Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Biológicas
Departamento de Estatística

Monografia

UTILIZAÇÃO DE REDES NEURAIS ARTIFICIAIS PARA CLASSIFICAÇÃO DE APLICATIVOS MALICIOSOS

Ismael de Almeida Dias

Ouro Preto – MG
2022

Ismael de Almeida Dias

UTILIZAÇÃO DE REDES NEURAIS ARTIFICIAIS PARA CLASSIFICAÇÃO DE APLICATIVOS MALICIOSOS

Monografia apresentada ao Curso de Estatística da Universidade Federal de Ouro Preto como parte dos requisitos necessários para a obtenção do grau em Bacharelado em Estatística.

Orientador: Prof. Dr. Tiago Pereira

Ouro Preto – MG

2022

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E BIOLOGICAS
COLEGIADO DO CURSO DE ESTATISTICA

FOLHA DE APROVAÇÃO

Ismael de Almeida Dias

Utilização de Redes Neurais Artificiais para Classificação de Aplicativos Maliciosos

Monografia apresentada ao Curso de Estatística da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Estatística

Aprovada em 24 de junho de 2022

Membros da banca

Dr. Tiago Martins Pereira - Orientador - Universidade Federal de Ouro Preto
Drª. Diana Campos de Oliveira - Universidade Federal de Ouro Preto
Dr. Marcelo Carlos Ribeiro - Universidade Federal de Ouro Preto

Professor Dr. Tiago Martins Pereira, orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 24/06/2022

Documento assinado eletronicamente por **Tiago Martins Pereira**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 26/10/2022, às 13:47, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0348658** e o código CRC **69E06949**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: (31)3559-1312 - www.ufop.br

AGRADECIMENTOS

Agradeço a Deus pela vida e por ter me permitido aprender tanto nesse período de graduação, Ele me deu força para não desistir e permanecer, mesmo nos momentos mais complicados e difíceis.

Agradeço aos meus pais, William Cândido Dias e Elaine Conceição de Almeida Dias, e ao meu irmão, Arthur de Almeida Dias, por todo o apoio emocional e financeiro, pela força, pelo ânimo e por terem acreditado em mim desde o começo.

Agradeço à minha noiva, Leyla Dowsley, por sua alegria e força constante que tornam a vida mais fácil. Sou grato por sempre dizer que eu seria capaz, eu não teria conseguido sem a sua ajuda.

Agradeço à congregação do Retiro da Assembleia de Deus da regional de Nova Contagem pela ajuda em oração durante todo esse tempo.

Agradeço ao meu orientador, Professor Doutor Tiago Martins Pereira, por ter me auxiliado em todos os detalhes e etapas para a construção desse trabalho.

Agradeço a todos os meus colegas e amigos do curso de estatística, que ultrapassaram comigo as barreiras que muitas vezes pareciam impossíveis, sempre com muito esforço e estudo.

Agradeço à empresa Oper – e especialmente ao meu querido amigo Guilherme Gil - por ter me ensinado tanto através do estágio e das experiências adicionais que vem me proporcionando, possibilitando que eu me torne a cada dia um estatístico melhor.

Agradeço a Israel Alves por ter me ajudado no início de tudo, me auxiliando na inscrição e me dando valiosas dicas e informações sobre a UFOP.

Por fim, agradeço a todo o corpo docente do departamento de estatística da UFOP pelas excelentes aulas, pela ótima estrutura e pela motivação a permanecer no curso.

“Porque Deus amou o mundo de tal maneira que deu o seu Filho unigênito, para que todo aquele que nEle crê não pereça, mas tenha a vida eterna. Porque Deus enviou o seu Filho ao mundo, não para que condenasse o mundo, mas para que o mundo fosse salvo por Ele.”

João 3:16-17

RESUMO

Utilização de redes neurais artificiais para classificação de aplicativos maliciosos

Autor: Ismael De Almeida Dias

Orientador: Prof. Dr. Tiago Pereira

O algoritmo de redes neurais artificiais é amplamente utilizado quando se trata de classificação, resultando em um nível de acurácia elevado em diversas aplicações. Para esse estudo, o objetivo é a aplicação do multilayer perceptron com adição de diversas técnicas de pré-processamento para a classificação de aplicativos maliciosos através do estudo das permissões a eles solicitadas no momento da instalação e execução. O modelo final obtido utilizou 64 variáveis como input aos neurônios de entrada e 1 camada com 62 neurônios, obtendo uma acurácia de 86%, além de um índice Kappa de 0,75. Os processos envolvendo os resultados alcançados foram feitos utilizando o software R.

Palavras-chave: Redes Neurais Artificiais, Classificação, Multilayer Perceptron

ABSTRACT

Using artificial neural networks to classify malicious applications

Autor: Ismael De Almeida Dias

Advisor: Prof. Dr. Tiago Pereira

The artificial neural network algorithm is widely used when it comes to classification, resulting in a high level of accuracy in the most diverse applications. For this study, the objective is the application of the multilayer perceptron with the addition of several pre-processing techniques for the classification of malicious applications through the study of the permissions requested at the time of installation and execution. The final model obtained used 64 variables as input to the neurons and 1 layer with 62 neurons, obtaining an accuracy of 86%, in addition to a Kappa of 0.75. The processes involving the results achieved were done using the software R.

Key words: Artificial Neural Networks, Classification, Multilayer Perceptron

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

O advento da internet possibilitou à humanidade novas formas de evolução tecnológica, viabilizando o desenvolvimento de softwares capazes de facilitar as interações humanas e automatizar inúmeros tipos de trabalho. Desde então, cada vez mais dados são gerados e mais informações armazenadas. Nesse cenário, a privacidade e a segurança no dispositivo móvel se tornaram um tema pertinente, de modo que os aplicativos se consolidaram como o maior meio de uso das funcionalidades dos celulares.

As permissões no sistema operacional Android dão aos aplicativos a autorização para acessar os recursos do dispositivo. Esses acessos podem, eventualmente, colocar em risco informações confidenciais dos usuários. Com o passar dos anos, diversas ações foram implementadas pelo Google para evitar que os aplicativos solicitassem permissões desnecessárias para o uso normal de suas funcionalidades. Existe uma clara recomendação nas documentações do Android para que os desenvolvedores solicitem as permissões o mais tarde possível, exclusivamente quando o usuário utilizar alguma função que torne necessária a solicitação. Nem sempre as consequências do acesso a dados do aparelho são aparentes, sendo, em sua maioria, uma forma de extrair informações valiosas sobre comportamentos de consumo e interesses do usuário, a fim de lhe mostrar anúncios mais relevantes e induzi-lo a variados tipos de ideologia, como pode ter acontecido no famoso caso da Cambridge Analytica (SHAKIL, 2018), acusada de expor dados de milhões de usuários. Porém, neste trabalho, o objetivo não é analisar as interações que acontecem através da mineração de dados do usuário de forma legal nos padrões da LGPD (Lei Geral de Proteção de Dados), e sim estudar como as permissões estão ligadas à ocorrência de aplicações maliciosas que oferecem um risco real e ilegal aos usuários. Esse problema pode se intensificar ainda mais com o crescente número de lançamentos na Play Store (loja de aplicativos oficial do Android), frente à demanda social elevada de conteúdo e entretenimento, fazendo com que o Google se esforce para tornar seu sistema operacional cada vez mais seguro.

É interessante, inicialmente, entender como essas permissões se relacionam com o sistema operacional. Quando uma aplicação solicita

permissões que não são utilizadas no seu uso normal, isso é chamado “*over privilege*” (FELT, 2011). Essas permissões abrem uma brecha na segurança, tornando possível o acesso a dados confidenciais dos usuários por parte de terceiros, podendo incluir fotos e arquivos pessoais, informações de localização, acesso ao microfone e à câmera, dados bancários, entre outros dados sensíveis. Por outro lado, permissões excessivamente conservadoras tornam a aplicação menos eficiente. Políticas corretamente configuradas devem garantir: que os aplicativos sejam protegidos uns dos outros; que a plataforma seja protegida dos aplicativos; a funcionalidade dos aplicativos e que os usuários não sejam indevidamente incomodados (STEVENS, 2013). Cumprir todos os itens nem sempre é uma tarefa fácil para o sistema, visto que há milhares de aplicativos disponíveis para download nas lojas (como a Play Store), podendo também ser atualizados, alterando suas configurações e funcionalidades, além da possibilidade (não recomendada) da instalação por fontes externas. Sendo assim, como identificar possíveis aplicativos maliciosos de forma eficiente? Possivelmente, esse é um problema que não apenas o Android, mas todos os sistemas operacionais enfrentam. A solução pode envolver diversos caminhos de acordo com cada loja de aplicativos ou programas.

Diversas ferramentas computacionais se mostram interessantes quando o assunto é classificação, principalmente quando se unem aos poderes de processamento atuais e às técnicas estatísticas. *Machine Learning*, ou Aprendizado de Máquina, é um subcampo da Inteligência Artificial onde, através de diversas iterações, os algoritmos podem autoajustar seus parâmetros a fim de produzirem melhores resultados. A estatística entra nesse contexto para o pré-processamento e na produção de métricas para os modelos encontrados, a fim de medir quais estão se saindo melhor. Frente a isso, a análise das relações entre a solicitação destas permissões por parte dos desenvolvedores e o possível risco oferecido tem grande valor prático. Este trabalho se propõe a estudar essas relações através de métodos de classificação por meio de *machine learning*. Essa abordagem no banco de dados deve possibilitar a obtenção de um modelo com um nível aceitável de acurácia para a classificação, dado as características da aplicação em adição às permissões solicitadas. É também de interesse, além de classificar, estudar em quais categorias e grupos

de aplicativos há uma prevalência significativa de malwares. Por fim, ponderar se a aplicação do modelo obtido para classificação é adequada, sendo também de interesse comum entender e, se possível, identificar quando uma aplicação está mais propensa a oferecer algum tipo de risco aos seus usuários. O presente trabalho apresenta brevemente as soluções encontradas por diversos autores, desenvolvendo uma análise a fim de obter resultados semelhantes.

2. REFERENCIAL TEÓRICO

2.1 Trabalhos Semelhantes

Existem duas grandes abordagens utilizadas na análise de aplicações maliciosas no Android, sendo: estática e dinâmica. Em resumo, a análise estática estuda a aplicação antes da sua instalação, sendo, portanto, a mais rápida. A análise dinâmica vai mais além, pois é capaz de estudar não apenas o código e seus arquivos, mas também as interações com o sistema (MAHINDRU et al., 2017). A base de dados utilizada neste trabalho foi obtida através de uma análise dinâmica.

Diversos autores propuseram abordagens tanto estáticas, quanto dinâmicas. Grace *et al.* (2012) propõe RiskRanker, um sistema automatizado escalável capaz de analisar se um determinado aplicativo apresenta comportamento perigoso, a análise de primeira ordem utiliza abordagem estática e a análise de segunda ordem apresenta capturas de comportamento de módulos a fim de mitigar as fraquezas da análise estática. Ongtang et al. (2009) propõe uma abordagem para mitigar o mau uso geral de aplicações Android, provendo um método prático no momento da instalação a fim de detectar formas e comportamentos maliciosos. Em um estudo de 2016, (ALATWI et al., 2016), foi proposto o uso de *machine learning* para a classificação de aplicativos de acordo com suas categorias, o mesmo autor fornece uma interessante tabela contendo as famílias de malwares no Android com seus tipos e características:

Tabela 1 – Malwares: tipos e definição. (ALATWI et al., 2016)

Tipo	Definição	Exemplo
Trojan	Se disfarça como um aplicativo inicial para ocultar sua identidade de malícia. Oferece funcionalidades úteis ao usuário, mas realiza atividades maliciosas em segundo plano sem o conhecimento do usuário.	FakeNetFlix, Zsone, Zitmo, Spitmo, Fakeplayer, Android.Foney
Backdoor	Permite o acesso remoto ao sistema e ignora o mecanismo de autenticação. Geralmente explora vulnerabilidades no sistema para obter privilégios de root; ele tem capacidade de permanecer indetectável.	Basebridge, Kmin, Obad
Worm	Copia e se espalha pelo nó de uma rede sem a necessidade de ser iniciado pelo usuário de um sistema.	Android.Obad.OS
Bot	Permite que um invasor controle remotamente o dispositivo a partir de um servidor chamado Bot-master. O invasor comanda o sistema e outros infectados para lançar um ataque como: DDoS.	Geinimi, Beanbot, Anserverbot
Spyware	Envia dados do usuário como: contatos, mensagens, localização e outros dados confidenciais para um servidor remoto.	Nickyspy, GPSSpy
Adware	Envia anúncios personalizados com base nos dados coletados de um usuário, como localização.	Plankton
Ransomware	Bloqueia o sistema para torná-lo inacessível até que algum resgate seja pago pelo usuário.	FakeDefender

Um framework que emprega técnicas de classificação utilizando *Machine Learning* (SHABTAI, 2012) foi proposto para a detecção de malwares, onde avalia os classificadores k-Means, Regressão Logística, Árvores de decisão, Redes Bayesianas e Naive Bayes. Cheng et al. (2007) propuseram SmartSiren, um sistema colaborativo de detecção e alerta de vírus para smartphone que, através da coleta de informações de comunicação, encontra comportamentos anormais. Através das fontes citadas é possível observar que existem diversas formas desenvolvidas para lidar com este problema e suas variações.

O uso de redes neurais artificiais para fins de classificação tem se mostrado uma excelente opção quando se tem uma grande quantidade de observações e variáveis, tornando-se ainda mais popular com os algoritmos de reconhecimento de imagens. Oliveira (2019) utiliza *Support Vector Machine*, K-*Nearest Neighbors* e *Random Forest*) e a *Convolutional Neural Network* (CNN)

para classificação de áreas queimadas através de imagens de alta resolução, onde a CNN obteve os melhores resultados. Além de reconhecimento de imagens, as redes neurais artificiais também apresentam bons resultados quando o assunto é encontrar padrões. Cunha et al. (2007) propõe um método de extração de padrões do sinal mioelétrico em uma prótese de mão de maneira simplificada, utilizando redes neurais artificiais. Os resultados obtidos, depois de 1000 ciclos, mostraram uma porcentagem de acerto de 100% em uma das arquiteturas.

2.2 Sistema operacional Android

Alatwi *et al.* (2016) descrevem de forma clara e resumida como as permissões funcionam dentro do sistema operacional estudado, “O Android executa aplicativos separadamente em *sandboxes*. Cada aplicativo é executado em um ambiente isolado, onde não tem acesso aos recursos do sistema. As permissões devem ser fornecidas ao aplicativo para ser capaz de acessar e usar os recursos do sistema que são necessários para sua funcionalidade”.

A partir do Android 11, algumas atualizações importantes aconteceram no sistema operacional com o objetivo de torná-lo mais seguro. As atualizações incluem as opções de: o usuário conceder uma permissão única e temporária para permissões relacionadas à localização, ao microfone ou à câmera; redefinição automática após alguns meses de permissões confidenciais concedidas e não exibição da caixa de diálogo quando o usuário negar mais de uma vez o acesso à permissão - esta ação implica “não perguntar novamente”. Além disso, em dispositivos antigos haverá a redefinição automática de permissões. Observa-se grande esforço por parte do Google em administrar de forma a encontrar um equilíbrio ideal para a privacidade e segurança do usuário, ao mesmo tempo que tenta não limitar, além do ideal, as ações dos aplicativos, o que prejudicaria o desenvolvimento e a funcionalidade dos mesmos.

> Permissões de execução, também conhecidas como permissões perigosas, dão ao seu app acesso adicional a dados restritos e permitem que ele execute ações restritas que afetam mais significativamente o sistema e outros apps. [...] Muitas permissões de execução acessam dados particulares do

> usuário, um tipo especial de dado restrito que inclui informações potencialmente confidenciais. (Android, 2021)

Fica claro, dado a leitura da documentação, que as permissões ditas de execução fornecem um acesso maior no aplicativo aos dados do usuário, estas permissões têm um papel de destaque pois podem oferecer riscos reais caso o desenvolvedor do aplicativo em questão tenha más intenções. Além disso, algumas permissões podem, inclusive, alterar configurações do dispositivo, tornando o potencial ainda pior de tais validações. É nítido, portanto, como as permissões têm um papel crucial na administração do sistema operacional Android no uso cotidiano dos consumidores, visto a crescente automatização das tarefas diárias, gerando um número cada vez maior de usuários conectados e fazendo uso dos aplicativos presentes na Play Store.

3. MÉTODOS E MATERIAIS

3.1 Pré-processamento

Os dados para o presente trabalho foram extraídos do site Kaggle.com (https://www.kaggle.com/saurabhshahane/android-permission-dataset), sendo uma base de dados estruturados que possui 173 variáveis referentes às permissões solicitadas de aplicativos, tendo 29999 observações. Também há: duas variáveis que indicam a quantidade de permissões seguras e perigosas solicitadas; uma variável binária que indica se cada aplicativo é ou não malicioso e algumas variáveis que descrevem os aspectos gerais da aplicação, sendo elas: Nome, Categoria, Descrição, Pacote, Preço, Média de Avaliações e Quantidade de Avaliações. A base foi baixada e pré-processada através do software R. As categorias, como apresenta a **Tabela *2***, mostram ser uma base de dados muito variada nas temáticas.

Tabela 2 – Categoria das aplicações.

Categoria	Frequência
Entretenimento	2322
Livros e Referência	1685
Viagem e Local	1445
Estratégia e Quebra-cabeça	1432
Personalização	1317
Arcade & Ação	1316
Casual	1253
Estilo de Vida	1181
Ferramentas	1078
Educação	1067
Finanças	861
Saúde e Fitness	826
Negócios	803
Comunicação	801
Cartas e Cassino	650
Mídia & Vídeo	611
Bibliotecas e Demonstrações	556
Quadrinhos	546
Esportes	544
Produtividade	482
Música e Áudio	397
Notícias e Revistas	390
Transportes	366
Outros	1630

Para uma análise bem-sucedida, é de extrema importância fazer um bom pré-tratamento nos dados. Inicialmente, foram removidos do banco de dados os aplicativos que estavam com o Nome, Categoria e Pacote repetidos, deixando apenas o primeiro registro dessas linhas, reduzindo, então, o número de linhas do banco de dados para 23559. É também de interesse que as variáveis que não geram informação sejam removidas, sendo assim, foram retiradas do banco de dados todas as colunas que possuíam apenas valores de 0 ou de 1 em sua totalidade, restando 162 variáveis. Em sequência, todas as observações que possuíam valores faltantes (NA) também foram removidas, sendo no total de 680 valores, correspondendo a 2,88% das observações.

Feitos os passos mais simples e iniciais, é necessário se preocupar com a questão de um possível desbalanceamento nas classes. Este desbalanceamento pode ocasionar um modelo com baixo desempenho, principalmente na classe que possui menos observações. *Undersampling* é a técnica que consiste em extrair aleatoriamente uma amostra do tamanho da quantidade de observações que se tem da classe menos frequente, das observações da classe mais frequente. Já o *oversampling* faz uma reamostragem dentro da classe menos frequente, a fim de que a quantidade se iguale à classe mais frequente, como mostra a **Figura *1***.

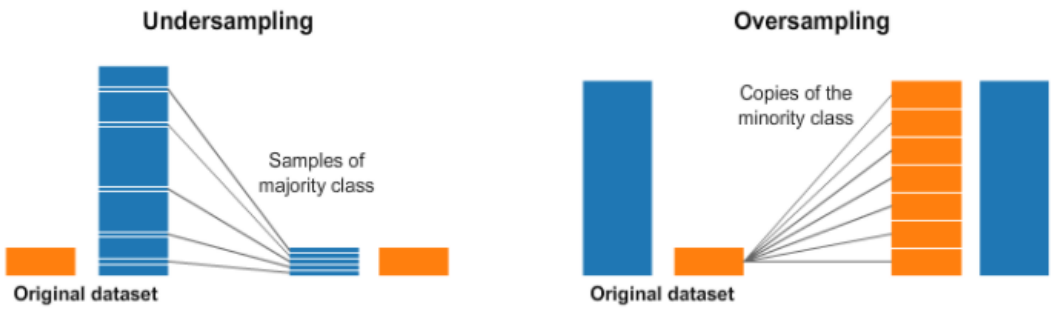

Figura 1 – *Oversampling and Undersampling*

Analisando o gráfico de frequência das classes (**Figura *2***) pode-se observar que o banco de dados está desbalanceado. Para lidar com esse problema, é interessante pontuar que o tamanho da amostra é consideravelmente grande, sendo assim, não é um problema utilizar o método de *undersampling* descrito anteriormente. Vale destacar que o aprendizado de máquina através do método das redes neurais artificiais demanda um alto poder de processamento. Dessa forma, uma amostra demasiadamente grande poderia levar a tempos muito longos de treinamento, dificultando a análise e o encontro dos melhores parâmetros.

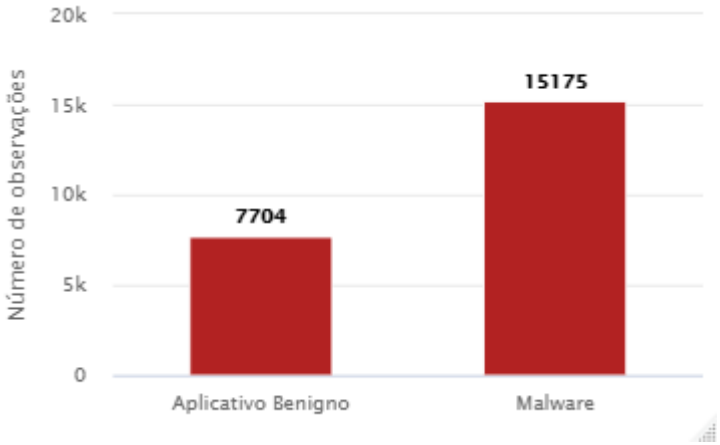

Figura 2 – Frequência das classes antes da aplicação das técnicas de *Oversampling* ou *Undersampling*

Utilizando algumas funções do pacote Themis (HVITFELDT, 2021) no software R, foi possível aplicar o método *undersampling*, tendo como resultado o mesmo valor para as duas classes: 15408 observações totais. Plotando o mesmo gráfico na **Figura 3**, observa-se que o problema do desbalanceamento das classes foi corrigido.

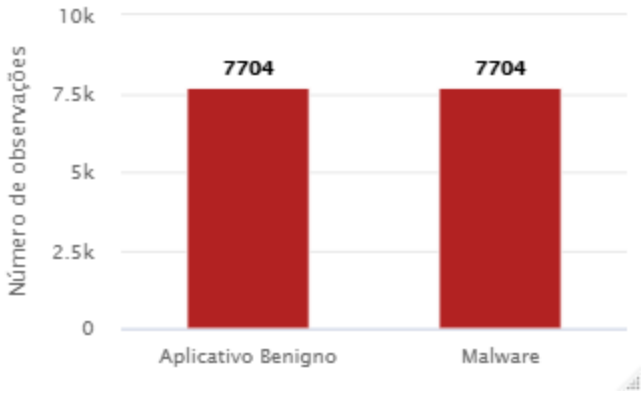

Figura 3 – Frequência das classes depois da aplicação da técnica de *Undersampling*

Porém, ao realizar diversos testes, foi possível observar que a melhor técnica para corrigir o desbalanceamento das classes é o *oversampling*, mesmo resultando em um tempo muito maior para o treinamento da rede neural. A principal diferença nos testes foi a de que a acurácia é bem maior para o modelo de oversampling em comparação ao undersampling. O resultado foi um total de

30350 observações na base de dados final, sendo 15175 observações para cada classe, como mostra a **Figura 4**.

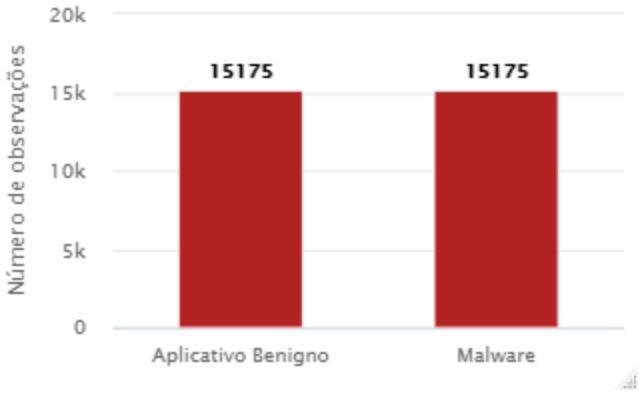

Figura 4 - Frequência das classes depois da aplicação da técnica de *Oversampling*

Com as classes devidamente balanceadas, é importante remover do banco de dados as variáveis relacionadas às permissões que possuem poucas informações, pois elas podem fazer com que o treinamento do modelo seja muito mais demorado e ineficiente. Para definir qual variável será removida, o método utilizado foi identificar as permissões que foram menos solicitadas pelos aplicativos do banco de dados em geral e remover o percentil 33%. Desse modo, as variáveis que possuíam pouquíssima informação, ou seja, as permissões quase nunca solicitadas, foram removidas, restando, do total inicial, apenas 110 variáveis.

3.2 Redes Neurais Artificiais

As redes neurais artificiais podem ser definidas como “técnicas computacionais que apresentam um modelo matemático inspirado na estrutura neural de organismos inteligentes e que adquirem conhecimento através da experiência” (CARVALHO et al., 1998), sendo amplamente utilizada nas áreas de aprendizado de máquina e inteligência artificial. A apresentação do primeiro modelo de neurônio artificial (MCCULLOCH et al., 1943) certamente foi um momento importante, onde, em breve, o tipo mais simples de rede neural seria inventada em 1958 por Frank Rosenblatt, o Perceptron (DATA SCIENCE

ACADEMY, 2022). É possível ver, através da **Figura *5***, a semelhança entre um neurônio e o Perceptron.

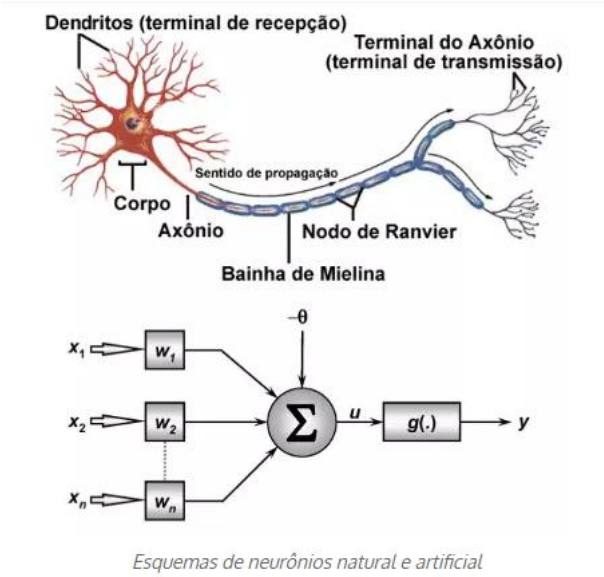

Figura 5 – Esquemas de neurônios natural e artificial. Disponível em: https://www.atenaeditora.com.br/wp-content/uploads/2019/05/e-book-Redes-Neurais-Artificiais-uma-Abordagem-para-Sala-de-Aula.pdf. Acesso em 22 dez. 2021

O modelo apresentado do Perceptron pode ser descrito desta forma, matematicamente (MCCULLOCH et al., 1943):

$$u_k = \sum_{j-1}^{m} w_{kj} x_j$$
$$y_k = \varphi(u_k + \theta_k)$$

onde: x_j – Sinais de entrada.

w_{kj} – Pesos sinápticos ou pesos.

y_k – Sinais de saída.

$\varphi()$ – Função de ativação.

Θ - Bias.

A função de ativação é uma parte fundamental para o bom desempenho de uma rede neural. As funções de ativação mais comuns e amplamente utilizadas são:

- Unidade Linear Retificada (ReLU)
- Sigmoide
- Tangente Hiperbólica (TanH)

Para o trabalho em questão, será utilizada a função sigmoide (ou logística), representada da seguinte forma:

$$\sigma(x) = \frac{1}{1 + e^x}$$

É interessante destacar que a função sigmoide possui diversas vantagens e desvantagens, de acordo com a aplicação do problema. Para o caso de classificação de duas categorias, ela apresenta bons resultados. Porém, como desvantagem, os gradientes podem assumir valores muito próximos de 0 e deixarem de aprender de fato (DATA SCIENCE ACADEMY, 2022). Além disso, os valores de saída vão sempre variar de 0 a 1, sendo assim, esse output pode não ser adequado de acordo com o problema que se deseja resolver. Para o caso do banco de dados em questão, é de interesse classificar o aplicativo em duas classes distintas (malware ou benigno), portanto, a aplicação dessa função é coerente.

Um único *perceptron* é capaz de resolver apenas funções linearmente separáveis (DATA SCIENCE ACADEMY, 2022). Porém, quando se tem um problema real, raramente os dados poderão ser resolvidos com esse modelo simples. Sendo assim, é possível interligar os neurônios, formando uma rede neural composta de múltiplos *perceptrons*. Essa rede é chamada de *multilayer perceptron* (**Figura *6***), possuindo capacidade para lidar com problemas não linearmente separáveis, viabilizando diversas aplicações.

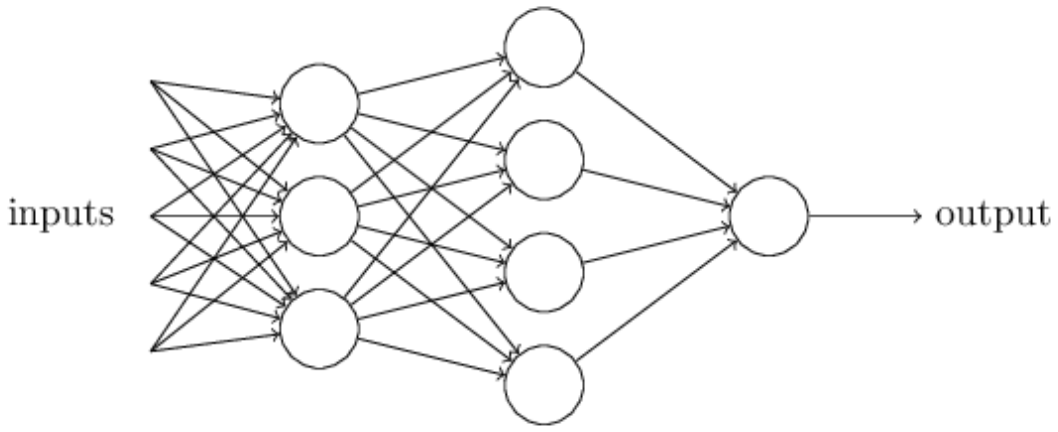

Figura 6 – *Multilayer Perceptron*. (FURTADO,2019)

No aprendizado supervisionado, há valores alvo onde se deseja que a rede neural convirja. Para que isso aconteça, é necessário encontrar os pesos adequados. Fazer essas tentativas de forma aleatória não seria possível, afinal seria necessário um número infinito de repetições. Para que os pesos corretos sejam encontrados sem um custo operacional inatingível, o algoritmo *backpropagation* possibilita, através de diversas operações envolvendo derivadas por meio do gradiente descendente, a otimização dos pesos para que a rede neural possa aprender a mapear corretamente as entradas e, assim, convergir para uma solução ótima para o problema. Esse algoritmo é a peça-chave das redes neurais artificiais, pois permitiu resolver problemas que outros algoritmos não conseguiam. Pode-se usar a regra delta para atualização dos pesos:

$$Novo\ Peso\ =\ Peso\ Antigo - Derivada\ *\ Taxa\ de\ Aprendizagem$$

A taxa de aprendizagem, se for um valor alto, fará com que seja mais difícil para os pesos convergirem, dado que, com frequência, ultrapassará o valor ideal dos pesos. Caso seja um valor muito baixo, torna lenta demais a aprendizagem. O valor mais comumente utilizado é de 0.01, de forma que também dependerá do problema em questão.

3.3 Método de validação cruzada com repetição K-fold

A validação cruzada consiste em particionar os dados em dois grupos diferentes, sendo um para o treinamento da rede neural e outro de teste para fazer a medição das métricas como acurácia, sensibilidade, entre outras. É de interesse utilizar a validação cruzada por k-fold com o objetivo de obter uma medida de precisão do modelo mais confiável e precisa. Quando se tem diversas partições do mesmo banco de dados para teste, a medição da acurácia tende a ser mais confiável. A **Figura *7*** representa de forma didática como é feito este método:

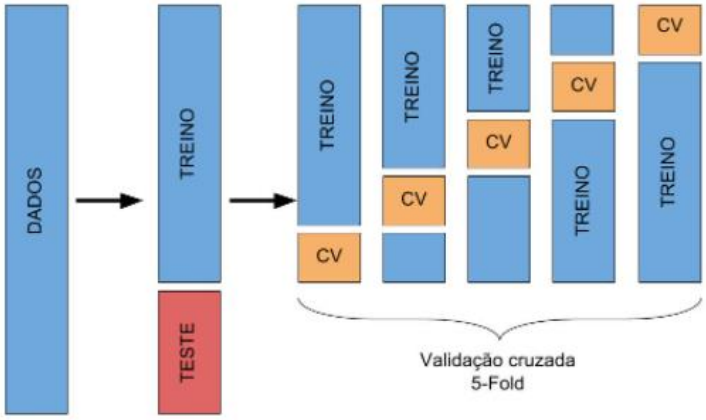

Figura 7 – Método de validação cruzada. Disponível em: http://cursos.leg.ufpr.br/ML4all/apoio/reamostragem.html. Acesso em 17 ago. 2021

3.4 Variáveis *Dummy*

É comum em um banco de dados com uma grande quantidade de variáveis haver a presença de uma ou mais variáveis qualitativas. Como não é possível realizar cálculos, nem mesmo equações com variáveis qualitativas, é interessante a procura de uma forma de quantificar essas variáveis, a fim de que elas sejam incluídas nas equações propostas, sem que haja a necessidade da remoção das mesmas, o que ocasiona certamente perda de informação. Para resolver esse impasse, surge a possibilidade da inserção de variáveis *dummy*. Em relação a trabalhos sobre esse tema, Dufour (1980) discorre propondo uma interpretação para variáveis *dummy*. A proposta é construir variáveis artificiais que irão possuir valores de 0 ou 1, de acordo com a presença ou ausência de cada uma das categorias contidas na variável qualitativa. A vantagem é que, dessa forma, é possível inserir as categorias em formato de novas variáveis no modelo, levando em conta mais informação do que haveria caso a variável qualitativa tivesse sido removida. A desvantagem no contexto de um modelo de redes neurais artificiais é que, consequentemente, se adiciona uma quantidade grande de variáveis, caso a variável qualitativa de interesse tenha muitas categorias, o que é o caso da base em questão deste estudo. Isso pode tornar o treinamento da rede neural muito demorado e, em alguns casos, inviável.

Na base de dados deste trabalho, a variável “categoria” é qualitativa, possuindo um total de 30 categorias, 23 delas são mostradas na **Tabela *2***. Com a finalidade de obter resultados com níveis aceitáveis de acurácia, esta variável foi transformada em 30 variáveis *dummy*. Dessa maneira, o banco de dados passa a possuir apenas variáveis quantitativas para o treinamento do modelo.

3.5 Seleção de variáveis com RFE

No banco de dados deste trabalho, mesmo após o pré-processamento, ainda restaram 110 variáveis, o que pode ser custoso para um modelo de redes neurais artificiais, pois pode levar muito tempo para o treinamento, dificultando muito a análise e os testes. Sendo assim, uma ótima opção para a seleção de variáveis é utilizar RFE (*Recursive Feature Elimination*). Esse algoritmo consiste em realizar o treinamento do modelo - neste caso, utilizando o algoritmo *Random Forest* - com todas as variáveis inicialmente e eliminar de forma recursiva as variáveis menos importantes até que reste as variáveis que produzem um modelo com a maior acurácia. A escolha da aplicação deste método com o modelo de *Random Forest* para o banco de dados ocorre, pois treinar um modelo de *Random Forest* é menos custoso computacionalmente do que treinar um modelo de Redes Neurais Artificiais. Como desvantagem, será necessário pressupor que as variáveis ideais para o ajuste do modelo de *Random Forest* serão as mesmas para o ajuste do modelo de Redes Neurais Artificiais, mesmo sendo algoritmos diferentes, entretanto, ambos são algoritmos úteis para problemas de classificação.

Como resultado, foram selecionadas 64 variáveis, como mostra a **Figura *8***, (nelas estão inclusas todas as variáveis *Dummies*) que maximizaram a acurácia.

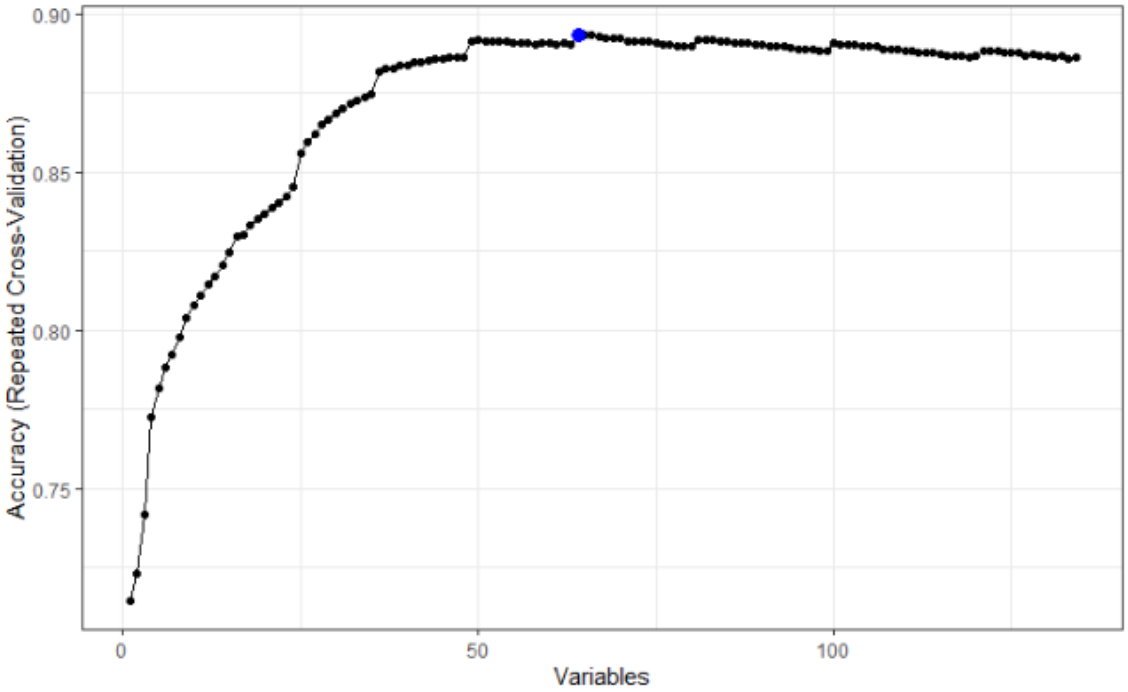

Figura 8 – Acurácia de acordo com a quantidade de variáveis selecionadas.

Todas as variáveis selecionadas se encontram descritas no **ANEXO A**. Essas serão as variáveis utilizadas para o treinamento do modelo de redes neurais artificiais.

4. RESULTADOS

4.1 Análise Descritiva

Nesta etapa é de extrema importância analisar as variáveis relevantes do banco de dados e como elas se comportam quando são da Classe dos aplicativos Malignos e dos Benignos, essa análise prévia pode ajudar a entender melhor os resultados e dar maior suporte para as interpretações dos pesos da rede neural. Para começar, a **Figura *9*** mostra o *Boxplot* da variável Número de Avaliações quando se trata de aplicativos Benignos e Malignos antes da aplicação da normalização min/max.

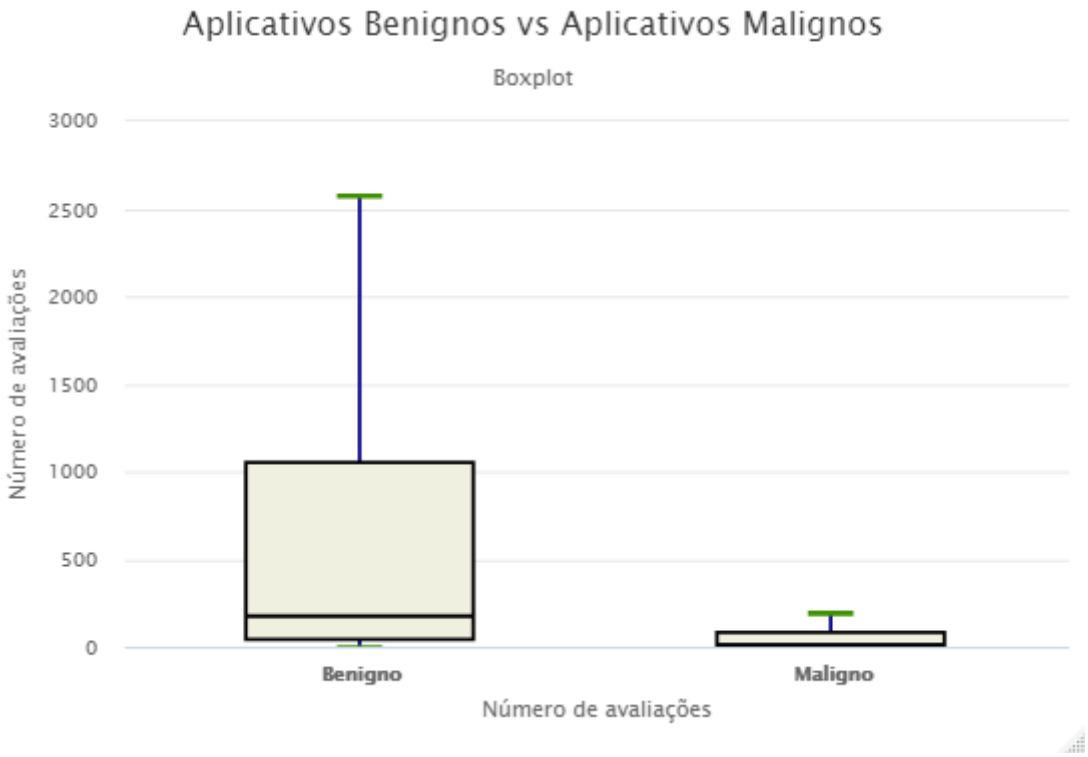

Figura 9 – *Boxplot* - Número de avaliações.

Figura 10 – Classe: Aplicativos Malignos.

Figura 11 – Classe: Aplicativos Benignos

É possível observar que os aplicativos benignos possuem um número bem maior de avaliações nos aplicativos em geral, mesmo havendo menos frequência, possivelmente pelo motivo de serem mais populares e bem mais baixados, visto que um aplicativo malicioso pode ser descoberto e desinstalado enquanto um aplicativo benigno tem uma probabilidade muito maior de ser indicado para amigos e familiares. Pode-se observar também que a variância é bem maior para este grupo, não tendo os valores tão fortemente concentrados em poucas avaliações, indicando uma interação muito maior do usuário na página de avaliação. Esse resultado é o que se espera de aplicações que não prejudiquem nem tentem acessar informações daqueles que fazem download. Já do lado dos aplicativos maliciosos vê-se que há um volume muito grande de aplicações com poucas ou nenhuma avaliação e mesmo os que possuem muitas avaliações não são tão numerosas em relação a classe oposta.

Outro aspecto do banco de dados interessante de estudar é a categoria, pois é provável que algumas categorias de aplicativos sejam mais propensas a terem aplicativos maliciosos desenvolvidos do que outras. Há diversas categorias que possivelmente envolvem aplicações de diversos formatos e que solicitam diferentes tipos de permissões, o que garante uma amostra bem variada com uma quantidade satisfatória de informação para o treinamento do modelo de redes neurais. Vale também observar a possibilidade

de haver uma diferença substancial na frequência das categorias quando lidamos com as duas diferentes classes (Malignos e Benignos).

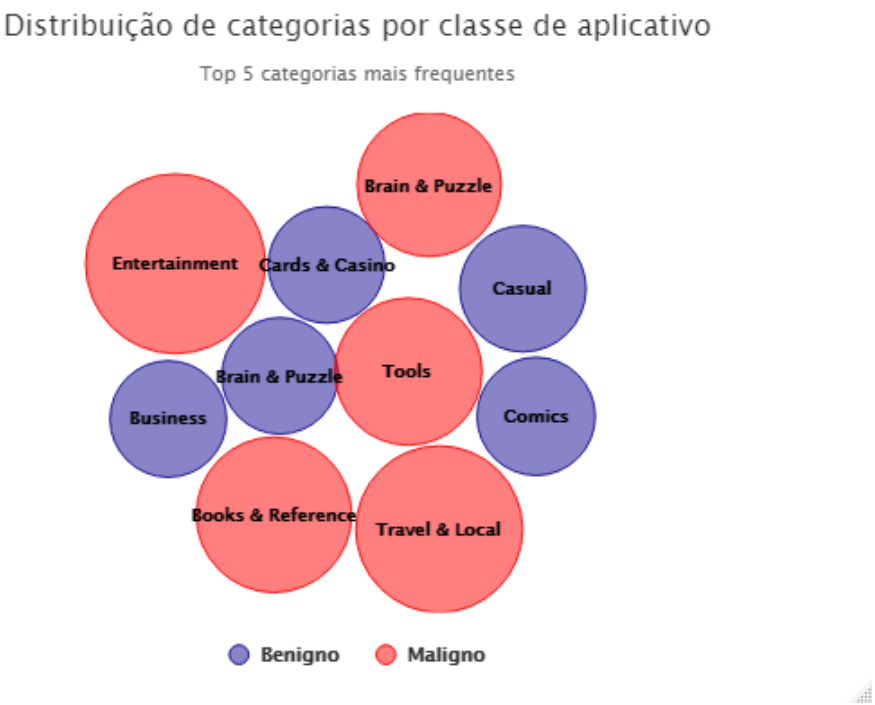

Figura 12 – Gráfico *Packedbubble*: Categorias em relação as classes.

É interessante notar quando se extrai as 5 categorias mais frequentes de cada classe, na **Figura *12***, que a maior frequência dos aplicativos maliciosos no banco de dados se concentra nas categorias de Entretenimento, Viagens e Livros, enquanto as 3 mais frequentes dos aplicativos benignos são: Casual, Cartas e Cassino e Negócios. Portanto, esta variável se mostra importante quando se trata de classificação, pois demonstra que há diferença nas categorias em cada tipo de classe de aplicativo. A única categoria que foi presente em ambas as classes foi “Estratégia e Quebra Cabeça”.

Acerca das permissões, há uma variável no banco de dados que informa a quantidade de permissões perigosas solicitadas pelos aplicativos, ou seja, aquelas permissões que são potencialmente perigosas e por consequência dão mais liberdade e oportunidades para que haja invasões de privacidade, roubo de informações ou até mesmo espionagem, além de outros riscos diversos.

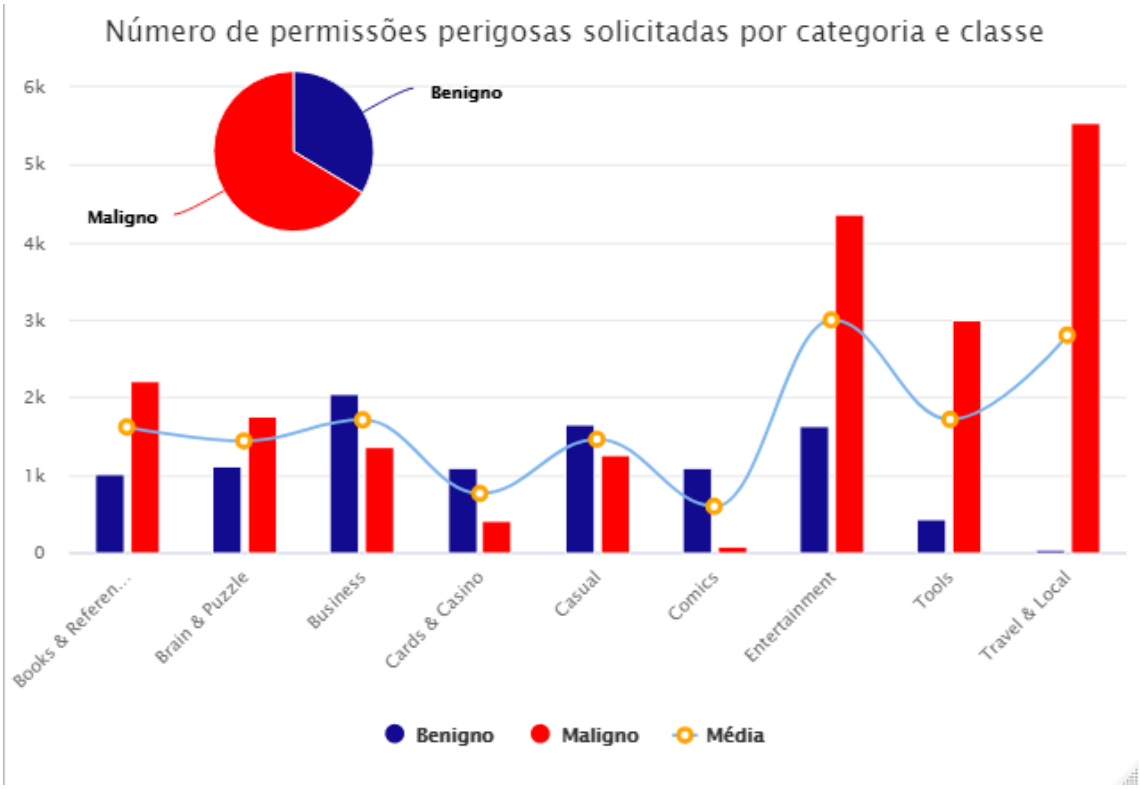

Figura 13 – Número de permissões perigosas solicitadas em relação a classe e a categoria.

Observando a **Figura *13***, onde foram selecionadas as 5 categorias mais presentes em cada classe, totalizando as 10 mais frequentes, nota-se que os aplicativos maliciosos solicitam um número de permissões perigosas em média muito maior do que os aplicativos benignos. Essa informação é valiosa, pois indica que há uma forte prevalência em algumas categorias quando se diz respeito a quantidade de permissões perigosas solicitadas para a classe de aplicativos maliciosos, principalmente nas 3 últimas categorias, onde os valores estão muito acima da média. Vale destacar a categoria “Comics” onde, mesmo tendo maior prevalência em aplicativos Benignos, a diferença não é tão extrema quanto quando observamos “Viagem e Local”, por exemplo, do lado oposto, isso demonstra mais uma vez que a variável categoria tem grande valor discriminativo para o banco de dados.

Por fim, a avaliação do usuário pode fornecer importantes insights e possíveis diferenças que ocorrem entre as classes nesse sentido, como mostra a **Figura *14***.

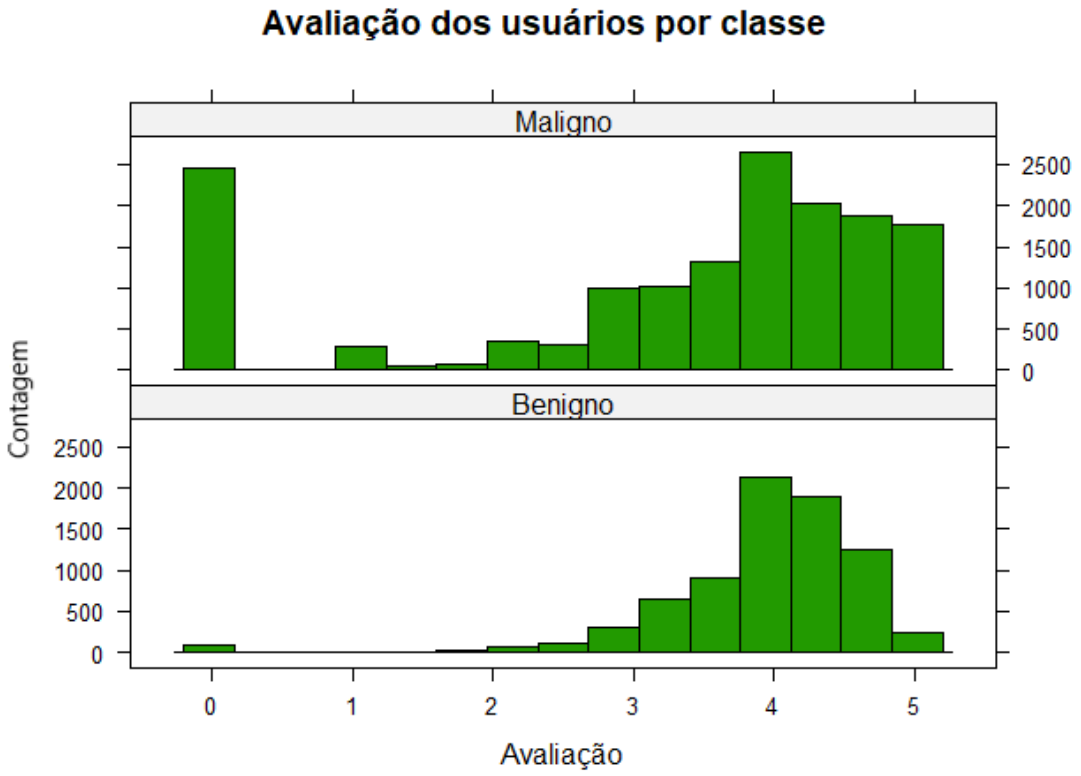

Figura 14 – Avaliação dos usuários por classe.

É notório que há uma prevalência muito maior de avaliações próximas a 0 para os aplicativos malignos, o que novamente faz muito sentido, indicando forte insatisfação daqueles que fizeram download e possivelmente se arrependeram em algum momento. Já na região de notas mais altas, próximas a 4, os aplicativos benignos novamente se diferenciam, dessa vez com menos variância nessa região de pontuação, enquanto na classe de malwares, há uma maior variação, com muitos aplicativos também tendendo a nota 5, que é a nota máxima. Com toda certeza essa variável contém informação que auxiliará de forma consistente no modelo, dado o que foi visto através desta análise descritiva.

4.2 Rede Neural

4.2.1 Parâmetros

Para que a tarefa de obter um bom classificador seja alcançado através de uma rede neural, é crucial que se encontre a quantidade certa de

neurônios que irão compor as camadas ocultas. Como se trata do *multilayer perceptron*, há apenas uma camada oculta, como mostra a **Figura *6***. Depois de sucessivos testes utilizando o método de validação cruzada com repetição K-fold com 10 partições, a quantidade de neurônios que se mostrou ideal e que gerou uma melhor acurácia foi 62, como mostra a **Figura *15***. Nota-se que a partir de certo ponto, aumentar a quantidade de neurônios não necessariamente aumenta a capacidade do modelo de produzir uma melhor classificação. Isso ocorre, pois há limitações quando se trata de redes neurais com apenas uma camada oculta. Sendo assim, fazer testes com uma quantidade superior à apresentada poderia se mostrar ineficiente, pois além de custar muito poder computacional, torna o modelo mais complexo, sendo mais interessante utilizar o princípio da parcimônia.

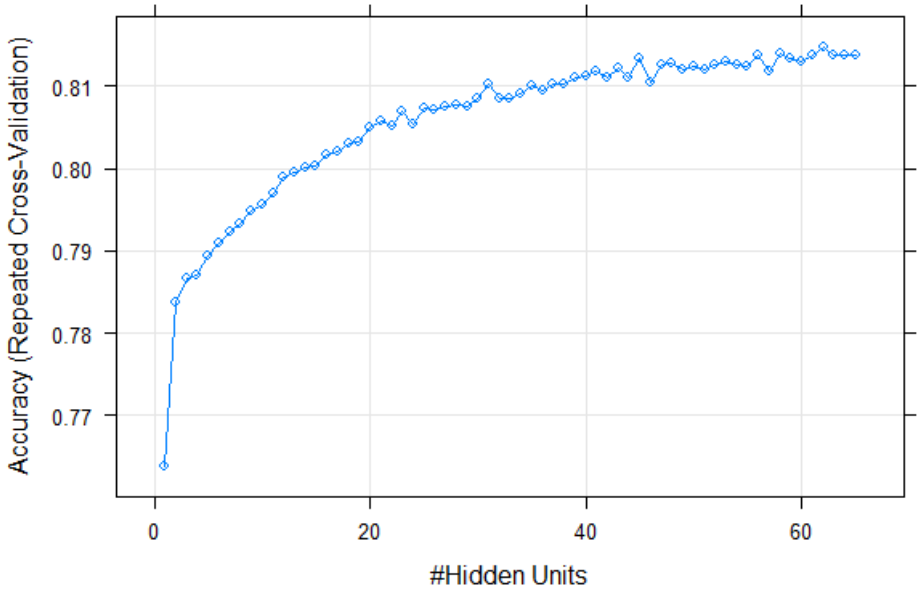

Figura 15 – Acurácia alcançada de acordo com a quantidade de neurônios.

Da mesma forma, o índice Kappa corroborou com a quantidade ideal de neurônios escolhida, como mostra a **Tabela *3***.

Tabela 3 – Índice capa em relação á quantidade de neurônios.

Quantidade de Neurônios	Kappa
1	0,5275
2	0,5674
3	0,5734
4	0,5742
5	0,5789
6	0,582
7	0,5845
8	0,5864
9	0,5896
10	0,5911
11	0,5939
12	0,598
13	0,5992
14	0,6002
15	0,6008
16	0,6033
17	0,604
18	0,6062
19	0,6065
20	0,6098
21	0,6113
22	0,6104
23	0,6139
24	0,6106
25	0,6145
26	0,6142
27	0,6151
28	0,6155
29	0,6149
30	0,6169
31	0,6206
32	0,6168

Quantidade de Neurônios	Kappa
33	0,6169
34	0,6183
35	0,6201
36	0,6188
37	0,6207
38	0,6207
39	0,6221
40	0,6226
41	0,6236
42	0,622
43	0,6244
44	0,622
45	0,6266
46	0,6209
47	0,625
48	0,6256
49	0,624
50	0,6246
51	0,6239
52	0,6252
53	0,6258
54	0,6253
55	0,6248
56	0,6276
57	0,6235
58	0,6278
59	0,6267
60	0,6261
61	0,6274
62	0,6297
63	0,6276
64	0,6277

A função de ativação utilizada, como indicado na sessão de metodologia, foi a função Sigmoide (ou Logística), sendo ideal para modelos de classificação onde é possível dividir ou discriminar as classes com valores de 0 ou 1.

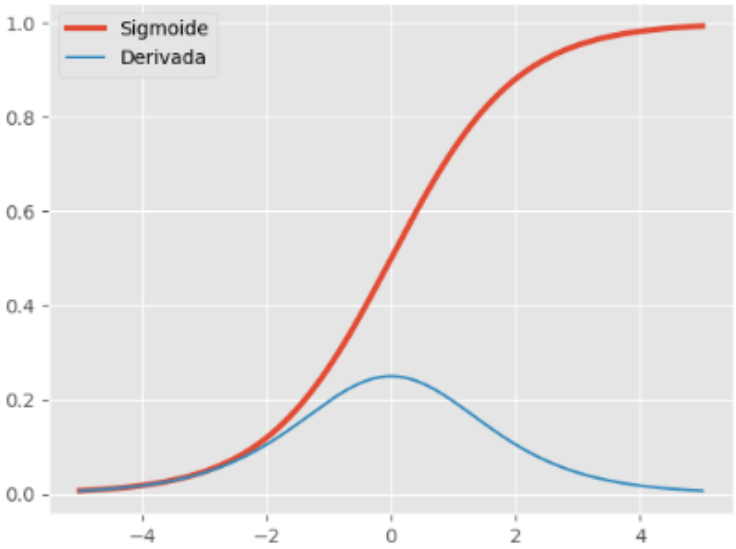

Figura 16 – Função Sigmoide e sua Derivada.

Os pesos iniciais foram definidos aleatoriamente, pois a rede está sendo treinada pela primeira vez e o método para ajuste dos pesos será *backpropagation*. A rede terá 2 outputs (0 e 1), correspondentes às classes de aplicativos benignos e malignos, respectivamente. Antes de partir para os resultados, é notório apresentar a informação de que para obter os parâmetros indicados foram realizadas diversas tentativas, todas geraram um valor de acurácia menor do que 75%. As tentativas anteriores envolveram o treinamento de modelos que:

- Envolviam um número muito superior de variáveis (acima de 100 variáveis).
- Não tornavam a variável “categoria” uma variável *dummy*.
- Aplicavam o método de *undersampling* ao invés de *oversampling* para balanceamento das classes.

Todas estas tentativas foram extremamente importantes para entender onde era necessário aplicar os diferentes tratamentos a fim de obter um resultado de acurácia superior. Dito isto, já é possível seguir para a análise dos resultados obtidos.

4.2.2 Modelo Final

Há 4 medidas muito relevantes quando se trata de mensurar o desempenho de uma rede neural. A primeira se chama acurácia, sendo a medida que irá informar quanto o modelo está classificando corretamente cada uma das classes. Essa medida traz um importante ponto de atenção sobre dados que são desbalanceados, pois caso a acurácia do modelo esteja menor ou igual à proporção da classe mais frequente, isso indica que na verdade seu modelo está apenas classificando todas as observações como sendo da classe mais frequente, com certeza não é esse resultado que se espera.

Tabela 4 – Matriz de confusão.

Matriz de Confusão			
		Referência	
		Benignos	Malignos
Previsão	Benignos	11004	2193
	Malignos	1136	9947

No caso da rede neural ajustada neste trabalho, o valor obtido para a acurácia foi de 86%, sendo um valor satisfatório e que gera um bom nível de acerto nas classificações, visto que a classe mais frequente obtinha antes do início do tratamento 66% da frequência no banco de dados.

Tabela 5 – Medidas de desempenho de modelo.

Estatísticas	
Acurácia	86%
Sensibilidade	82%
Especificidade	91%
Kappa	0,75

A segunda medida relevante é a sensibilidade (ou recall). Essa importante métrica auxilia a entender o nível de classificação quando a ocorrência é positiva, ou seja, quando o aplicativo é malicioso. Com essa medida

é possível dizer que o modelo de rede neural deste trabalho consegue classificar com sucesso cerca de 8 a cada 10 aplicações maliciosas provenientes da base de dados utilizada. Esse é um grande resultado.

Figura 17 – Assertividade: 8 a cada 10 malwares.

A métrica especificidade informa a porcentagem de classificação correta dos aplicativos benignos. Por último, o índice Kappa fornece uma métrica para a qualidade da classificação, sendo utilizada para esta análise a classificação sugerida por Landis et al. (1977), mostrada na Tabela 6.

Tabela 6 – Coeficiente Kappa.

Coeficiente Kappa	
< 0	Péssima
0-0,2	Ruim
0,21-0,4	Razoável
0,41-0,6	Moderada/Boa
0,61-0,8	Muito Boa
0,81-1,0	Excelente

De forma geral, os parâmetros indicam que a rede neural obtida possui um bom nível de Acurácia, Sensibilidade, Especificidade e obteve a classificação de muito boa no que se diz respeito ao Coeficiente Kappa.

5. CONSIDERAÇÕES FINAIS

As estatísticas descritivas trouxeram importantes visualizações para os dados antes do ajuste do modelo, que apresentou um bom nível de classificação, sendo capaz de classificar uma porcentagem significativa dos aplicativos corretamente do banco de dados. Os métodos utilizados no pré-processamento foram cruciais para o bom desempenho do modelo, todos os testes feitos anteriormente sem a aplicação das mais variadas técnicas aqui apresentadas, resultaram em um nível de acurácia menor ao alcançado. O algoritmo de redes neurais artificiais utilizado se mostrou muito útil e fácil de ser configurado, apenas tendo como ponto negativo o tempo de treinamento, levando em alguns testes mais de 3 dias para ser finalizado. Como sugestão para próximos trabalhos, a aplicação de um modelo mais robusto utilizando redes neurais profundas poderá resultar em uma classificação ainda melhor. Outra técnica que poderá trazer bons resultados é o *Random Forest*, como mostrou o algoritmo RFE que foi aplicado para a seleção das variáveis.

6. REFERÊNCIAS

ALI ALATWI, H.; OH, T.; FOKOUE, E.; STACKPOLE, B., 2016, **Android malware detection using category-based machine learning classifiers**.

CARVALHO; DE PADUA BRAGA, A.; LUDERMIR, T. B., 1998, **Fundamentos de redes neurais artificiais**.

CHENG, J.; WONG, S. H.; YANG, H.; LU, S., 2007, **Smartsiren: virus detection and alert for smartphones**.

CUNHA, F. L.; FRANCA, J. E.; ORTOLAN, R. L.; JUNIOR CLIQUET, A., 2007, **O uso de redes neurais artificiais para o reconhecimento de padrões em uma prótese mioelétrica de mão**.

Data Science Academy. **Deep Learning Book**, 2022. Disponível em: <https://www.deeplearningbook.com.br/ o-perceptron-parte-2/ >. Acesso em: 10 Janeiro. 2022.

DUFOUR, 1980, **Dummy variables and predictive tests for structural change**.

ENCK, W.; ONGTANG, M.; MCDANIEL, P., 2009, **On lightweight mobile phone application certification**.

FELT, A. P.; CHIN, E.; HANNA, S.; SONG, D. *et al.*, 2011, **Android permissions demystified**.

Facebook suspende Cambridge Analytica por violação de políticas. *In*: SHAKIL, Ismail. Globo.com. [*S. l.*], 17 mar. 2018. Disponível em: https://g1.globo.com/economia/tecnologia/noticia/facebook-suspende-cambridge-analytica-por-violacao-de-politicas.ghtml. Acesso em: 10 fev. 2022.

FURTADO, 2019, **Redes neurais artificiais: uma abordagem para sala de aula.**

GRACE, M.; ZHOU, Y.; ZHANG, Q.; ZOU, S. *et al.*, 2012, **Riskranker: scalable and accurate zero-day android malware detection**.

MAHINDRU, A.; SINGH, P., 2017, **Dynamic permissions based android malware detection using machine learning techniques**.

MCCULLOCH, W. S.; PITTS, 1943, **A logical calculus of the ideas immanent in nervous activity.**

OLIVEIRA, 2019, **Uso de aprendizagem de máquina e redes neurais convolucionais profundas para a classificação de áreas queimadas em imagens de alta resolução espacial.**

SHABTAI, A.; KANONOV, U.; ELOVICI, Y.; GLEZER, C. *et al, 2012,* **“Andromaly”: a behavioral malware detection framework for android devices.**

R. STEVENS, J. GANZ, V. FILKOV, P. DEVANBU AND H. CHEN, *2013,* **Asking for (and about) permissions used by Android apps.**

ANDROID. **Permissões no Android**. Disponível em: https://developer.android.com/guide/topics/permissions/overview?hl=pt-br. Acesso em: 8 de jan. 2022

EMIL HVITFELDT, 2021, **themis: Extra Recipes Steps for Dealing with Unbalanced Data.** R package version 0.1.4. https://CRAN.R-project.org/package=themis

LANDIS; J. RICHARD; GARY G. KOCH, 1977, **The Measurement of Observer Agreement for Categorical Data.**

7. ANEXOS

Anexo A - Output do Software R. Variáveis selecionadoas:

[1] "number_of_ratings" , [2] "category_Travel...Local" , [3] "category_Tools",

[4] "price", [5] "category_Comics", [6] "category_News...Magazines"

[7] "category_Transportation", [8] "category_Sports"

[9] "category_Libraries...Demo", [10] "category_Productivity"

[11] "category_Shopping" , [12] "category_Music...Audio"

[13] "category_Cards...Casino" , [14] "category_Entertainment"

[15] "category_Business" , [16] "category_Social" ,[17] "category_Photography"

[18] "category_Health...Fitness" , [19] "rating"

[20] "dangerous_permissions_count" , [21] "category_Weather"

[22] "storage_._modify_delete_usb_storage_contents_modify_
delete_sd_card_contents_d"

[23] "network_communication_._view_wi.fi_state_s"

[24] "safe_permissions_count" , [25] "hardware_controls_._control_vibrator_s"

[26] "category_Casual" ,[27] "system_tools_._prevent_device_from_sleeping_d"

[28] "network_communication_._view_network_state_s"

[29] "phone_calls_._read_phone_state_and_identity_d"

[30] "category_Arcade...Action"

[31] "your_location_._coarse_network.based_location_d"

[32] "category_Medical" , [33] "category_Books...Reference"

[34] "system_tools_._change_network_connectivity_d"

[35] "category_Education", [36] "your_location_._fine_gps_location_d"

[37] "system_tools_._automatically_start_at_boot_s"

[38] "your_personal_information_._read_contact_data_d"

[39] "category_Brain...Puzzle" , [40] "category_Communication"

[41] "category_Finance" , [42] "category_Sports.Games"

[43] "category_Media...Video"

[44] "hardware_controls_._take_pictures_and_videos_d"

[45] "services_that_cost_you_money_._directly_call_phone_numbers_d"

[46] "hardware_controls_._record_audio_d"

[47] "system_tools_._set_wallpaper_s" , [48] "category_Racing"

[49] "category_Lifestyle"

[50] "network_communication_._receive_data_from_internet_s"

[51] "system_tools_._modify_global_system_settings_d"

[52] "your_accounts_._discover_known_accounts_s"

[53] "services_that_cost_you_money_._send_sms_messages_d"

[54] "your_personal_information_._write_contact_data_d"

[55] "system_tools_._kill_background_processes_s"

[56] "hardware_controls_._change_your_audio_settings_d"

[57] "system_tools_._mount_and_unmount_filesystems_d"

[58] "category_Personalization", [59] "hardware_controls_._control_flashlight_s"

[60] "your_personal_information_._read_sensitive_log_data_d"

[61] "system_tools_._change_wi.fi_state_d"

[62] "system_tools_._retrieve_running_applications_d"

[63]"your_personal_information_._write_browser_s_history_and_bookmarks_d"

[64] "your_location_._mock_location_sources_for_testing_d"