina previamente treinados. A Tabela 5 apresenta a rota disponível pela API. Além disso, o Apêndice B apresenta o código fonte dessa API.

Tabela 5 – Tabela de rotas da API secundária.

Tipo de requisição	Rota	Descrição
POST	results/	Retorna o resultado de predição dos 15 algoritmos de AM.

Fonte – Elaborado pelo autor.

4.4.4 Versão final do aplicativo

O aplicativo DtreePred foi a solução computacional desenvolvida para ser utilizada na prática clínica com o intuito de obter a predição do impacto de uma variante genética, principalmente quanto a sua patogenicidade.

A Figura 11 ilustra a tela de login do sistema DtreePred. Essa é a primeira tela que aparece para todos os usuários que instalarem o aplicativo, ou que acessarem o webapp pela seguinte URL: https://danhfg.github.io/.

Para poder acessar o sistema, o usuário primeiro deve realizar o cadastro clicando no botão cadastrar-se, depois preencher o formulário de cadastro, apresentado na Figura 12, e por fim verificando o email de confirmação enviado para o email cadastrado.

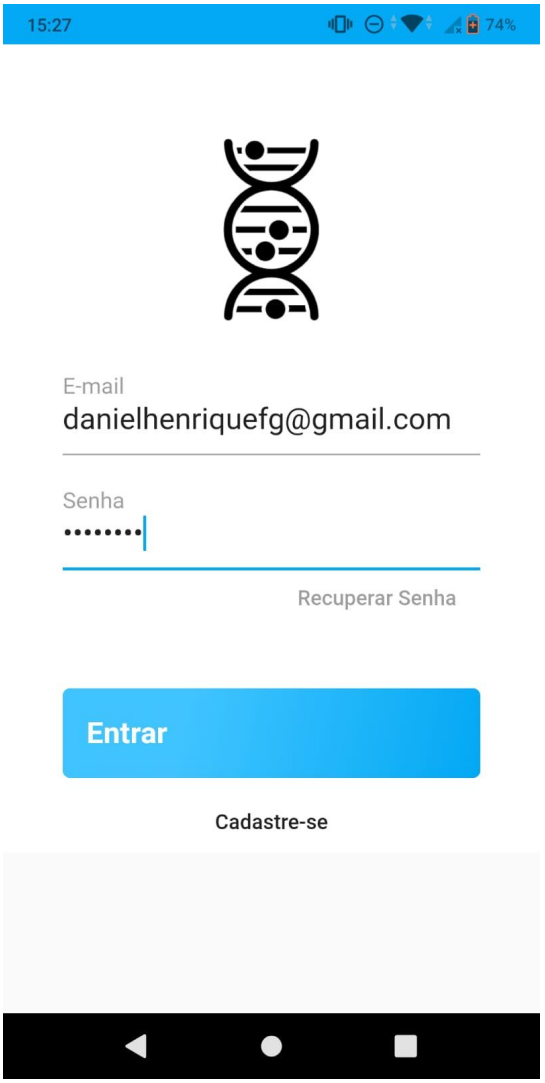


Figura 11 – Tela de Login no DtreePred.

Fonte – Elaborado pelo autor.

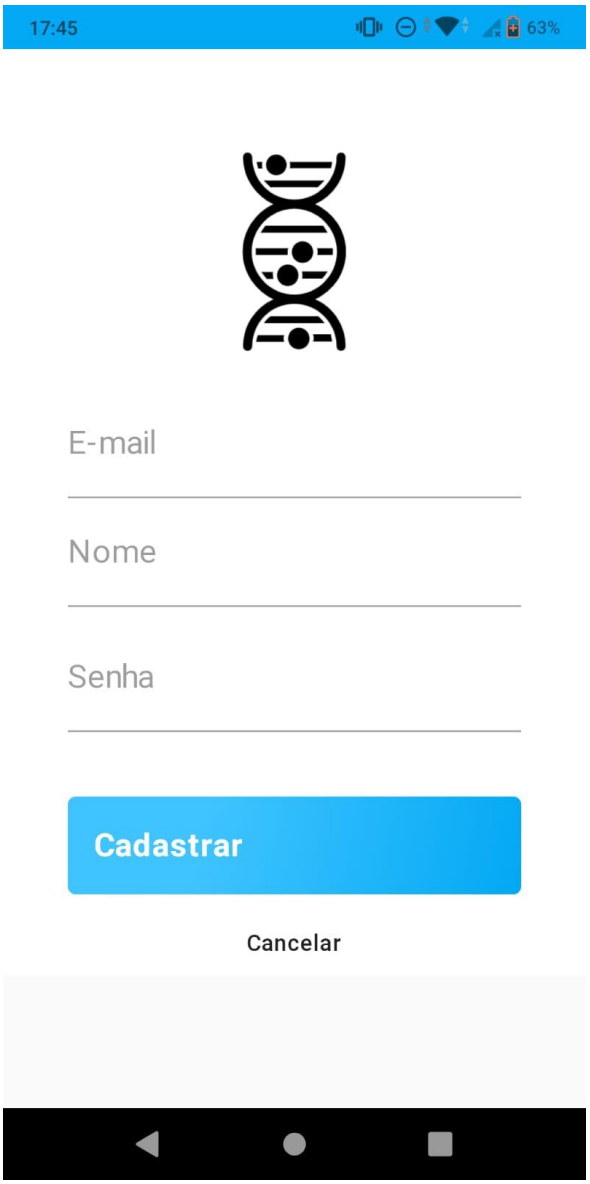


Figura 12 – Tela de Cadastro no DtreePred.

Fonte – Elaborado pelo autor.

Assim que efetuar o acesso ao sistema, o usuário pode se deparar com duas telas. Caso ele não tenha realizado nenhuma solicitação de predição para o sistema, uma mensagem informando que não possui nenhuma solicitação cadastrada é informada, conforme mostra a Figura 13.

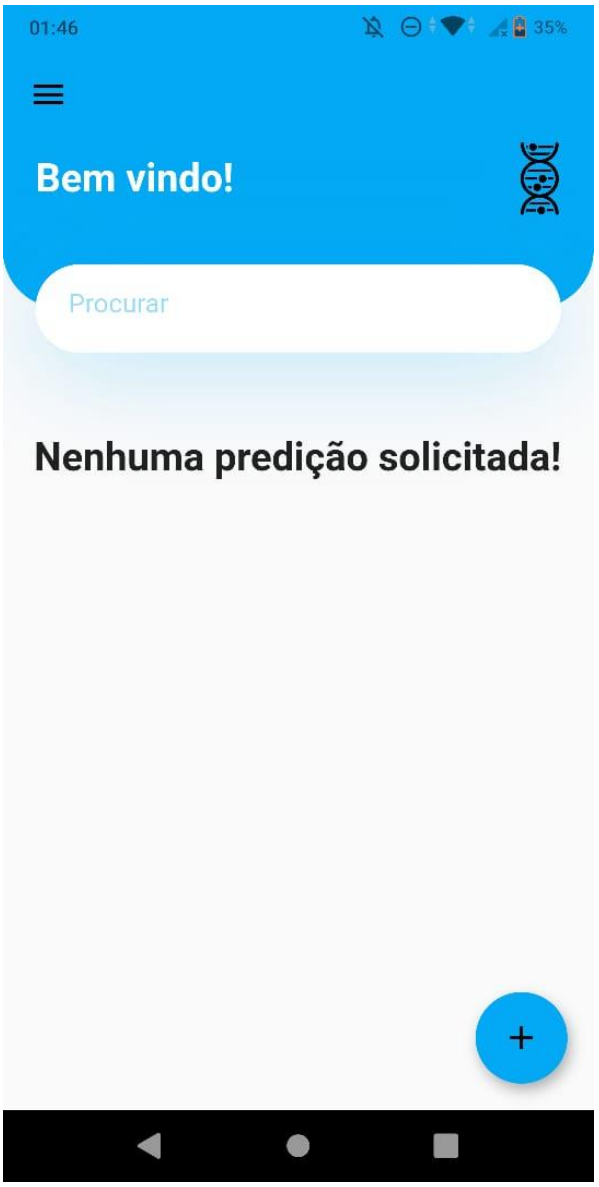


Figura 13 – Tela de visualização das predições.

Fonte – Elaborado pelo autor.

Por outro lado, caso o usuário já tenha solicitado predições previamente, todas elas são mostradas na tela, como apresenta a Figura 14. O resultado de cada uma das solicitações é dividido primeiramente com as informações passadas pelo usuário, como a posição da variante, o nucleotídeo alternativo encontrado, o nucleotídeo referência e a qual cromossomo pertence aquela mutação. Abaixo dessas informações é possível encontrar a predição para essa mutação, caso a árvore de decisão a preveja como neutra, é apresentado para o usuário uma letra “N” verde, por outro lado, caso a árvore preveja que a mutação seja patogênica, o usuário

visualiza uma letra “P” vermelha.

O aplicativo apresenta ainda algumas informações adicionais, sendo elas a quantidade de preditores contidos no dbNSFP que informaram que essa variante é uma mutação patogênica, e a quantidade de algoritmos de aprendizado de máquina que também informaram que a mutação é patogênica.

Além do mais, o aplicativo permite que o usuário exclua alguma predição após pressionar um botão cinza com a figura de uma lixeira.

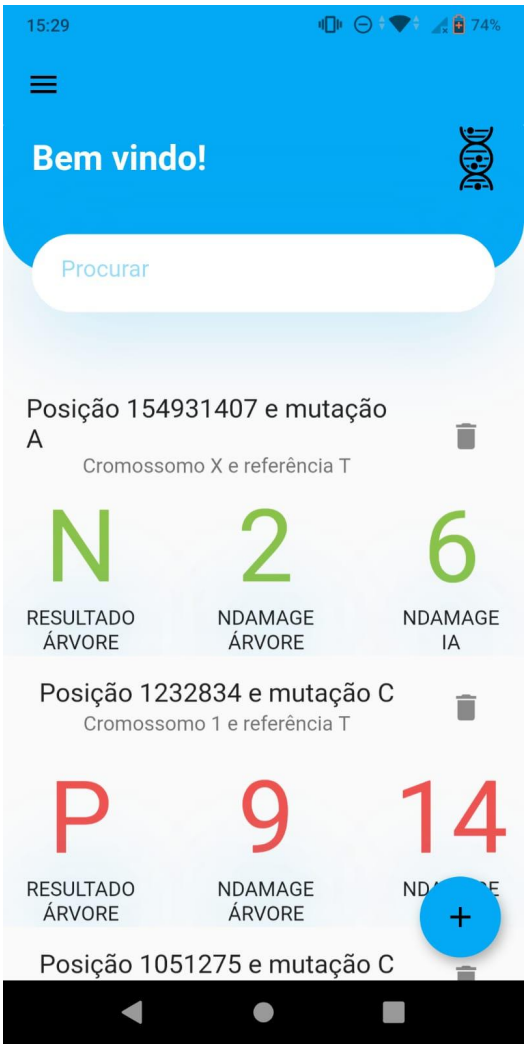


Figura 14 – Tela de visualização dos resultados das predições.

Fonte – Elaborado pelo autor.

Caso o usuário queira solicitar uma predição, ele deve clicar no botão azul apresentado na Figura 14, com isso ele será redirecionado para a tela

da exibida na Figura 15, que se trata de um formulário no qual o usuário deve cadastrar o cromossomo, a posição, do nucleotídeo referência e o alternativo da variante que ele deseja obter a predição.

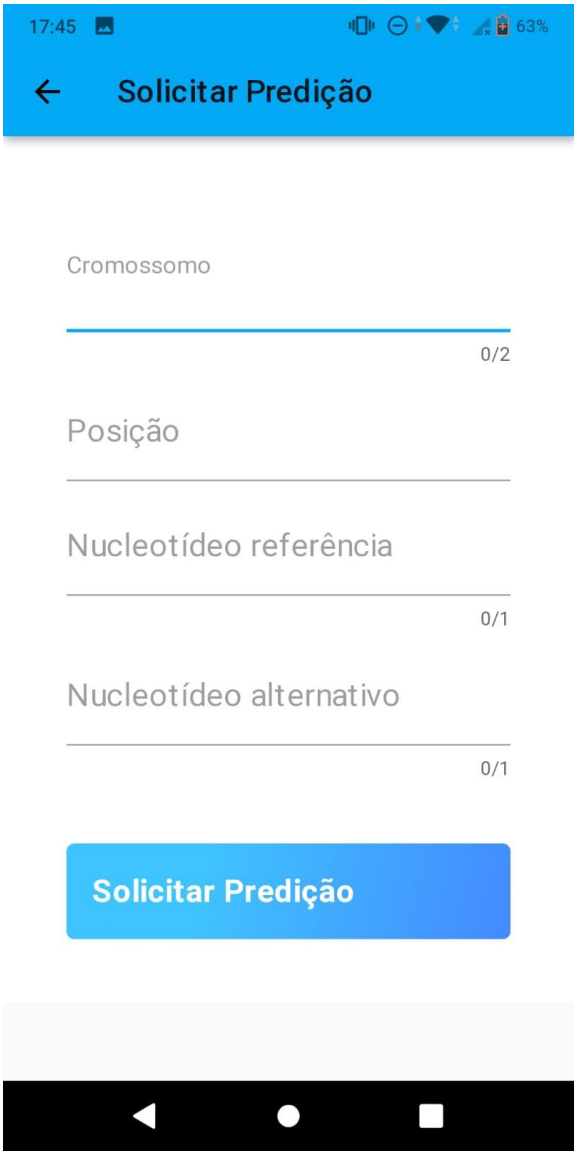


Figura 15 – Tela de cadastro de variantes.

Fonte – Elaborado pelo autor.

Por fim, o usuário também pode obter mais detalhes sobre os resultados de suas predições. No caso, caso o usuário pressione a letra correspondente a predição de determinada variante, ele é redirecionado a uma tela listando o resultado das predições dos preditores contidos no dbNSFP, como expõe a Figura 16. Além de que o usuário também pode visualizar os resultados das predições dos preditores de AM caso ele

pressione o número correspondente a quantidade de preditores de AM que informaram que a mutação é patogênica (representado pela variável NDAMAGE IA). Nesse caso, o usuário é redirecionado para a tela expressa pela Figura 17.

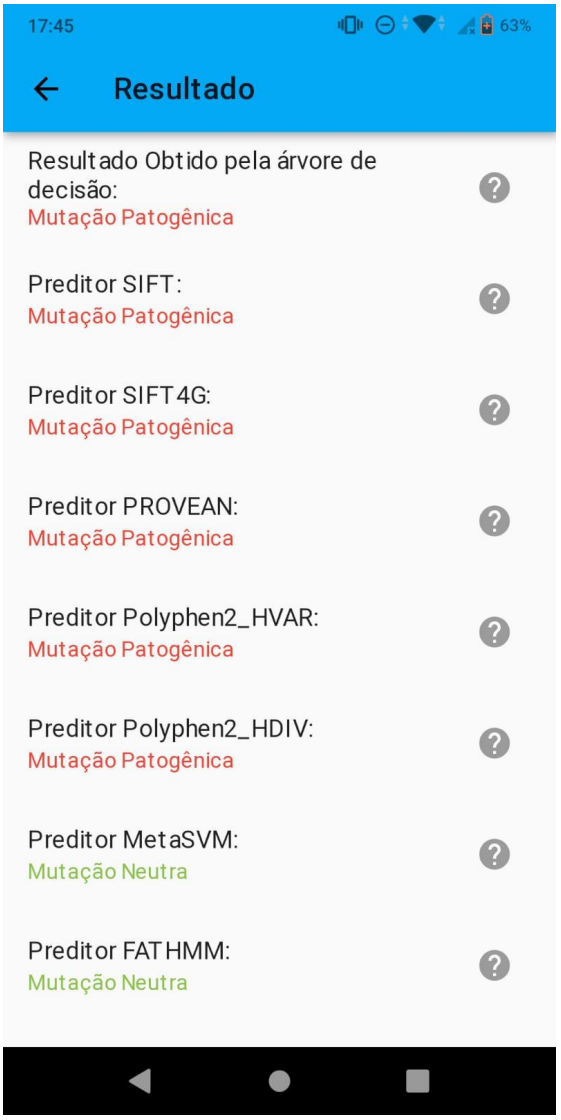


Figura 16 – Tela de listagem dos resultados preditores contidos no dbNSFP.

Fonte – Elaborado pelo autor.

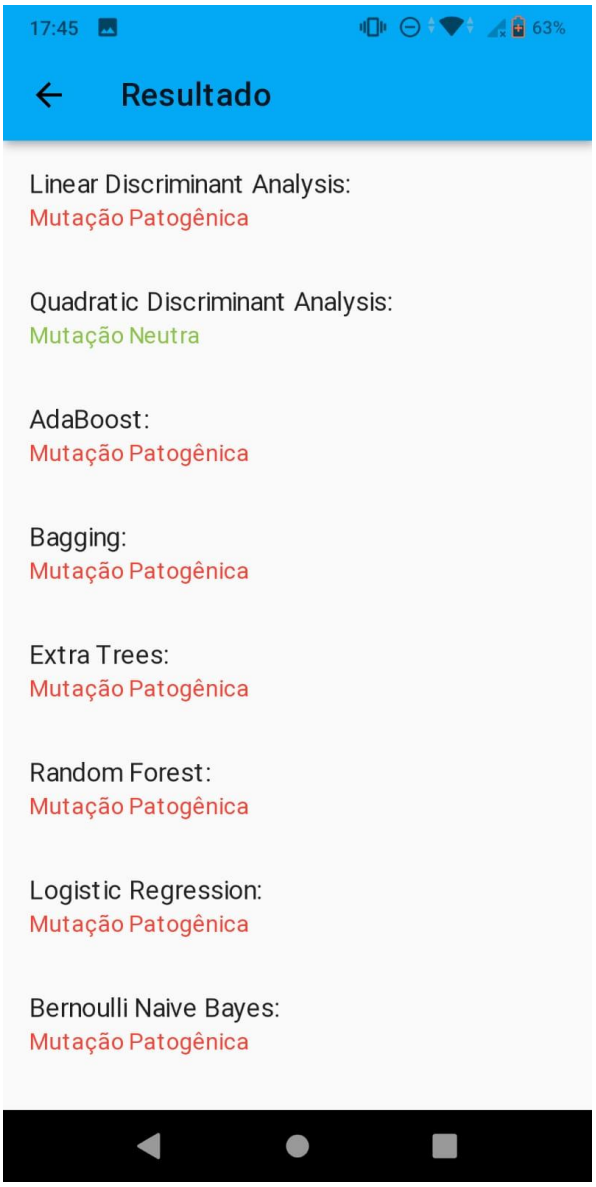


Figura 17 – Tela de listagem dos resultados dos algoritmos de aprendizado de máquina.
Fonte – Elaborado pelo autor.

4.5 Teste de usabilidade do aplicativo

Para o teste de usabilidade do sistema, foram escolhidas 10 variantes com seu significado conhecido e descrito no ClinVar, um banco de dados que contém a “interpretações da relação dessa variante com a saúde humana e as evidências que sustentam cada interpretação”(LANDRUM et al., 2014). As variantes escolhidas estão contidas na Tabela 6.

Dessa forma, uma demonstração de usabilidade do aplicativo está disponível no seguinte link: https://youtu.be/RgVJXNh_32o. No vídeo, é

possível observar que de maneira rápida e simples, o usuário pode obter a computação de suas solicitações de predição de maneira rápida e sucinta. Além disso, é possível observar que todos os resultados obtidos condizem com o que foi descrito no ClinVar.

Tabela 6 – Tabela de variantes utilizadas no teste de usabilidade.

Cromossomo	Posição	Alelo referência	Alelo alternativo	Interpretação no ClinVar
1	45331529	C	T	Patogênica
1	1050446	G	A	Benigna
1	1014228	G	A	Benigna
2	11218994	C	A	Benigna
11	2572870	G	C	Patogênica
11	2572870	G	A	Patogênica
12	102844356	T	A	Patogênica
12	15631648	A	G	Benigna
14	75006701	A	G	Patogênica
16	16163159	C	T	Patogênica

Fonte – Elaborado pelo autor.

5 Considerações finais

O presente trabalho apresenta conceitos de ciência da computação, aprendizado de máquina e bioinformática. O objetivo do projeto foi desenvolver um aplicativo que auxilie o médico na identificação de mutações patogênicas na prática clínica.

Uma vez que existem diversos preditores no mercado, a elaboração de um aplicativo centralizado que apresenta os resultados desses diversos preditores, bem como o resultado da árvore de decisão proposta por Nascimento, et al. (2020) e os algoritmos de aprendizado de máquina, auxiliará o médico a ter um melhor entendimento da precedência dessa mutação de maneira rápida e fácil.

Dessa forma, o aplicativo desenvolvido se encontra disponível em formato web https://danhfg.github.io/. Já o apk para instalação em celulares Android se encontra disponível em https://github.com/Danhfg/Danhfg.github.io/blob/main/app/android.apk.

Além disso, a API Rest que o aplicativo consome se encontra disponível em https://bioinfo.imd.ufrn.br/daniel_backend/api/.

Referências

ALESSANDRIA, Simone. Kayfitz; KAYFITZ, Brian. **Flutter Cookbook**: Over 100 Proven Techniques and Solutions for App Development with Flutter 2.2 and Dart. Packt Publishing, 2021.

BOOCH, Grady; CRUPI, John; FOWLER, Martin; MALKS, Dan; ALUR, Deepak. **Core J2EE Patterns**: best practices and design strategies. 2. ed. Proquest Information And Learning Company, 2003. 419 p.

BRACHA, Gilad. **The Dart Programming Language**. Addison-Wesley, 2015.

EVANS, Eric. **Domain-Driven Design**: tackling complexity in the heart of software. Addison Wesley, 2003. 446 p.

CHOI, Yongwook; CHAN, Agnes P.. PROVEAN web server: a tool to predict the functional effect of amino acid substitutions and indels. **Bioinformatics**, Oxford, v. 31, n. 11, p. 2745-2747, 18 ago. 2015. Disponível em: https://doi.org/10.1093/bioinformatics/btv195. Acesso em: 26 jan. 2022.

CONSORTIUM, Genome Reference (org.). **GRCh37**: genome reference consortium human build 37 (grch37). Genome Reference Consortium Human Build 37 (GRCh37). 2009. Disponível em: https://www.ncbi.nlm.nih.gov/assembly/GCF_000001405.13/. Acesso em: 24 jan. 2022.

CONSORTIUM, Genome Reference (org.). **GRCh38**: genome reference consortium human build 38. Genome Reference Consortium Human Build 38. 2013. Disponível em: https://www.ncbi.nlm.nih.gov/assembly/GCF_000001405.26/#/st. Acesso em: 24 jan. 2022.

GÉRON, Aurélien. **Hands-On Machine Learning with Scikit-Learn and TensorFlow**: Concepts, Tools, and Techniques to Build Intelligent Systems. O'reilly Media, 2017.

IVERSEN, Jakob; EIERMAN, Michael. **Gearning Mobile App Development**:

A Hands-on Guide to Building Apps with iOS and Android. Addison-Wesley Professional, 2013.

Landrum, Melissa J. ClinVar: public archive of relationships among sequence variation and human phenotype. **Nucleic Acids Research**, [s. l], v. 42, n. 1, 14 nov. 2013. Disponível em: https://doi.org/10.1093/nar/gkt1113. Acesso em: 29 dez. 2021.

LIANG, Y. Daniel. Introduction to Java Programming. 9. ed., Prentice Hall, 2012.

LI, Heng. Tabix: fast retrieval of sequence features from generic TAB-delimited files. **Bioinformatics**, v. 27, n. 1, 1 mar. 2011. Disponível em: https://doi.org/10.1093/bioinformatics/btq671. Acesso em: 29 dez. 2021.

LIU, Xiaoming; LI, Chang; MOU, Chengcheng; DONG, Yibo; TU, Yicheng. DbNSFP v4: a comprehensive database of transcript-specific functional predictions and annotations for human nonsynonymous and splice-site SNVs. **Genome Medicine**, v. 12, n. 103, p. 1-1, 02 dez. 2020. Disponível em: https://genomemedicine.biomedcentral.com/articles/10.1186/s13073-020-00803-9. Acesso em: 29 dez. 2021.

MASSE, M. **REST API Design Rulebook**: Designing Consistent RESTful Web Service Interfaces. [S.l.]: O'Reilly Media, 2011. ISBN 9781449319908.

MCLAREN, William; GIL, Laurent; HUNT, Sarah e; RIAT, Harpreet Singh; RITCHIE, Graham R s; THORMANN, Anja; FLICEK, Paul; CUNNINGHAM, Fiona. The Ensembl Variant Effect Predictor. **Genome Biology**, v. 17, n. 1, p. 122-122, 6 jun. 2016.

NASCIMENTO, Priscilla Machado do; MEDEIROS, Inácio Gomes; FALCÃO, Raul Maia; STRANSKY, Beatriz; SOUZA, Jorge Estefano Santana de. A decision tree to improve identification of pathogenic mutations in clinical practice. **Bmc Medical Informatics And Decision Making**, [s. l], v. 52, n. 1, 10 mar. 2020. Disponível em: https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s129

11-020-1060-0. Acesso em: 29 dez. 2021.

REDDY, K. Siva Prasad. **Beginning Spring Boot 2**: Applications and Microservices with the Spring Framework. Apress, 2017.

RELAN, Kunal. **Building REST APIs with Flask**: Create Python Web Services with MySQL. Apress, 2019.

SOMMERVILL, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011. 529 p.

SUMMERFIELD, Mark. **Programming in Python 3**: A Complete Introduction to the Python Language. Addison-Wesley Professional, 2008.

WALTERS-SEN, Lauren C.; HASHIMOTO, Sayaka; THRUSH, Devon Lamb; RESHMI, Shalini; GASTIER-FOSTER., Julie M.; ASTBURY, Caroline; PYATT, Robert E.. Variability in pathogenicity prediction programs: impact on clinical diagnostics. **Molecular Genetics & Genomic Medicine**, v. 3, n. 2, p. 99-110, 3 dez. 2014. Disponível em: https://doi.org/10.1002/mgg3.116. Acesso em: 29 dez. 2021.

APÊNDICE A – Processamento da solicitação de predição solicitada pelo usuário

```
ProcessBuilder pb;
pb = new ProcessBuilder("./tabix", "dbNSFP4.1a.txt.gz",
nsSNV.getChr()+":"+nsSNV.getPos().toString()+"-"+
            nsSNV.getPos().toString(),"-p", "vcf", "| awk
'($3==",nsSNV.getRef()," && $4==",nsSNV.getAlt(),")'");
File out = new File("./",user.getIdUser().toString()+
        nsSNV.getPos().toString()+ nsSNV.getAlt()+"out.vcf");
pb.redirectOutput(out);
File outLog = new File("./",user.getIdUser().toString()+
        nsSNV.getPos().toString()+ nsSNV.getAlt()+"out.log");
pb.redirectError(outLog);
Process p = pb.start();

nsSNV.setPid(p.pid());
nsSNV.setAlive(true);
nsSNVRepository.save(nsSNV);

CompletableFuture<Process> cfp = p.onExit();
cfp.thenAccept(
    ph_ ->
        {
            nsSNV.setAlive(false);
            Scanner object = null;
            try {
                object = new Scanner(new
File("./",user.getIdUser().toString()+
                            nsSNV.getPos().toString()+
nsSNV.getAlt()+"out.vcf"));
                String result = object.nextLine();
                String resultMl = processResultML(result);
                nsSNV.setResultML(getMlResults(resultMl));
                result = processResult(result);
                nsSNV.setResult(result);
```

```
                nsSNVRepository.save(nsSNV);
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            }finally{
                out.delete();
                outLog.delete();
                nsSNVRepository.save(nsSNV);
            }
            nsSNVRepository.save(nsSNV);
        });
```

APÊNDICE B – API desenvolvida em flask para retornar o resultado dos modelos de aprendizado de máquina

```
from flask import Flask, request
import json
import pandas as pd
import joblib

classe_modelos = ["LinearDiscriminantAnalysis",
                  "QuadraticDiscriminantAnalysis",
                  "AdaBoostClassifier", "BaggingClassifier",
                  "ExtraTreesClassifier", "RandomForestClassifier",
                  "LogisticRegression", "BernoulliNB",
                  "GaussianNB", "KNeighborsClassifier",
                  "MLPClassifier", "LinearSVC", "NuSVC",
                  "SVC", "DecisionTreeClassifier",
                  "ExtraTreeClassifier"]

modelos = {}
for algoritmo in classe_modelos:
    modelos[algoritmo] =
              (joblib.load('modelos/modelo_ia'+algoritmo+'.pyobj'))
app = Flask(__name__)
app.run(debug=False)

@app.route('/results',methods=['POST'])
def getResults():
    entrada = request.data.decode("utf-8").split(',')
    entradaDf = pd.DataFrame([pd.Series(entrada)])
    entradaDf.columns = ['SIFT', 'Polyphen', 'PROVEAN', 'ExAC',
                         'Ndamage', 'COMMON']
    resultStr = ""
    for algoritmo in classe_modelos:
        resultStr += algoritmo + ':' +
                     str(modelos[algoritmo].predict(entradaDf)[0])
                     + "\n"
```

```
    return resultStr, 200
```

APÊNDICE C – Treinamento dos modelos de AM

```
import pandas as pd
import joblib

from sklearn.discriminant_analysis import
LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import
QuadraticDiscriminantAnalysis
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.svm import LinearSVC
from sklearn.svm import NuSVC
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import ExtraTreeClassifier

data1 = pd.read_csv(
    "drive/MyDrive/TCC/clinvar_version_2017_05_30.txt",
    sep='\t')
data2 = pd.read_csv(
    "drive/MyDrive/TCC/clinvar_version_2019_09_23.txt",
    sep='\t')
dado_unificado = pd.concat([data1, data2])

classe_modelos = [LinearDiscriminantAnalysis,
                  QuadraticDiscriminantAnalysis,
                  AdaBoostClassifier, BaggingClassifier,
                  ExtraTreesClassifier, RandomForestClassifier,
                  LogisticRegression, BernoulliNB,
                  GaussianNB, KNeighborsClassifier,
```

```
                  MLPClassifier, LinearSVC, NuSVC,
                  SVC, DecisionTreeClassifier,
                  ExtraTreeClassifier]

for algoritmo in classe_modelos:
  modelo = algoritmo().fit(X=dado_unificado[['SIFT', 'Polyphen',
'PROVEAN', 'ExAC', 'Ndamage', 'COMMON']],
           y=dado_unificado['ClinVar'])
  joblib.dump(modelo,
           'modelos/modelo_ia'+algoritmo.__name__+'.pyobj')
```

ANEXO A – Acurácia dos preditores

Tabela apresentada por Nascimento, et al. (2020) comparando a árvore proposta, com os algoritmos de aprendizado de máquina e os preditores SIFT, Polyphen, PROVEAN e METASVM.

Classifier	**Accuracy *Mean (± Std. Dev.)**	**Predictor = N, Clinvar = 0 *Mean (± Std. Dev.)**	**Predictor = P, Clinvar = 0 *Mean (± Std. Dev.)**	**Predictor = N, Clinvar = 1 *Mean (± Std. Dev.)**	**Predictor = P, Clinvar = 1 *Mean (± Std. Dev.)**
Extreme Gradient Boosting	93 (0.3)	92 (0.5)	8 (0.5)	7 (0.3)	93 (0.3)
* Proposed Tree	92 (0.3)	91 (0.5)	9 (0.5)	8 (0.3)	92 (0.3)
Random Forest	92 (0.3)	91 (0.5)	9 (0.5)	8 (0.3)	92 (0.3)
Bagging	92 (0.3)	90 (0.5)	10 (0.5)	8 (0.3)	92 (0.3)
K Nearest Neighbors	92 (0.3)	89 (0.5)	11 (0.5)	6 (0.3)	94 (0.3)
Ada Boost	92 (0.3)	93 (0.5)	7 (0.5)	8 (0.3)	92 (0.3)
Extra Trees	91 (0.3)	90 (0.5)	10 (0.5)	8 (0.3)	92 (0.3)
Extra Tree	91 (0.3)	90 (0.5)	10 (0.5)	8 (0.3)	92 (0.3)
Linear Discriminant Analysis	91 (0.3)	88 (0.6)	12 (0.6)	8 (0.3)	92 (0.3)
Support Vector Machines (Linear kernel)	91 (0.3)	86 (0.6)	14 (0.6)	6 (0.3)	94 (0.3)
SKLearn Decision Tree	91 (0.3)	90 (0.5)	10 (0.5)	8 (0.3)	92 (0.3)
Multilayer Perceptron	91 (0.3)	85 (0.6)	15 (0.6)	6 (0.3)	94 (0.3)
Quadratic Discriminant Analysis	91 (0.3)	88 (0.5)	12 (0.5)	8 (0.3)	92 (0.3)
Bernoulli Naive Bayes	91 (0.3)	86 (0.6)	14 (0.6)	7 (0.3)	93 (0.3)
Support Vector Machines (RBF Kernel)	91 (0.3)	86 (0.6)	14 (0.6)	7 (0.3)	93 (0.3)
Logistic Regression	91 (0.3)	86 (0.6)	14 (0.6)	7 (0.3)	93 (0.3)

Gaussian Naive Bayes	90 (0.3)	84 (0.6)	16 (0.6)	6 (0.3)	94 (0.3)
Nu-Support Vector Machines	87 (0.4)	82 (0.6)	18 (0.6)	11 (0.3)	89 (0.3)
PROVEAN	83 (0.4)	75 (0.7)	25 (0.7)	13 (0.4)	87 (0.4)
MetaSVM	81 (0.4)	69 (0.6)	31 (0.6)	10 (0.4)	90 (0.4)
Polyphen	80 (0.4)	82 (0.8)	18 (0.8)	20 (0.3)	80 (0.3)
SIFT	80 (0.4)	77 (0.8)	23 (0.8)	18 (0.4)	82 (0.4)

Fonte: Retirado do artigo de Nascimento, et al. (2020).