UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE FÍSICA

CURSO DE GRADUAÇÃO EM FÍSICA

PAULO CLEBER FARIAS DA SILVA FILHO

UM ESTUDO SOBRE A DETECÇÃO DE EXOPLANETAS COM INTELIGÊNCIA ARTIFICIAL

FORTALEZA

2022

PAULO CLEBER FARIAS DA SILVA FILHO

UM ESTUDO SOBRE A DETECÇÃO DE EXOPLANETAS COM INTELIGÊNCIA ARTIFICIAL

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Física do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Física.

Orientador: Prof. Dr. Daniel Brito de Freitas

FORTALEZA

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S582e Silva Filho, Paulo Cleber Farias da.
Um estudo sobre a detecção de exoplanetas com inteligência artificial / Paulo Cleber Farias da Silva Filho. – 2022.
83 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Ciências, Curso de Física, Fortaleza, 2022.
Orientação: Prof. Dr. Daniel Brito de Freitas.

1. exoplanetas. 2. inteligência artificial. 3. missão Kepler. 4. rede neural. I. Título.

CDD 530

PAULO CLEBER FARIAS DA SILVA FILHO

UM ESTUDO SOBRE A DETECÇÃO DE EXOPLANETAS COM INTELIGÊNCIA ARTIFICIAL

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Física do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Física.

Aprovada em: 04 de Fevereiro de 2022

BANCA EXAMINADORA

Prof. Dr. Daniel Brito de Freitas (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Saulo Davi Soares e Reis
Universidade Federal do Ceará (UFC)

Prof. Dr. Luidhy Santana da Silva
Universidade Cruzeiro do Sul (UNICSUL)

Dedico este trabalho à minha família, às pessoas amigas e a todos os seres que me inspiraram a ser alguém melhor.

AGRADECIMENTOS

Todas as caminhadas de grandes percursos são repletas dos mais variados obstáculos e desafios. Eu não poderia ter superado os meus sem a influência de algumas pessoas (muitas, na verdade) a quem dedico este espaço para agradecer. Primeiramente à minha mãe, Alice, pelo apoio e amor incondicional ao longo de todas as etapas da minha vida. Agradeço também à minha família por todo o apoio.

Nesta minha jornada acadêmica até agora, agradeço ao meu orientador, o professor Daniel Brito de Freitas, pelo seu companheirismo, confiança e por me abrir as portas para o universo. Agradeço ao professor Saulo Davi Soares e Reis por ser uma das minhas maiores inspirações na minha graduação e, em especial, por incentivar indiretamente a realização deste trabalho. Agradeço ao professor Luidhy Santana da Silva por aceitar fazer parte da banca. Agradeço ao professor Afrânio de Araújo Coelho pelo apoio ao nosso grupo de pesquisa, o Stellar Team. Agradeço ao professor José Ramos Gonçalves pelas conversas descontraídas e, principalmente, pela sua vontade de ensinar. Agradeço ao professor Eduardo Bedê Barros por toda a parceria para com as gerações mais recentes do Diretório Acadêmico da Física. Agradeço também a todos os demais professores da Universidade Federal do Ceará que, de alguma forma, me influenciaram a crescer. Agradeço também aos professores e professoras que contribuíram para a minha formação, que acreditaram em mim e que me apoiaram ao longo do ensino fundamental e médio.

Agradeço às pessoas que fazem parte do Stellar Team. Em especial aos meus parceiros de bolsa José Ribamar Júnior, que esteve comigo no começo e em boa parte do desenvolvimento da minha iniciação científica, e Fernando Lima Filho, por compartilhar o entusiasmo com machine learning e ciência de dados em geral.

Agradeço a todas as pessoas com quem troquei experiencias ao longo do curso. Em especial à Lara Hissa, Cassimiro Albuquerque, Isabel Castro, Vasco Stascxak e Carlos Miguel, que foram as minhas amizades mais marcantes no departamento (e cuja ordem apresentada dos nomes foi escolhida de forma justa pelo nosso querido `d20`). Agradeço também ao Eduardo Rocha, Raul Batista, Sabrina Sá, Alessandro Magalhães, Ingrid Lima, Isis do Vale, Rayanne Lemos, à turma veterana que nos recebeu, às pessoas da minha gestão do Diretório Acadêmico, além de várias outras pessoas do departamento e de toda UFC que marcaram de alguma forma.

Agradeço à minha namorada, Tayná Alves, que me inspira de várias maneiras possíveis e que desperta o melhor de mim, por seu carinho, companheirismo e pelo seu sorriso

doce. Agradeço à Luanna Sousa, Felipe Lima, Wilker Cunha e Daniel Oliveira por mais de uma década de amizade e por todo apoio. Agradeço à Pamela Brenna por não desistir da minha luz, mesmo vendo muitas das minhas trevas. Agradeço ao professor Cleidson Costa, Eduardo Morais, Alexandre Laureano, Thales Wylmar, Gabriel Albano, Yury Mota e Vanessa Santos, por terem compartilhado comigo uma das minhas melhores experiências como estudante por meio das feiras de ciência nacionais e internacionais que participamos. Levo todas essas amizades para a vida. Um agradecimento especial à professora Rhonda, que ouvia as minhas poesias, a minha paixão pelo universo e as minhas visões de mundo durante a temporada de intercâmbio que passei no Canadá e que se tornou uma das minhas melhores lembranças dessa época.

Agradeço ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio financeiro por meio do Programa Institucional de Bolsas de Iniciação Científica (PIBIC) ao longo de parte da minha formação.

Por último, mas não menos importante, agradeço a mim mesmo por nunca ter dado ouvidos aos motivos que me mandavam desistir.

"Não perguntamos para que finalidade útil os pássaros cantam, pois o canto é o prazer deles, já que foram criados para cantar. Da mesma forma, não devemos perguntar por que a mente humana se preocupa em sondar os segredos dos céus... A diversidade dos fenômenos da Natureza é tão grande e os tesouros escondidos nos céus tão ricos, precisamente para que a mente humana nunca careça de alimento fresco."

(Johannes Kepler)

RESUMO

Missões espaciais como CoRoT, Kepler e TESS são essenciais na astrofísica para a busca de exoplanetas. Milhares de planetas já foram detectados com o aperfeiçoamento das técnicas de busca. No entanto, majoritariamente a maioria dos exoplanetas foram descobertos por inspeção visual das curvas de luz obtidas com as missões CoRoT e Kepler. Esses processos estão sujeitos a erros humanos e são impraticáveis quando um grande volume de dados está envolvido. As futuras missões espaciais PLATO e James Webb Space Telescope trarão volumes ainda maiores, o que apresenta um sério desafio para a metodologia de pesquisa que vinha sendo feita. Com base nisso, são necessários novos métodos para tratar e analisar as curvas de luz fotométricas provenientes dessas missões. O uso de inteligência artificial vem ganhando cada vez mais relevância na ciência de dados voltada às pesquisas astrofísicas e, particularmente, se mostra como uma ferramenta promissora para detecção automática de exoplanetas. Este trabalho tem como objetivo aplicar Aprendizado de Máquina para a detecção de exoplanetas com base nos dados gerados pela missão Kepler. Para tal, modelos computacionais como Floresta Randômica e Redes Neurais foram testados na implementação de um algoritmo que será usado para explorar as curvas de luz de qualquer missão. No que tange a presente pesquisa, mais de 15 mil curvas de luz foram divididas em três conjuntos: um para o treino dos modelos, outro para a validação dos mesmos e outro para testes. Sabe-se previamente quais curvas de luz possuem ou não planetas confirmados. Os resultados preliminares mostram que o nosso melhor modelo implementado foi o de Rede Neural, com uma acurácia de 90% nas classificações das curvas no conjunto de testes. Ele também conseguiu detectar dois planetas confirmados em dados adicionais com bastante ruído. Isso indica que novos casos de interesse podem ser descobertos ao implementar nosso algoritmo em um conjunto inexplorado de dados. Essa é uma importante perspectiva do nosso trabalho quando os dados das missões PLATO e James Webb estiverem disponíveis.

Palavras-chave: exoplanetas; inteligência artificial; missão Kepler; rede neural.

ABSTRACT

Space missions like CoRoT, Kepler and TESS are essential in astrophysics for the search of exoplanets. Thousands of planets have already been detected with the improvement of search techniques. However, most exoplanets were discovered by visual inspection of the light curves obtained with the CoRoT and Kepler missions. These processes are subject to human error and are impractical when a large volume of data is involved. The future PLATO and James Webb Space Telescope space missions will bring even greater volumes, which presents a serious challenge to the research methodology that was being done so far. Based on this, new methods are needed to treat and analyze the photometric light curves coming from these missions. The use of artificial intelligence has been gaining more and more relevance in data science focused on astrophysical research and, particularly, it shows itself as a promising tool for the automatic detection of exoplanets. This work aims to apply Machine Learning for the detection of exoplanets based on the data generated by the Kepler mission. To this end, computer models such as Random Forest and Neural Networks were tested in the implementation of an algorithm that will be used to explore the light curves of any mission. Regarding the present research, more than 15 thousand light curves were divided into three sets: one for training the models, another for their validation and another for testing. It is previously known which light curves have confirmed planets or not. Preliminary results show that our best implemented model was the Neural Network, with an accuracy of 90% in the classification of the curves in the test set. It was also able to detect two planets confirmed in additional noisy data. This indicates that new cases of interest can be discovered by implementing our algorithm on an unexplored dataset. This is an important perspective of our work when data from the PLATO and James Webb missions become available.

Keywords: exoplanets; artificial intelligence; Kepler mission; neural network.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AEC Antes da Era Comum.

AFP *Astrophysical False Positive*. Label para especificar falsos positivos no conjunto de dados.

AUC *Area Under Curve*. Área sob a curva do gráfico da precisão do modelo em função do seu recall.

BJD *Barycentric Julian Date*. É o JD corrigido para diferenças da posição da Terra em relação ao baricentro do Sistema Solar.

BKJD *Barycentric Kepler Julian Date*. É o BJD menos uma constante de 2.454.833 dias, contante essa que corresponde às 12h do dia 1 de janeiro de 2009.

BLS *Box Least Squares*. Ferramenta estatística usada para detectar trânsitos planetários e eclipses binários em curvas de luz.

CART *Classification and Regression Tree*. Algoritmo usado na construção de árvores de decisão binárias.

CoRoT *Convection, Rotation and planetary Transits*. Primeira missão espacial projetada e destinada para pesquisas exoplanetárias.

DR *Data Release*. Sigla usada para identificar a versão dos dados liberados ao público.

ESA *European Space Agency*, ou Agência Espacial Européia.

FITS *Flexible Image Transport System*. Formato de aquivo amplamente usado pela comunidade astrônoma para o armazenamento, transporte e análise de dados científicos.

FR *Floresta Randômica*. Um modelo de aprendizagem de máquina baseado em *ensamble methods* e árvores de decisão.

GDE *Gradiente Descendente Estocástico*. Um modelo de aprendizagem de máquina baseado no de regressão logística, mas com caráter randômico.

HGA *High-Gain Antena*. Componente do telescópio Kepler usado no envio dos dados coletados à Terra.

HVC *Hard Voting Classifier*. Método de classificação baseado em votos majoritários de algoritmos classificadores usado em *ensamble methods*.

IAU *International Astronomical Union*, ou União Astronômica Internacional.

JD *Julian Day*. Contagem contínua de dias desde o início do período Juliano. Mais precisamente, essa contagem começou ao meio meio dia de 1 de janeiro de 4713 AEC.

MAST *Mikulski Archive for Space Telescopes*. Arquivo de dados astronômicos que contém dados de dúzias de missões como Hubble, Kepler e TESS.

NaN *Not A Number*. Valor de ponto flutuante que representa dados ausentes.

NASA *National Aeronautics and Space Administration*, ou Administração Nacional da Aeronáutica e Espaço. Agência do Governo Federal dos Estados Unidos.

NTP *Non-Transiting Phenomenon*. Label para especificar fenômenos que não apresentam características de trânsito no conjunto de dados.

PC *Planet Candidate*. Label para especificar fenômenos que apresentam características de trânsito no conjunto de dados.

PDC *Presearch Data Conditioning*. Um dos componentes do software usados no tratamento de cuvas de luz no pipeline do Kepler.

PDCSAP Sigla usada para especificar uma curva de luz gerada por fotometria de abertura tratada pelo PDC.

PLATO *PLAnetary Transits and Oscillations of stars*. Terceira missão espacial da ESA para fins de estudos de sistemas planetários extrasolares, previsto para ser lançada em 2026.

RN *Rede Neural* [Artificial]. Um modelo de aprendizagem de máquina baseado em redes neurais biológicas.

ROC *Receiver Operating Characteristic Curve*. Uma curva ROC é uma representação gráfica do desempenho de um modelo de classificação binária baseada na TPV em função da TFP.

ROC-AUC Sigla para designar a área sob a curva ROC.

SAP *Simple Aperture Photometry*. Técnica usada para gerar gráficos do fluxo fotométrico com base em fotometria de abertura.

SVC *Soft Voting Classifier*. Método de classificação baseado em votos de algoritmos classificadores com base nas probabilidades apresentados pelos mesmos usado em *ensamble methods*.

TCE *Threshold-Crossing Events*. Sequência de quedas que possuem características de trânsito planetário nas curvas de luz.

TESS *Transiting Exoplanet Survey Satellite*. Missão espacial que deu sequência às missões Kepler e K2.

TFP *Taxa de Falsos Positivos*. Razão do número dos falsos positivos pela soma dos falsos positivos com o negativos verdadeiros.

TPV *Taxa de Positivos Verdadeiros*. Outra nomenclatura para o recall, que é a porção dos verdadeiros positivos que foram identificados corretamente por um modelo em relação a todos os positivos que existem no conjunto de dados.

UNK *Unknown*. Label para especificar dados cuja classificação é desconhecida no dataset.

WASP *Wide Angle Search for Planets*. Consórcio internacional de várias organizações acadêmicas que busca realizar uma vasta varredura em todo o céu para a busca de exoplanetas usando fotometria de trânsito.

SUMÁRIO

1 INTRODUÇÃO

Um exoplaneta é um planeta que está fora do sistema solar. Por conta disso, ele também é chamado de planeta extrassolar. A definição de planeta é motivo de debate até hoje, mas a última resolução da *União Astronômica Internacional* (IAU) sobre este tema diz que um planeta é um corpo celestial que (texto adaptado):

1. está em órbita ao redor de uma estrela;
2. tem massa suficiente para que sua própria gravidade se sobressaia sobre as forças de corpo rígido, de forma que ele assuma um equilíbrio hidrostático (aproximadamente redondo);
3. limpa a vizinhança em torno de sua órbita.

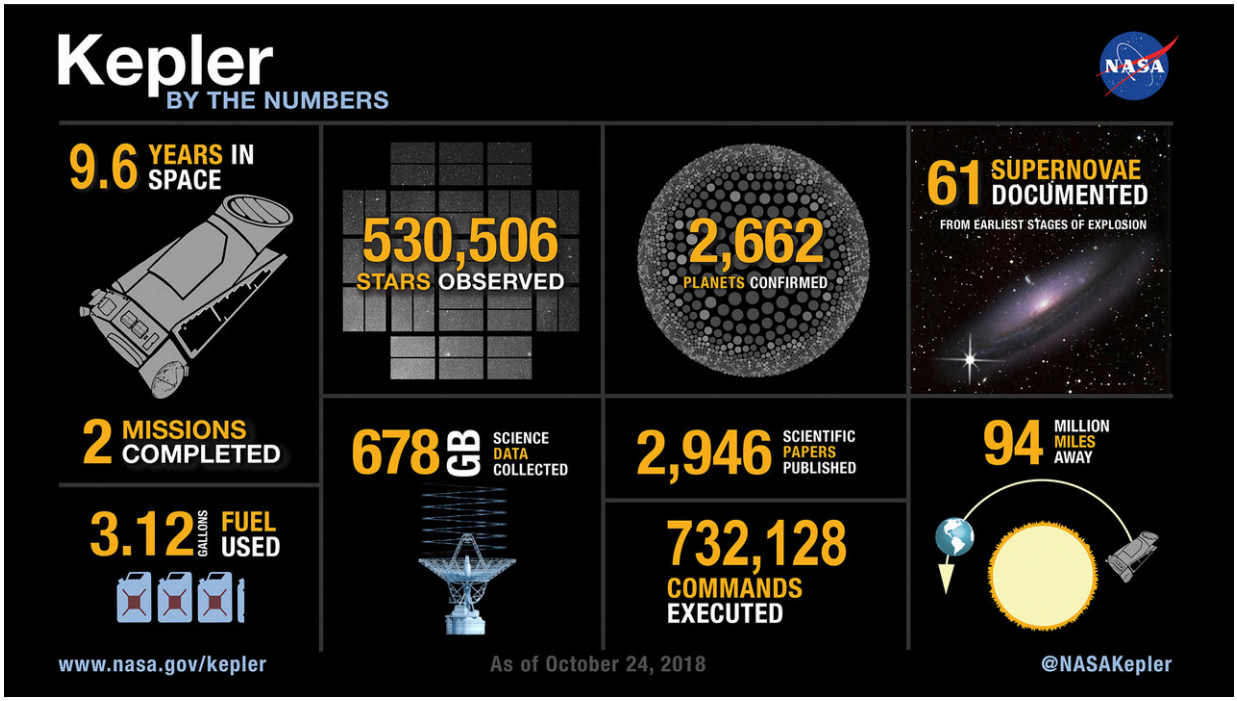
A descoberta de exoplanetas põe em xeque a excepcionalidade que o planeta Terra tem no universo (WENZ; POWELL, 2019). Além disso, ela é fundamental no estudo e desenvolvimento de teorias de formação planetária. De fato, toda essa área teve um desenvolvimento dramático depois que os primeiros planetas extrassolares foram descobertos (ARMITAGE, 2010).

O primeiro indício da existência de um planeta fora do sistema solar data de 1917 (GREICIUS, 2017). Essa discussão só tomou proporções mais sérias e embasadas em 1988, quando foi especulado que um planeta estaria orbitando a estrela Gamma Cephei (CAMPBELL *et al.*, 1988), apesar de ser inconclusiva por mais de uma década (HATZES *et al.*, 2003). O que geralmente se considera a primeira detecção genuína de um exoplaneta aconteceu em meados de 1992, onde foram descobertos dois planetas orbitando um estrela pulsar (WOLSZCZAN; FRAIL, 1992). No final de 1995 aconteceu a primeira detecção de um exoplaneta orbitando uma estrela semelhante ao Sol (MAYOR; QUELOZ, 1995), o que rendeu parte do Prêmio Nobel em Física de 2019 para para Michel Mayor e Didier Queloz.

Desde então, a área de exoplanetologia vem crescendo cada vez mais ao ponto de projetos inteiros serem dedicados a esse fim, como observatórios terrestres e missões espaciais. O *Wide Angle Search for Planets* (WASP) é a colaboração de pesquisas terrestres mais bem sucedida atualmente, tendo encontrado mais de 150 planetas (HELLIER, 2019). Projetos terrestres são muito importantes nas pesquisas astrofísicas, mas o maior número de contribuições vem de missões espaciais, tais como CoRoT, Kepler, K2 e TESS. Até o final de novembro de 2021, exatamente 4.575 exoplanetas foram descobertos no total (BRENNAN, 2020). Os gráficos da figura 40 mostram uma relação do número de planetas descobertos ao longo dos anos.

Cada uma dessas missões citadas leva o nome de seu telescópio, com exceção da K2, que também usou o *Kepler Space Telescope*[1] em suas observações. As duas missões juntas deram um tempo de vida útil ao Kepler de mais de nove anos, começando em 2009 e terminando em 2018. A *TESS Exoplanet Mission* foi a sucessora da K2. A figura 1 resume o legado deixado pelo Kepler.

Figura 1 – Legado do Kepler. Dados atualizados até outubro de 2018.



Fonte: NASA, disponível em https://exoplanets.nasa.gov. Acesso em 29 de novembro de 2021.

É possível acompanhar em tempo real o número de exoplanetas descobertos pela missão Kepler e pelas outras missões da NASA por meio do *NASA Exoplanet Archive*. Lá é possível ver que o número de exoplanetas confirmados pelo Kepler aumentou desde quando o mesmo foi aposentado, o que mostra que a comunidade científica não foi capaz de absorver todo o conteúdo da missão em seus nove anos de atividade. Novos telescópios ainda mais potentes estão sendo projetados para contribuírem cada vez mais com os estudos da exoplanetologia e do universo em geral. O *James Webb Telescope* (NASA), lançado em 25 de dezembro de 2021, e o PLATO (ESA), previsto para ser lançado em 2026, são exemplos deles.

Diante disso, a análise do grande volume de dados gerados pelos telescópios espaciais se torna cada vez mais desafiadora. Ainda nesta década, será humanamente impossível absorver todos esses dados em tempo viável, uma vez que novas missões virão com novos dados em

[1] Daqui em diante, o *Kepler Space Telescope* será referenciado apenas como Kepler. Não confundir com a missão Kepler.

uma quantidade ainda maior que as das gerações anteriores. Felizmente, o uso de inteligência artificial vem ganhando cada vez mais espaço nas pesquisas científicas e pode ajudar nessa tarefa (MONTGOMERY, 2019).

O Aprendizado de Máquina, ou Machine Learning, é uma subárea do que se entende por Inteligência Artificial. Existem algumas definições famosas para este termo, tal como a do Arthur Samuel (SAMUEL, 1959):

> **Definição 1.** *Aprendizado de Máquina é o campo de estudo que dá aos computadores a habilidade de aprender sem serem explicitamente programados.*

Samuel diz em seu artigo que uma máquina aprende por experiência, mas não dá uma definição clara do que exatamente significa "aprender". Tom Mitchell esclarece isso algumas décadas depois (MITCHELL, 1997):

> **Definição 2.** *Diz-se que um programa de computador aprende com a experiência **E** em relação a alguma tarefa **T** e alguma medida de performance **P**, se sua performance em **T**, conforme medido por **P**, melhora com a experiência **E**.*

Veremos mais adiante que achar certos padrões nos dados astronômicos é um dos pontos mais cruciais na busca e detecção de exoplanetas. A metodologia adotada aqui é analisar exemplos de dados que representam ou não planetas para que o algoritmo possa aprender com eles e desenvolver um modelo que os diferencie e os *classifique* corretamente. Esse é o objetivo, ou *tarefa* (***T***), de todo este trabalho. Esses exemplos, chamados de *conjunto de treino*, são tidos como a *experiência* (***E***) do algoritmo[2]. Estudaremos mais adiante algumas medidas de performance (***P***), mas um exemplo simples seria a razão do número de classificações corretas sobre o número total de classificações. Essa medida particular tem o nome de *acurácia*. Dadas as especificações anteriores, estamos tratando de um treinamento supervisionado de um algoritmo de classificação. É supervisionado, pois sabemos de antemão quais dados representam ou não planetas.

[2] À primeira vista, pode parecer estranho para algumas pessoas que os próprios dados sejam a experiência do modelo e não o processo de aprendizagem em cima deles, dada a noção intuitiva de experiência que temos. No entanto, a definição 2 deixa bem claro que um programa de computador aprende *com a* experiência ***E***. Isso poderia ser facilmente substituído por "um programa de computador aprende com os dados ***D*** [...]".

Vale ainda mencionar duas preocupações importantes relacionadas à definição 2. É possível que um modelo se atenha bastante aos dados usados no seu processo de aprendizado. Disso, segue a definição:

> **Definição 3.** *Um **overfitting** se dá quando um modelo descreve muito bem os dados do conjunto de treino (a sua experiência **E**), mas não consegue generalizar bem a sua tarefa **T** para novos dados.*

O contrário também pode acontecer:

> **Definição 4.** *Um **underfitting** se dá quando um modelo é muito simples e não consegue extrair informações o suficiente de sua experiência **E** para realizar sua tarefa **T** satisfatoriamete.*

Em ambos os casos, a performance ***P*** do modelo não é boa. Com base na problemática exposta e nos conceitos introduzidos acima, o objetivo geral deste trabalho é:

Criar um algoritmo que usa aprendizado de máquina para classificar dados provenientes da missão Kepler como planetas ou não.

Para isso, os objetivos específicos são:

- Tomar como base a metodologia abordada por (SHALLUE; VANDERBURG, 2018);
- Entender os dados da missão Kepler;
- Revisar e aplicar o método de trânsito planetário para a detecção de exoplanetas;
- Baixar e tratar os dados do *Autovetter Planet Candidate Catalog for Q1-Q17 DR24* (CATANZARITE, 2015), que podem ser encontrados no *NASA Exoplanet Archive*;
- Revisar alguns dos vários algoritmos de aprendizado de máquina para tarefas de classificação;
- Criar modelos com base nesses algoritmos, testá-los e compará-los em busca daquele que apresenta a melhor performance;
- Aprimorar os parâmetros de aprendizagem, como discutido na seção 4.2, para aumentar ainda mais a performance do melhor modelo.
- Analisar a coerência do melhor modelo por meio das características astrofísicas que mais influenciam as suas classificações;
- Testar o melhor modelo com outros dados da missão Kepler, além do catálogo usado, de forma a avaliar os limites de aplicação do mesmo.

2 ENTENDENDO OS DADOS

Após uma contextualização geral do assunto e uma vez entendida a natureza do problema que queremos resolver, o próximo passo do trabalho é a obtenção, exploração e preparação dos dados para serem usados no treinamento do algoritmo. Todos os dados produzidos pelo Kepler para catálogo do Autovetter foram baixados através do Mikulski Archive for Space Telescopes (MAST). Naturalmente, um entendimento geral desses dados é necessário para que então possamos explorá-los.

2.1 A Missão Kepler e Seus Produtos

A missão Kepler foi desenvolvida para determinar a frequência de planetas com tamanhos similares ao da Terra que estejam próximos da zona de habitabilidade de estrelas semelhantes ao Sol (BORUCKI *et al.*, 2010). A zona de habitabilidade é uma região ao redor de uma estrela onde é possível encontrar água em seu estado líquido em superfícies de planetas rochosos por um período de tempo extenso (GONZALEZ *et al.*, 2001). Para essa finalidade, o telescópio passou quatro anos observando variações no brilho de aproximadamente 160.000 estrelas-alvo em uma região fixa, próxima às constelações de Cygnus e Lyra[3] (THOMPSON *et al.*, 2016).

Dados brutos[4] dificilmente estão prontos para serem usados para seus devidos fins e precisam der tratados por uma sucessão de operações específicas de cada contexto. O conjunto dessas operações leva o nome de pipeline de dados (QUEMY, 2019). Os dados brutos do Kepler são agrupados trimestralmente em pacotes, chamados de quarters e referenciados como Q0, Q1, Q2 e assim por diante[5], descompactados e então agrupados por cadência e tipo de pixel (JENKINS *et al.*, 2010c).

A cadência é o intervalo que em que os dados são coletados. No caso do Kepler, existem os dados de longa cadência e curta cadência. A longa cadência é caracterizada por intervalos de 29,4 minutos (JENKINS *et al.*, 2010a), ao passo que a curta cadência é caracterizada por intervalos de 1 minuto (GILLILAND *et al.*, 2010). Estaremos interessados nos dados de longa cadência neste trabalho. Além disso, os pixels coletados podem ser de três tipos: referentes

[3] A figura 41 mostra uma ilustração do campo de visão do Kepler.

[4] Por “dados brutos", me refiro a dados não tratados que vêm diretamente da fonte.

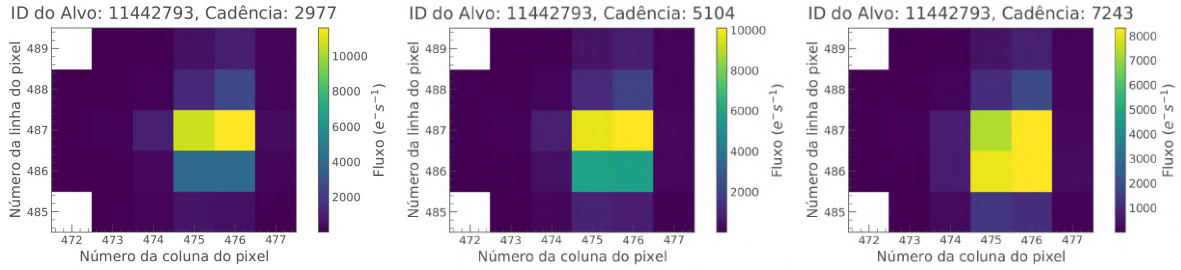
[5] A exceção desse padrão se dá aos dois primeiros quarters. O primeiro, Q0, se refere aos primeiros 10 dias da missão. O Q1 se refere a aproximadamente os 34 dias seguidos de Q0. Todos os demais quarters têm algo em torno de 90 dias.

a um alvo (target), referentes ao plano de fundo (background) ou referentes a dados colaterais usados para calibração (collateral).

Uma vez separados, esses dados são processados pelo *Kepler Science Processing Pipeline* e posteriormente arquivados. Esse pipeline é responsável, dentre outras coisas, por fornecer pixels calibrados e dados de fluxo fotométrico corrigidos (JENKINS *et al.*, 2010b). Após esse processamento, os dados são enfim armazenados em arquivos no formato FITS (Flexible Image Transport System)[6]. Por conta dos aprimoramentos nesse pipeline, os dados brutos são processados e arquivados muitas vezes, de forma que cada processamento é unicamente classificado com um número de liberação de dados (DR ou Data Release). No caso dos dados usados neste trabalho, estamos lidando com o DR24, como comentado no capítulo 1.

Como produto desse pipeline, temos *Target Pixel Files* (TPFs) tratados e curvas de luz, além dos próprios dados brutos[7]. Os TPFs também são usados nas missões K2 e TESS e cada um deles consiste em um conjunto de imagens e outros dados específicos de uma estrela-alvo[8]. Naturalmente, há uma imagem para cada cadência, como pode ser visto na figura 2, gerada com o auxílio do pacote *Lightkurve* do Python[9] (LIGHTKURVE COLLABORATION *et al.*, 2018).

Figura 2 – Imagens geradas a partir do TPF do Q6 da estrela Kepler-90.



Fonte: Elaboração própria.

Nessa imagem só é possível ver o brilho dos pixels da estrela-alvo (target), uma vez que o brilho do background já é removido no pipeline do Kepler. O brilho no background consiste tipicamente de *luz zodiacal*[10] (MORRIS *et al.*, 2020), que algumas vezes atrapalha a visualização de pequenas variações no fluxo. O background então fica “limpo", mas seus dados originais também estão armazenados no arquivo FITS e podem ser acessados para análises.

A partir do TPF de uma estrela, é possível extrair um gráfico que relaciona o seu

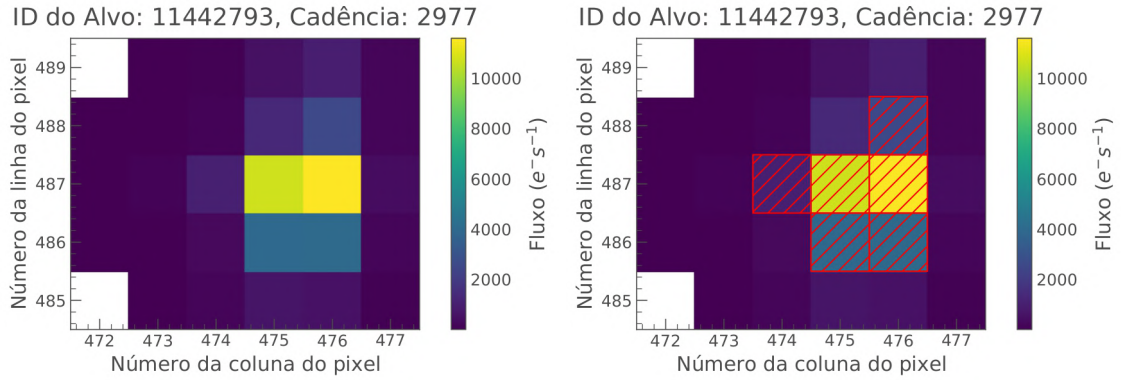
[6] Mais informações sobre esse tipo de arquivo aqui.
[7] Os dados brutos são úteis em alguns objetivos específicos, então eles também são liberados.
[8] Mais informações sobre os TPFs aqui.
[9] Todos os códigos para gerar as imagens estão no GitHub.
[10] Luz zodiacal é basicamente a luz de estrelas e outras fontes luminosas refletidas por poeira cósmica.

fluxo de luz com o tempo. Esse gráfico é chamado de *curva de luz* e é gerado a partir de técnicas de fotometria de abertura (CLEVE *et al.*, 2016). A técnica mais comum é a *Simple Aperture Photometry* (SAP), que consiste em selecionar os pixels que maximizam a razão sinal-ruído (SNR) do alvo para então somá-los e gerar uma curva de luz unidimensional (BRYSON *et al.*, 2010). A razão sinal-ruído, como o próprio nome sugere, é a potência do sinal (P_S) dividida pela potência do seu ruído (P_R) (JOHNSON, 2006), dada por

$$\text{SNR} = \frac{P_S}{P_R}. \tag{2.1}$$

A seleção dos pixels ótimos é feita pelo próprio pipeline do Kepler por meio de uma máscara de abertura e está armazenada no arquivo FITS. Contudo, é possível usar qualquer máscara previamente criada. A figura 3 mostra um exemplo da seleção dos melhores pixels para uma dada cadência por meio da máscara criada pelo pipeline do Kepler.

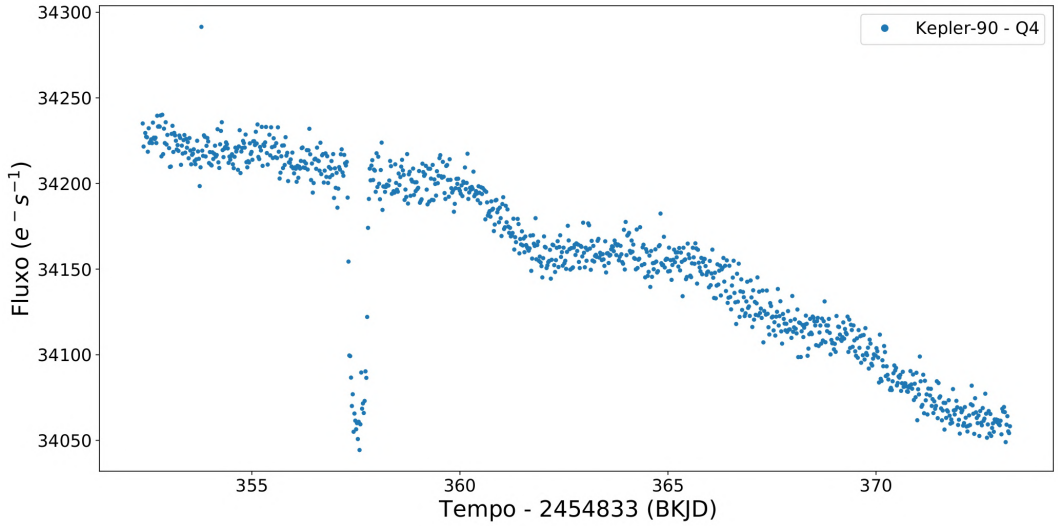
Figura 3 – Exemplo de seleção dos pixels que otimizam o SNR da estrela.



Fonte: Elaboração própria.

Esse processo é feito para todas as cadências do arquivo (tanto para os fluxos, quanto para os seus erros e outros parâmetros da medida) e os resultados são armazenadas em uma tabela. É possível gerar gráficos como os da figura 4 usando os dados de fluxo e tempo gerados.

Figura 4 – Exemplo de curva de luz gerada por SAP.



Fonte: Elaboração própria.

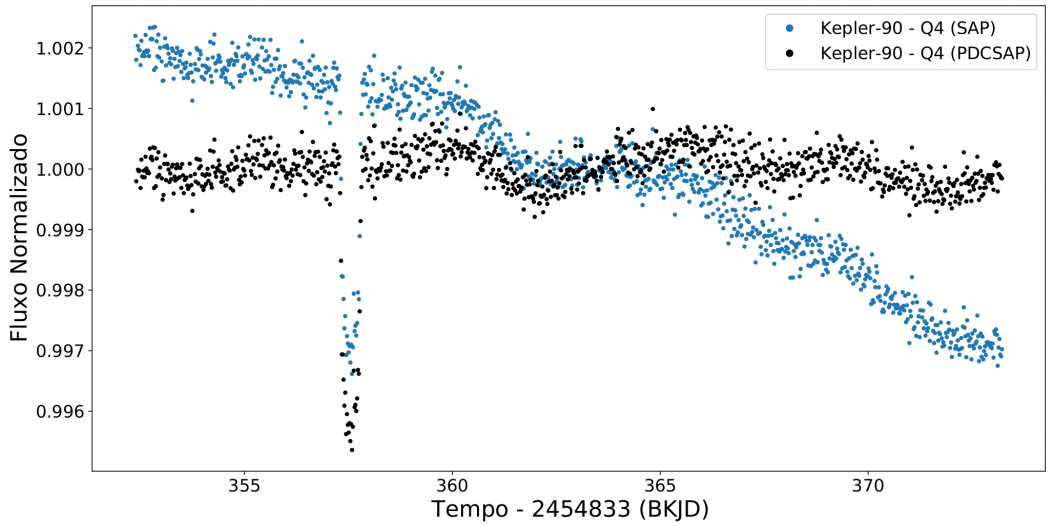
O tempo é exibido na escala BKJD (*Barycentric Kepler Julian Date*), que é uma unidade derivada da *Barycentric Julian Date* (BJD[11]) e é o referencial de tempo usado na contagem de dias julianos com correções da posição da Terra em relação ao baricentro do Sistema Solar, menos uma constante de 2.454.833 dias. Essa constante corresponde às 12h do dia 1 de janeiro de 2009 e foi definida assim para que o satélite usasse menos memória ao processar valores de tempo.

2.2 Tratamento Preliminar dos Dados

A curva de luz da figura 4 apresenta uma tendência decrescente gerada por erros sistemáticos cuja origem se dá por uma variedade de causas astrofísicas ou instrumentais (HIPPKE; ANGERHAUSEN, 2015; TWICKEN *et al.*, 2010). Um método comum para tratar isso, dos vários que existem na literatura, é usando o filtro de Savitzky-Golay (SAVITZKY; GOLAY, 1964), que remove essas tendências indesejadas e preserva características importantes da curva que serão discutidos na próxima seção. Esse filtro é usado inclusive no próprio pipeline do Kepler, no componente chamado *Presearch Data Conditioning* (PDC) (TWICKEN *et al.*, 2010). O fluxo tratado pelo pipeline recebe a especificação PDCSAP e pode ser visto na figura 5.

[11] Mais informações sobre BJD aqui.

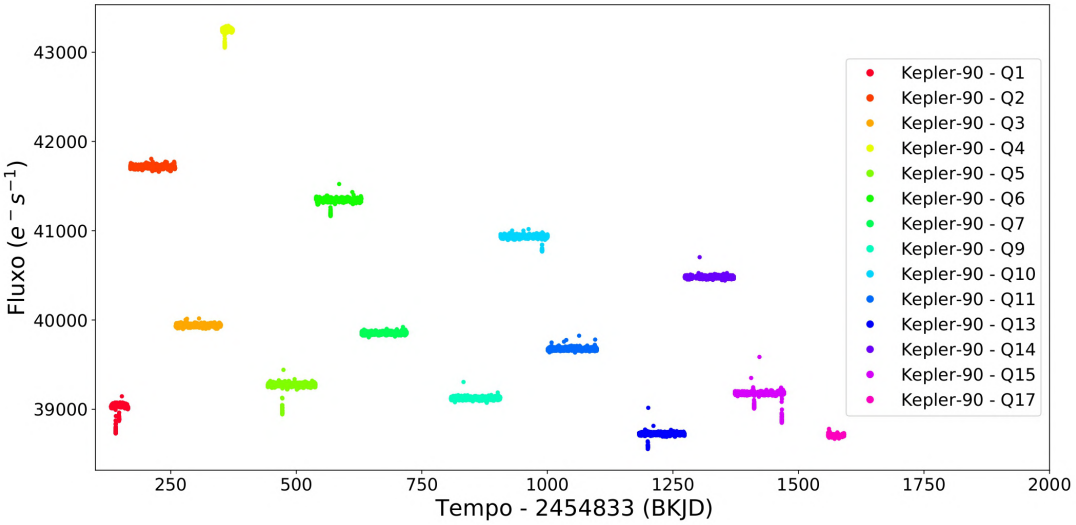
Figura 5 – Comparação entre curvas geradas por SAP e PDCSAP. Ambas foram normalizadas para serem vistas na mesma escala.



Fonte: Elaboração própria.

Todo esse tratamento foi feito em apenas um dos quarters da estrela. Repetindo esse processo de filtragem para todos os seus quarters e os inserindo em um mesmo gráfico, obtemos o resultado da figura 6. O motivo de cada quarter estar em uma escala diferente se dá por conta do giro de aproximadamente 90° que o Kepler dava para manter seus painéis solares em direção ao Sol, quando necessário, sem alterar seu campo de visão. Isso fazia com que a estrela passasse a ser observada em uma outra área do plano focal do telescópio, e a variação da magnitude do fluxo de um quarter para outro se dá porque, em geral, cada área desse plano tinha uma sensibilidade fotométrica diferente das demais (CLEVE; CALDWELL, 2016).

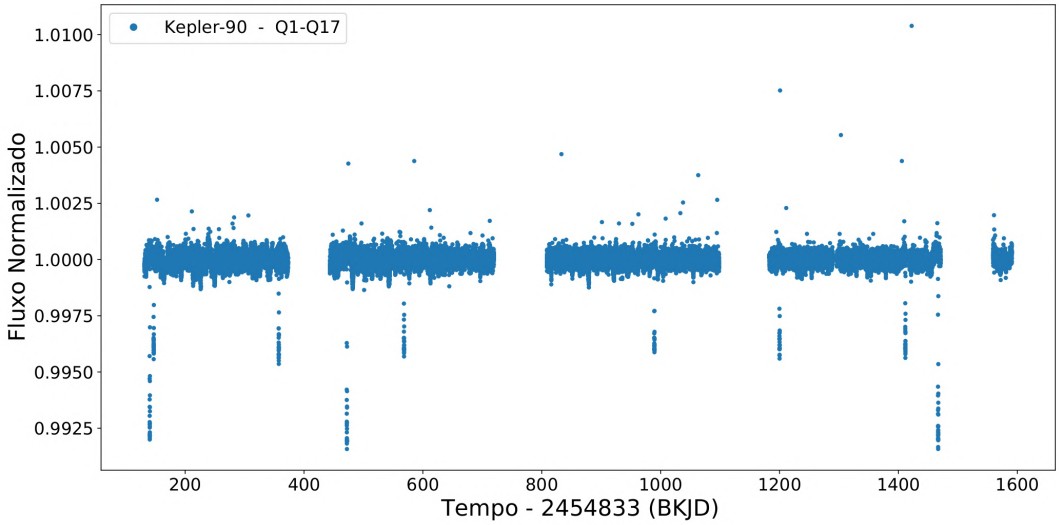
Figura 6 – Todos os quarters da estrela Kepler-90 filtrados individualmente.



Fonte: Elaboração própria.

Esse problema pode ser resolvido fazendo uma normalização no fluxo de cada quarter individualmente. A figura 7 mostra a curva normalizada.

Figura 7 – Curva de luz completa da estrela Kepler-90.



Fonte: Elaboração própria.

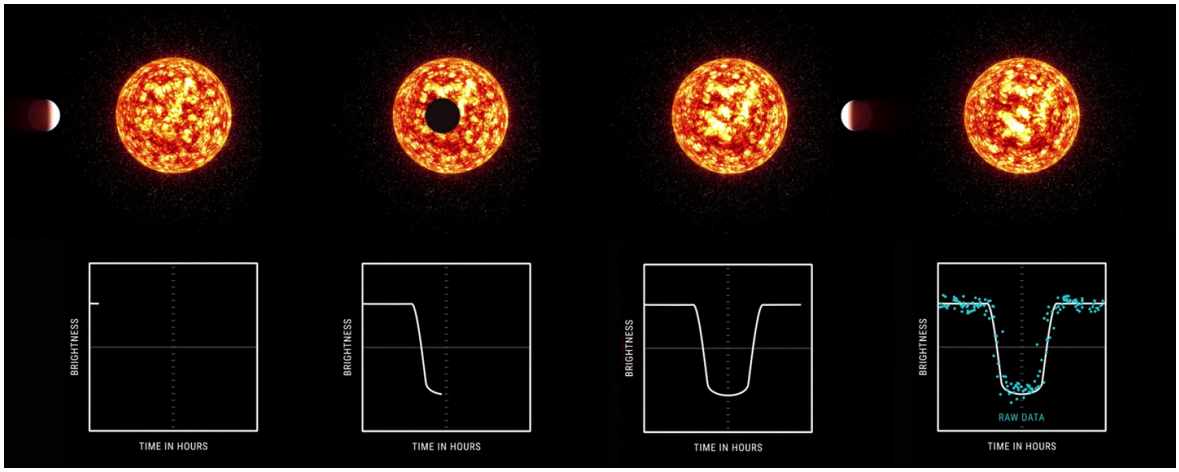
Essa curva de luz resume toda a observação da estrela Kepler-90 feita pelo telescópio. Algumas descontinuidades (ou gaps) na curva são observadas e possuem várias causas. Um dos tipos mais comuns de gap diz respeito ao download mensal dos dados, onde o telescópio muda sua orientação para apontar sua *High Gain Antenna* (HGA) para a Terra. Isso dura algo em torno de um dia e o telescópio não coleta dados nesse período. Variações de temperatura nos componentes do telescópio também são causadoras gaps (CLEVE; CALDWELL, 2016). A falta de medidas naturalmente também é fonte de gaps. Dados com baixa precisão fotométrica são descartados pelo pipeline do Kepler e substituídos por NaNs (CLEVE; CALDWELL, 2016; PASCUAL-GRANADO *et al.*, 2015). Há um espaço no arquivo FITS dedicado exclusivamente para as especificações de qualidade das medidas.

Pontos distantes do valor 1 no eixo das ordenadas podem caracterizar outliers, que são valores que se desviam notavelmente dos demais (GRUBBS, 1969). Nesta etapa, faríamos um tratamento estatístico para removermos esses dados, mas antes devemos levar em consideração mais um fator astrofísico que viabiliza todo este trabalho: o método usado para a detecção de exoplanetas.

2.3 Detecção de Exoplanetas Pelo Método de Trânsito Planetário

O método de trânsito planetário (BORUCKI; SUMMERS, 1984) foi o método de detecção de exoplanetas que mais se destacou ao longo dos anos, como mostra o gráfico da figura 40. Ele basicamente consiste em observar uma queda no fluxo de luz de uma estrela e avaliar se essa queda se deu devido ao trânsito de um planeta[12]. Quando o Kepler, por exemplo, observa uma estrela e um planeta passa na frente do seu campo de visão, o fluxo de luz que chega ao plano focal do telescópio sofre uma queda, como mostra a figura 8.

Figura 8 – Exemplo de curva de luz com a presença de um trânsito planetário.



Fonte: NASA Video, disponível em https://youtu.be/BFi4HBUdWkk. Acesso em 1 de dezembro de 2021.

Como o eixo das abscissas nesse gráfico diz respeito ao tempo, a largura dessa queda no fluxo corresponde à *duração* do trânsito. Por esse ser um movimento de translação, cada ocorrência dessa queda corresponde a um ano do planeta em questão. Chamamos o intervalo entre essas ocorrências de *período*. Olhando mais uma vez para a figura 7, vemos facilmente certas quedas bem acentuadas no fluxo. No entanto, alguma são maiores do que outras. Isso porque essas são assinaturas de dois planetas distintos, os Kepler-90g e Kepler-90h[13]. A *profundidade* do trânsito está associada com o raio do planeta. Naturalmente, quanto maior for o planeta transitando entre sua estrela e o telescópio, maior será a queda no fluxo. Uma última característica importante que vai nos interessar posteriormente é o que se chama na literatura de *epoch*, que é uma referência temporal que diz a primeira vez em que um dado trânsito foi observado na curva de luz. Por motivos de conveniência posterior, farei uso de t_0 em vez de

[12] Esse método também observa a variação na cor do fluxo estelar para a análise espectroscópica da atmosfera do exoplaneta.

[13] A exemplo do Sol, uma estrela pode ter vários planetas a orbitando. Uma visualização animada disso pode ser encontrada aqui.

“epoch". Disso, o t_0 do trânsito de maior profundidade na figura 7, por exemplo, é algo em torno de 140 BKJD. Todas essas propriedades são usadas para caracterizar um trânsito planetário. Um sinal periódico detectado pelo pipeline do Kepler na curva de luz, como quedas que possuem características de trânsito, são denominadas por *Threshold Crossing Event* (TCE) (BATALHA *et al.*, 2010).

Esse é um método poderoso e conceitualmente simples para a detecção de exoplanetas. Contudo, nem todo TCE na curva de luz é sinal de um exoplaneta. De fato, o *modus operandi* para a validação de exoplanetas desta forma é avaliar todas as demais possibilidades astrofísicas e instrumentais que possam explicar tal comportamento no fluxo. Se nada, senão um exoplaneta, puder explicar tal TCE, então a validação é viabilizada.

Com esse embasamento, é possível então identificar TCEs em curvas de luz. Isso se dá ao procurar por sinais periódicos nos dados por meio de algum método matemático-computacional e caracterizar as demais propriedades do trânsito, como explicado no apêndice A. No caso do gráfico da figura 7, uma identificação qualitativa é imediata a olho nu para dois dos oito planetas que orbitam a Kepler-90. O próximo passo do trabalho foi treinar um algoritmo de machine learning a reconhecer quais TCEs são, de fato, trânsitos planetários com base em análises feitas previamente pela comunidade astrofísica.

3 METODOLOGIA

Como mencionado anteriormente, os dados usados neste trabalho vêm do *Autovetter Planet Candidate Catalog for Q1-Q17 DR24* (CATANZARITE, 2015). Esse catálogo foi gerado pelo ***autovetter*** (MCCAULIFF *et al.*, 2015), que é um algoritmo de machine learning para a classificação automática de TCEs. O catálogo tem uma relação direta, mas independente, com o produzido pelo ***robovetter*** (COUGHLIN *et al.*, 2015), que também é um algoritmo para a classificação automática de TCEs, mas que não usa machine learning. Ambos os algoritmos fazem uso dos dados tratados pelo pipeline do Kepler para gerar seus produtos. Mais precisamente, o autovetter classifica TCEs com labels descritos por:

Tabela 1 – Tipos de labels da classificação de TCEs pelo autovetter.

Label	Descrição
Planet Candidate (PC)	Contém sinais que são consistentes com trânsitos planetários e que não há razões para descartar a hipótese de se tratar de um planeta.
Astrophysical False Positive (AFP)	Contém sinais de origem astrofísica que podem imitar trânsitos planetários, como eclipses binários, estrelas pulsantes, manchas solares e outros sinais periódicos que se configurem como fortes evidências para descartar um a hipótese de um trânsito devido a um planeta.
Non-Transiting Phenomenon (NTP)	Contém sinais que se dão evidentemente por origens instrumentais ou de ruído.
Unknown (UNK)	Se refere a TCEs não incluídos no conjunto de treino do algoritmo, mas que fazem parte do resto do conjunto de dados usado.

Fonte: Autovetter (MCCAULIFF *et al.*, 2015; CATANZARITE, 2015). Texto adaptado.

Existem outros trabalhos na literatura que se utilizam de inteligência artificial para a classificação de TCEs, cada qual com as suas vantagens. No caso particular discutido acima, o autovetter utiliza um algoritmo de machine learning conhecido como Floresta Randômica ou *Random Forest* (BREIMAN, 2001), que é famoso por ser rápido de treinar e por produzir modelos de alta qualidade. Contudo, nosso interesse é reconhecer padrões em curvas de luz e existem algoritmos na literatura que vêm ganhando cada vez mais relevância nesse tipo de tarefa: as *Redes Neurais Artificiais*, ou simplesmente *Redes Neurais*.

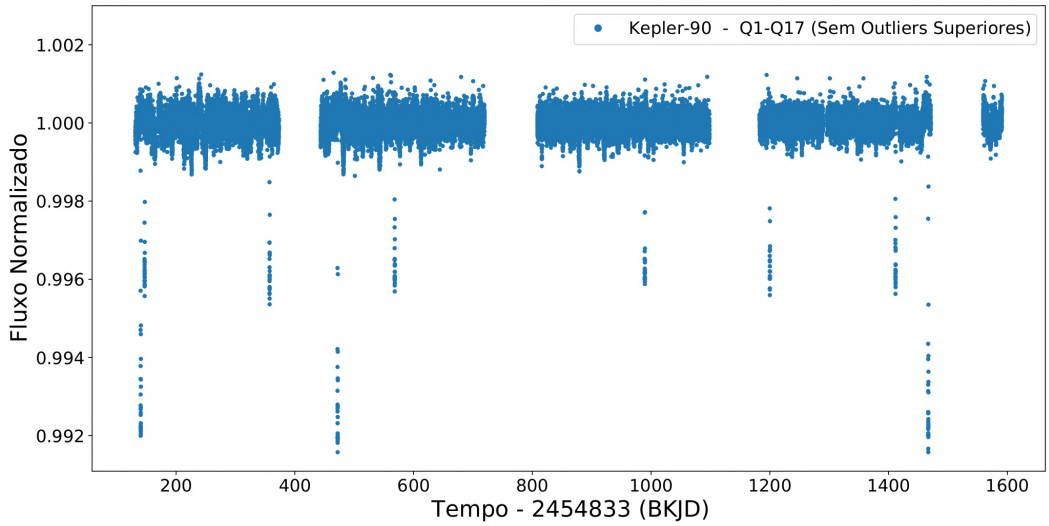
Em suma, uma rede neural artificial é um modelo de machine learning inspirado no funcionamento das redes neurais biológicas dos nossos cérebros, onde a atividade nervosa pode ser tratada por meio de conceitos lógico-matemáticos (MCCULLOCH; PITTS, 1943). Isso

abre caminho para uma subárea do machine learning que hoje é conhecida como aprendizado profundo, ou *Deep Learning*, (SCHMIDHUBER, 2014). Esses conceitos serão melhor abordados ainda neste capítulo, mas vale dizer que o deep learning é muito bem sucedido em tarefas como classificação de imagens e reconhecimento de padrões complexos. Essa é a motivação central do trabalho de (SHALLUE; VANDERBURG, 2018), cuja metodologia essencial será revisada aqui com detalhes para a implementação do algoritmo criado neste trabalho.

3.1 Tratamento dos Dados

Independente do modelo usado na construção de um algoritmo de machine learning, é imprescindível que os dados usados no treino estejam devidamente tratados. Para isso, toda a metodologia de tratamento de dados apresentada no capítulo 2 foi seguida, mas alguns passos adicionais foram tomados. Por conta do que foi discutido na seção 2.3, apenas os outliers superiores da curva de luz foram removidos, de forma a preservar os trânsitos. O valor escolhido para essa remoção foi de 3σ e um resultado de exemplo pode ser visto na figura 9.

Figura 9 – Curva de luz da estrela Kepler-697 sem outliers.



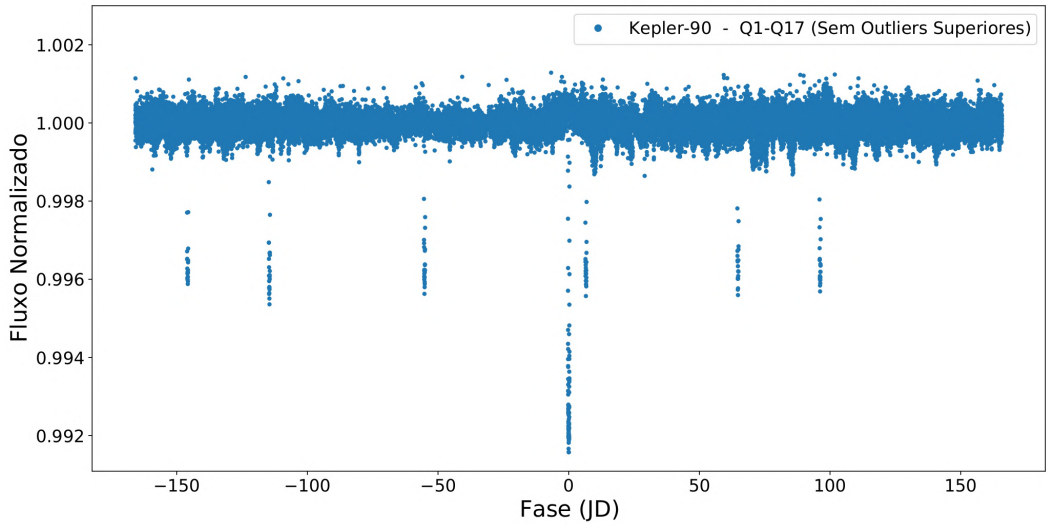
Fonte: Elaboração própria.

O que queremos analisar é a queda do fluxo de luz recebido pelo telescópio, logo todos os detalhes desse evento são importantes. Na seção 2.3 discutimos propriedades do trânsito como duração e profundidade, características essas extraídas do *formato* da curva resultante da queda. O formato dessa curva é tão importante que por ele podemos determinar outros parâmetros planetários e estelares do sistema em questão, como o raio do planeta, parâmetro de impacto,

escurecimento do limbo da estrela[14], dentre outros (MANDEL; AGOL, 2002; SOUTHWORTH, 2008; SOUTHWORTH, 2009; SING, 2010). É imprescindível então que o algoritmo faça uma análise minuciosa dele.

Poderíamos “cortar" a curva de luz, de forma a focar em uma das quedas no fluxo ocasionada por um dado TCE. Devido à cadência das medidas, uma única queda dessas geralmente não nos mostra com precisão o formato do trânsito por conta da relativa escassez de observações ao longo do evento, especialmente se a sua duração for curta. Diante disso, podemos usar uma técnica muito comum na análise de curvas de luz chamada de *dobramento de fase* (phase folding) (GREGORY; LOREDO, 1992). Arbitrariamente falando, ela consiste em dividir a curva de luz em ciclos definidos por um dado período e então sobrepor esses ciclos, centralizando-os em um referencial t_0. Ao fazer isso, passamos a ter mais informações acerca do formato do trânsito e o fluxo agora se dá em função da fase em vez do tempo. Tomando o exemplo do trânsito do planeta Kepler-90 h, tendo $t_0 = 140,480$ BKJD e seu período $p = 331,603$ BKJD[15], o resultado desse processo se dá pelo gráfico da figura 10.

Figura 10 – Diagrama de fase da estrela Kepler-90 para o trânsito do planeta Kepler-90 h.



Fonte: Elaboração própria.

O intervalo de valores da fase no eixo das abscissas é igual ao período do sinal (de $-p/2$ até $p/2$) e se dá em *dias julianos* (JD). Comparando os gráficos das figuras 9 e 10, é possível ver um maior número de pontos descrevendo o trânsito central no segundo gráfico do que em qualquer uma das quedas individuais no fluxo por esse TCE no primeiro gráfico.

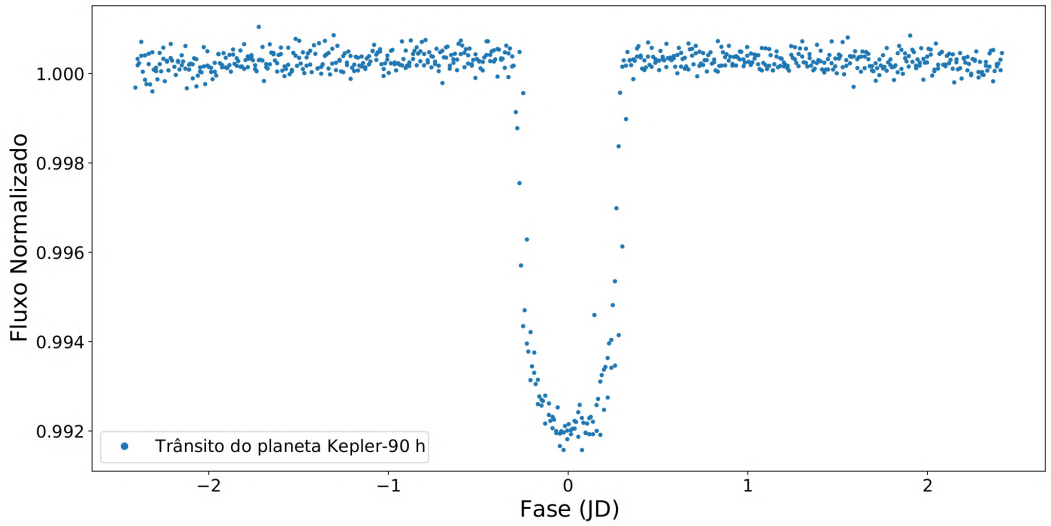
[14] Do latim, “limbus"significa borda. O escurecimento do limbo é uma característica observada em estrelas onde o centro do disco observado parece mais claro que sua borda. Isso se dá essencialmente por fenômenos termodinâmicos e eletromagnéticos na estrela.

[15] Mais informações sobre esse planeta aqui

Contudo, devemos olhar mais de perto esse evento para realmente analisá-lo. Podemos selecionar apenas os dados do fluxo e do tempo que estão nos arredores do trânsito de interesse, como mostra gráfico da figura 11.

Temos então uma visão localizada do trânsito, que será denominada de visão local. Para manter um padrão quando esse mesmo procedimento for feito para as demais curvas de luz, todas as visões locais têm seus TCEs localizados aproximadamente no centro do gráfico e um intervalo igual a k durações do trânsito para esquerda e direita dos mesmos. A escolha desse valor é totalmente arbitrária e a convenção neste trabalho é $k = 4$. Ainda seguindo a ideia de padronizar todos os dados, um problema que surge ao ampliar o trânsito dessa forma é a quantidade de pontos que cada curva de luz terá. Trânsitos mais periódicos, por exemplo, terão mais pontos constituindo o seu gráfico no diagrama de fase.

Figura 11 – Trânsito detalhado do planeta Kepler-90 h.



Fonte: Elaboração própria.

Isso pode ser resolvido usando uma técnica chamada de *binning de dados* (data binning). Nela, basicamente fazemos um histograma dos dados. Mais precisamente, definimos vários intervalos de largura δ no eixo do tempo (ou no da fase, neste caso), separados por uma distância λ, e escolhemos como vamos categorizar os dados do fluxo que estejam contidos em cada um desses intervalos. Escolhemos representá-los pela mediana dos valores nesse intervalo. Os valores desses dois parâmetros também são arbitrários até certo ponto, mas foi observado que alguns trânsitos ficavam mais visíveis fazendo $\delta > \lambda$, onde os intervalos se sobrepõem.

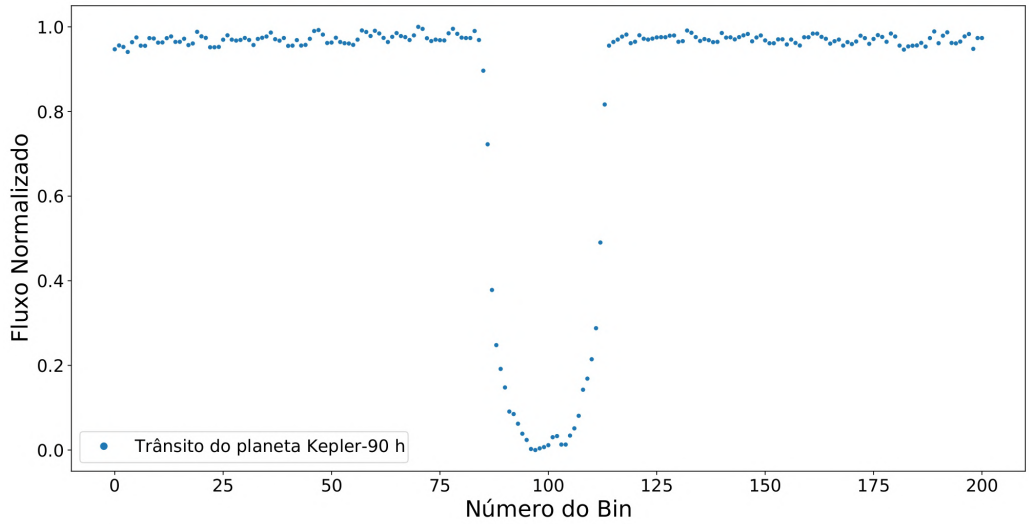
Ao fazer esse procedimento, um outro parâmetro que surge naturalmente é a quantidade de *bins* que a nova curva de luz terá, ou seja, a quantidade de pontos que vai descrever

a curva de luz. Para manter um padrão geral para as curvas de luz, estabelecemos que todas as curvas de luz locais terão exatamente 201 pontos, ou bins. Esse número ímpar se dá para que a curva tenha um ponto central onde o trânsito possa idealmente ser posicionado de forma simétrica. Como foi estabelecido um número fixo de bins, a relação que nos dá os valores dos parâmetros é

$$\lambda = \frac{\delta}{2} = \frac{\Delta d}{N_b}, \tag{3.1}$$

em que N_b é o número de bins e Δd diz respeito à quantidade de dados (pontos) da curva original que a descreve no intervalo de tempo considerado. Por fim, como estamos interessados apenas no formato do trânsito, os valores do fluxo foram normalizados entre 0 e 1 a fim de evitar que o a sua profundidade ganhe demasiada influência na caracterização dos planetas[16]. A curva local final para o exemplo do planeta Kepler-90 h é dada pela figura 12.

Figura 12 – Visão local do trânsito do Kepler-90 h.



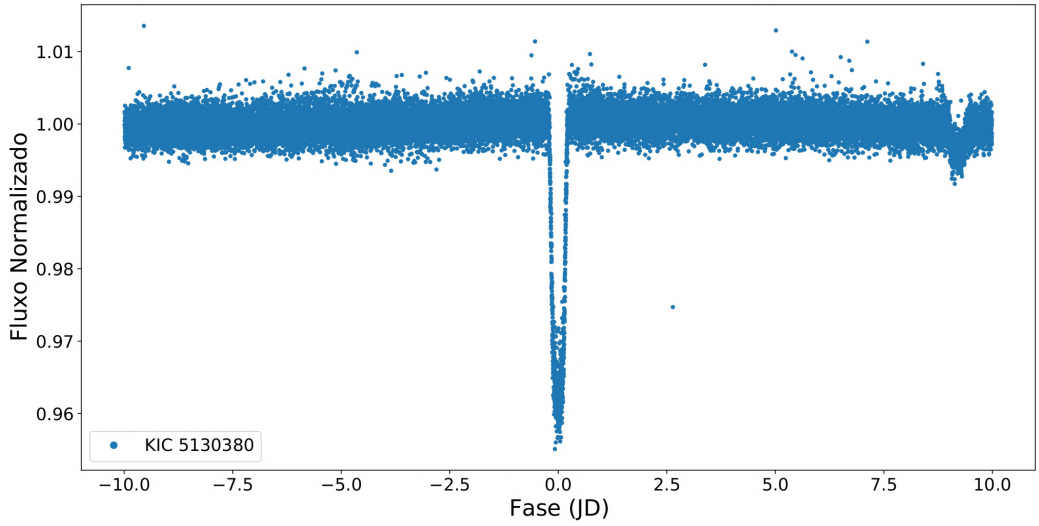
Fonte: Elaboração própria.

Ao término desse processo, todas as informações do trânsito passam a ser descritas por um *array* unidimensional. Isso é particularmente interessante do ponto de vista do machine learning, pois com o binning teremos menos ruídos nos dados, além de modelos mais eficientes e mais rápidos de serem treinados (KE *et al.*, 2017). Contudo, estaremos perdendo informações valiosas se considerarmos apenas a visão local de um TCE. Fazendo todo o processo de tratamento

[16] A profundidade do trânsito pode sim fornecer informações importantes acerca das características de um planeta, mas não tem uma relação direta com o a detecção do mesmo. Olhando de fora do sistema solar, Júpiter causaria uma grande queda no fluxo de luz do Sol, mas não é por isso que Mercúrio deixaria de ser um planeta ou vice-versa.

de dados na curva de luz da estrela KIC 5130380 e escolhendo o primeiro dos seus dois TCEs na tabela do catálogo, geramos o diagrama de fase da figura 13.

Figura 13 – Diagrama de fase da estrela KIC 5130380 para um de seus TCEs.



Fonte: Elaboração própria.

Se dobramos a curva de luz com base no período de um TCE específico, esperamos que todo sinal, além do de interesse, que tiver seu mesmo período se some em algum lugar fixo na curva. É justamente isso que acontece neste caso, pois vemos duas quedas bem definidas nesse diagrama. A queda do fluxo centralizada é correspondente ao TCE escolhido para análise, mas a outra queda diz respeito ao segundo TCE dessa estrela[17], que vemos ter o mesmo período que o TCE em destaque afora uma diferença de fase. Isso nos dá um indício de que esse sistema se trata de uma estrela binária[18] (PRŠA; ZWITTER, 2005) e não necessariamente de um planeta em trânsito. De fato, vemos pela figura 10 que um trânsito com o período diferente (o do planeta Kepler-90 g) do que estamos analisando se distribui de forma não uniforme no gráfico.

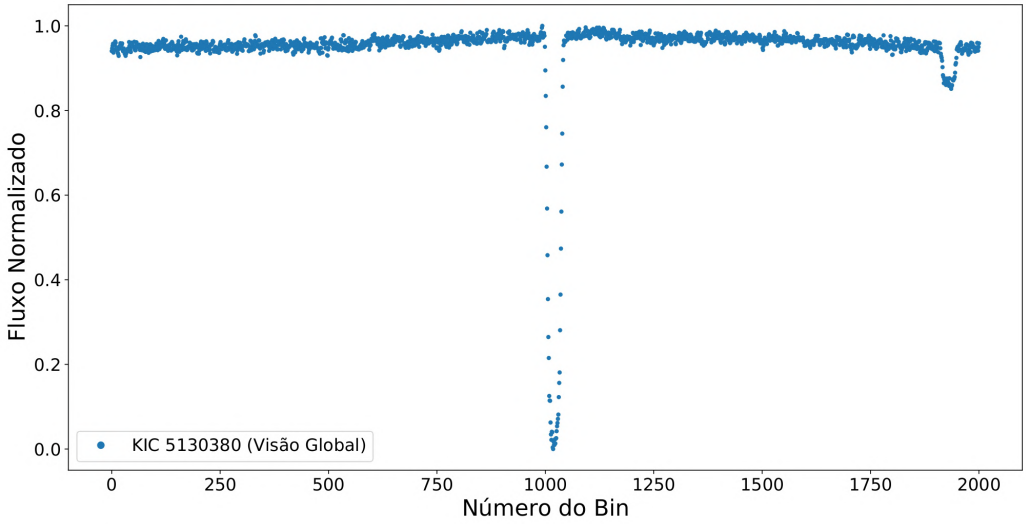
Dessa forma, também é importante levar em consideração o que acontece em todo o diagrama de fase. Seguimos um procedimento semelhante ao das curvas locais no que diz respeito ao binning de dados. Como agora o Δd é maior, é natural pensar em representar a curva por um número de bins maior para evitar a perda de informações importantes como as da figura 13. Chamamos essas curvas maiores de curvas globais e definimos um número fixo de bins igual

[17] Neste caso, a designação dos TCEs se dá como KIC 5130380[1] e KIC 5130380[2] para os primeiro e segundo TCEs, respectivamente.

[18] Uma estrela binária é um sistema formado por duas estrelas que se encontram próximas uma da outra. Elas orbitam o centro de massa do sistema e por conta dessa proximidade, a luz de uma pode ser “escondida"pela outra para um observador que se encontra longe do sistema quando as estrelas se alinham. Além de uma queda secundária, uma outra característica muito recorrente é o formato de “V" que a curva assume, com ou sem trânsito secundário. Mais informações aqui.

a 2001. A equação 3.1 também vale neste caso. O resultado desse processo pode ser visto no gráfico da figura 14.

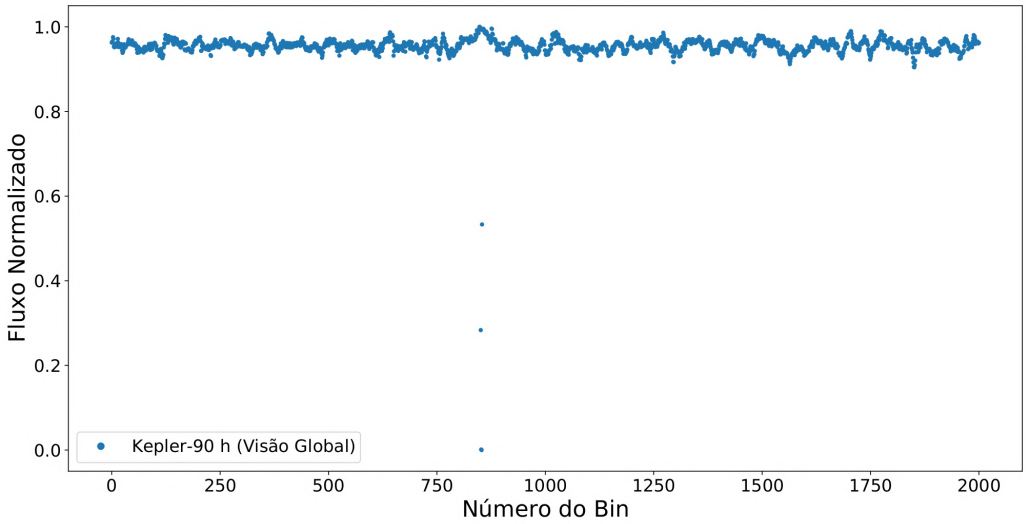
Figura 14 – Curva de luz global do KIC 5130380[1]



Fonte: Elaboração própria.

Um motivo para não usarmos apenas as curvas globais é que informações mais sutis são perdidas no processo de binning de dados em alguns casos. Mais especificamente, trânsitos muito curtos podem acabar ficando completamente inseridos na largura δ, de forma que seu formato seja parcial ou completamente perdido em uma visão global, como mostra a curva global do próprio Kepler-90 h na figura 15.

Figura 15 – Curva de luz global do Kepler-90 h



Fonte: Elaboração própria.

O processo de binagem retirou toda a influência significante que o trânsito do Kepler-

90 g poderia ter na curva, mas também o fez para o próprio Kepler-90 h. É difícil esperar que o algoritmo entenda que há um planeta nessa curva global, mas não há dúvidas ao olhar para sua representação local. Em casos como os da figura 14, é esperado que a curva global tenha mais serventia em informar que o TCE não é um planeta do que o contrário. Com isso, o tratamento de dados está concluído. Para ilustrar, a figura 42 mostra algumas curvas globais já tratadas e escolhidas aleatoriamente do dataset e a figura 43 mostra suas contrapartes locais.

3.2 A Construção do Algoritmo

O processo de construção do algoritmo consiste em fazer a implementação de vários modelos de machine learning e avaliar qual deles se sai melhor no reconhecimento de exoplanetas. Uma vez escolhido o mais promissor, ele será trabalhado com mais profundidade para um ajuste de *hiperparâmetros*, de forma a entregar resultados ainda melhores. Hiperparâmetros são parâmetros que controlam vários aspectos do processo de aprendizagem de um modelo e afetam diretamente sua performance (CLAESEN; MOOR, 2015). Em geral, temos liberdade para escolher arbitrariamente os seus valores.

A padronização das curvas de luz é muito importante, pois ela garante que todos os modelos testados trabalhem com o mesmo número de *features* para cada *instância* ou *amostra* analisada. Os termos “instância" e “amostra" são sinônimos e se referem a cada elemento do conjunto de dados, que no caso deste trabalho são os pares das curvas de luz (global e local) dos TCEs tratados. “Features", por sua vez, são as características de uma instância. Neste caso, os features são os bins e seus respectivos valores para o fluxo. Padronizar as curvas de luz também garantiu bastante otimização computacional, uma vez que o volume de dados para lidar passou de 90 Gb para pouco mais de 950 Mb.

Um total de 15.517 pares de curvas foram gerados por esse processo e em seguida eles foram divididos aleatoriamente em três grupos: um conjunto de treino, um conjunto de validação e um conjunto de teste. O conjunto de treino e o de teste, como os nomes sugerem, são usados para treinar e testar os modelos, respectivamente. O conjunto de validação serve para nos ajudar a ajustar os hiperparâmetros, como veremos em breve. A divisão foi feita alocando 80% dos dados para o conjunto de treino, 10% para o de validação o 10% para o de teste. Vale ressaltar que o algoritmo deve se sair bem ao avaliar TCEs nunca vistos antes. Para ter uma boa noção do quão bem ele classifica novos TCEs como planetas ou não, os dados de teste ficaram completamente inacessíveis ao longo do processo de treino e validação dos modelos só foram

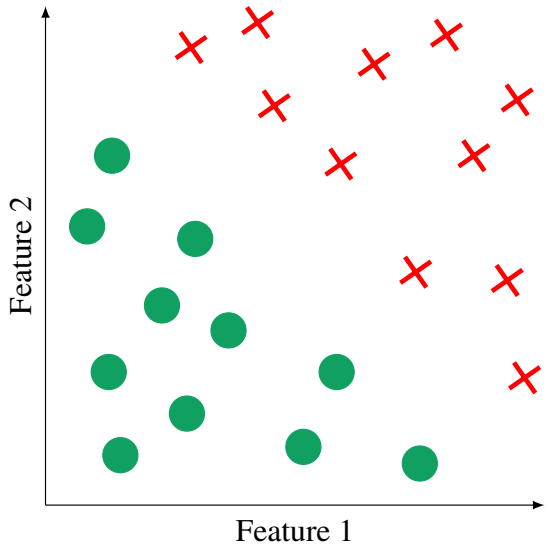
usados na fase de teste propriamente dita.

Os algoritmos testados foram os de **Regressão Logísitca**, **Gradiente Descendente Estocástico**, **Floresta Randômica** e **Rede Neural Artificial**. Os resultados encontrados são discutidos no capítulo 4. O restante deste capítulo focará em breves abordagens sobre conceitos teóricos dos modelos testados, baseadas principalmente em (GÉRON, 2019).

3.3 Regressão Logística

O processo de treino do algoritmo consiste essencialmente em determinar uma *hipótese* que melhor descreva os dados de interesse. Como um exemplo básico dessa aplicação, podemos supor que os TCEs sejam descritos por dois features genéricos, de forma que eles possam ser representados em um espaço $\mathbb{R}^2$ tal como na figura 16.

Figura 16 – Exemplo de representação de um espaço de features bidimensional.



Fonte: Elaboração própria.

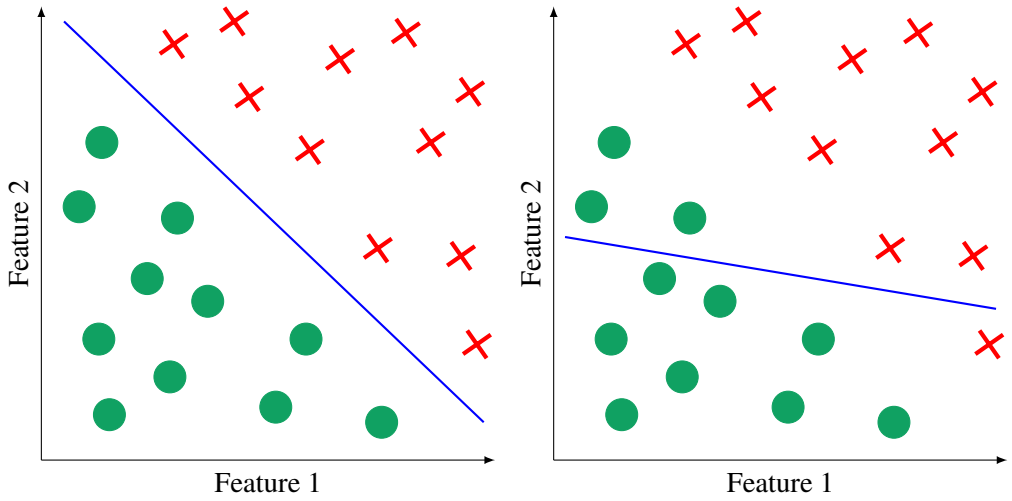
Os TCEs que se tratam de planetas são representados pelos símbolos verdes; os que não, pelos vermelhos. Naturalmente, essa é uma classificação binária, que foi o caso considerado neste trabalho. TCEs com labels AFP e NTP foram todos tratados como não-planetas e aqueles com o label UKN foram descartados. Apesar de alguns poucos erros observados por inspeção manual de alguns TCEs, todos os labels foram tratados como verdade absoluta.

Neste caso do exemplo, a hipótese vai levar em consideração uma função linear que separa as duas classes em regiões bem definidas no espaço de features. Essa função é chamada de *fronteira de decisão* e é dada em termos dos features. A figura 17 mostra dois exemplos de

fronteiras que o algoritmo pode determinar.

Figura 17 – Exemplos de fronteiras de decisão para classificar os TCEs.

(17.a) Fronteira razoável. **(17.b)** Fronteira ruim.



Fonte: Elaboração própria.

Os TCEs representados nas figuras fazem parte do conjunto de treino. Quando o algoritmo aprende com os dados de treino e estabelece uma fronteira de decisão, ele pode classificar novos TCEs ao avaliar em qual região do espaço de features aquele novo dado pertence. No caso específico deste exemplo, essa fronteira é dada por

$$x = \theta_0 + \theta_1 x_1 + \theta_2 x_2, \tag{3.2}$$

em que x_1 e x_2 são os dois features genéricos considerados neste exemplo, os parâmetros θ_1 e θ_2 são os pesos atribuídos aos features para que a fronteira de decisão se ajuste aos dados e o parâmetro θ_0 é uma constante, chamada de termo de viés, também usado no ajuste da fronteira de decisão.

Existem situações em que uma fronteira linear não é capaz de dividir bem os dados, como mostrado na figura 44. A equação 3.2 levaria em conta termos não lineares para essas situações. No nosso caso, o espaço de features para as curvas locais têm 201 dimensões, ao passo que o das globais têm 2001 dimensões. Desta forma, a fronteira de decisão é tal que separa os respectivos hiperespaços em regiões apropriadas para a classificação dos dados. A equação 3.2 pode ser generalizada para descrever dados com mais features e escrita em uma forma matricial como

$$\boldsymbol{X}^{(i)} = \boldsymbol{\theta}^T \boldsymbol{x}^{(i)}, \quad \text{onde} \quad \boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}, \quad \boldsymbol{x}^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \quad \text{e} \quad x_0^{(i)} = 1. \tag{3.3}$$

Os índices i indicam qual das m amostras estamos considerando. Como o termo $x_0^{(i)}$ foi inserido por motivos de coerência da multiplicação matricial, o espaço de features agora é $\mathbb{R}^{n+1}$. Esse termo sempre é igual a 1 para qualquer amostra do conjunto de dados.

Grosso modo, o aprendizado do algoritmo se dá quando o mesmo encontra os valores ótimos de $\boldsymbol{\theta}$ que definam uma fronteira de decisão que descreva bem os dados de treino e que consiga generalizar os resultados para novos dados. Dada uma fronteira de decisão, definimos a nossa hipótese como

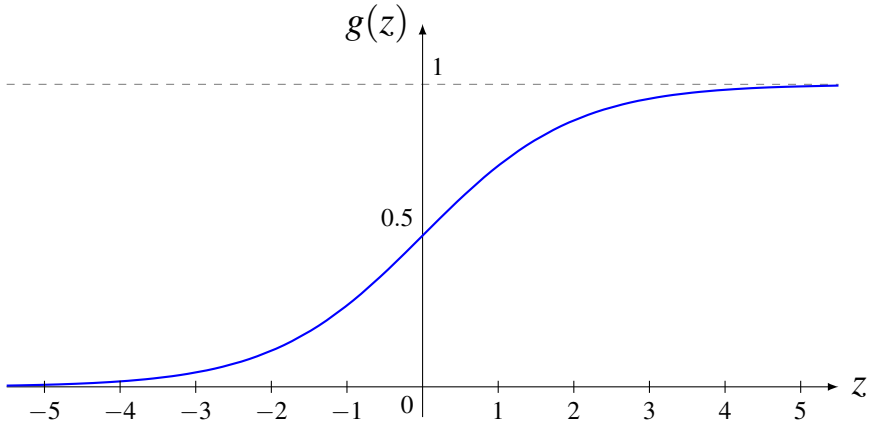
$$h_\theta(\boldsymbol{x}^{(i)}) = g\left(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)}\right). \tag{3.4}$$

A função $g(z)$ é não-linear e seu papel é avaliar se uma dada instância $\boldsymbol{x}^{(i)}$ pertence a uma classe ou não. Uma escolha comum para ela é a função conhecida como *função logística* ou *função sigmoidal*, que é dada por

$$g(z) = \frac{1}{1+e^{-z}}. \tag{3.5}$$

A figura 18 mostra seu gráfico. Os possíveis valores de $g(z)$ estão no intervalo entre 0 e 1.

Figura 18 – Gráfico da função logística.



Fonte: Elaboração própria.

Quando $z > 0$, $g(z)$ é maior que 0.5 e o algoritmo considera que aquele dado seja um positivo[19]. Fazendo $z = \boldsymbol{\theta}^T \boldsymbol{x}^{(i)}$, temos então que

$$h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) = \frac{1}{1+e^{-\boldsymbol{\theta}^T \boldsymbol{x}^{(i)}}}. \tag{3.6}$$

Dessa forma, podemos entender que a hipótese nos retorna a probabilidade de um TCE ser um planeta, dados os seus features $\boldsymbol{x}$, parametrizados por $\boldsymbol{\theta}$. Isso é matematicamente equivalente a

$$h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) = P(y = 1 | \boldsymbol{x}^{(i)}; \boldsymbol{\theta}). \tag{3.7}$$

Neste caso, y é a classificação que o modelo dá ao TCE. Se for planeta, então se estabelece que $y = 1$; caso contrário, $y = 0$. Naturalmente, a soma das probabilidades de um TCE ser planeta e de não ser planeta é 100%, ou seja,

$$P(y = 0 | \boldsymbol{x}^{(i)}; \boldsymbol{\theta}) + P(y = 1 | \boldsymbol{x}^{(i)}; \boldsymbol{\theta}) = 1. \tag{3.8}$$

Para que o algoritmo escolha os parâmetros $\boldsymbol{\theta}$, ele deve considerar o quão certa está a hipótese. A partir das equações acima, é possível encontrar uma função que mede o erro que a hipótese comete em relação ao dado que ela deveria prever. Chamamos essa função de **função custo** e ela é dada por

$$J(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\left[\boldsymbol{y}^{(i)} \log\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)\right) + (1-\boldsymbol{y}^{(i)}) \log\left(1 - h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)\right)\right] \tag{3.9}$$

Os melhores valores de $\boldsymbol{\theta}$ se dão quando esse erro é o menor possível. O algoritmo empregado para minimizar essa função custo é chamado de **gradiente descendente**. Seu funcionamento consiste em inicializar valores aleatórios para todos os $n+1$[20] parâmetros θ e interativamente ir somando ou subtraindo uma dada quantidade de todos eles para que o valor de $J(\boldsymbol{\theta})$ vá diminuindo ao longo das interações. Disso, o algoritmo do gradiente descendente se dá por repetir simultaneamente para todos os θ_j a operação

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) \quad \Rightarrow \quad \theta_j = \theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}\left[h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - \boldsymbol{y}^{(i)}\right]\boldsymbol{x}^{(i)}. \tag{3.10}$$

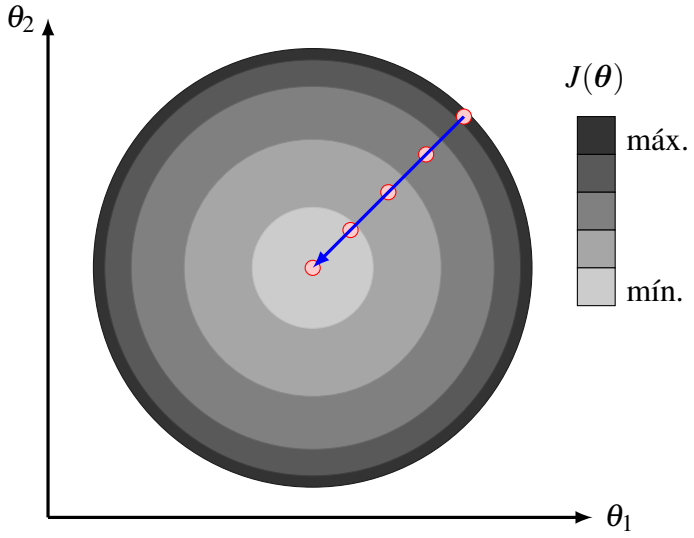
Isso é repetido até a função custo convergir para um valor mínimo. O (hiper)parâmetro α é chamado de taxa de aprendizagem e dita o quão rápido essa minimização ocorre. Seu valor

[19] Essa escolha é arbitrária, poderia ser o contrário.

[20] O n é o número de features no conjunto de treino. O "+1"se dá por conta do termo de viés θ_0.

afeta diretamente o desempenho do modelo, uma vez que valores muito baixos deixam o treino demorado e valores muito altos podem fazer com que o algoritmo divirja. No caso do exemplo dos dois features mostrado na figura 16, a representação do caminho que o gradiente descendente faz no espaço dos $\boldsymbol{\theta}$ até convergir em um valor mínimo se dá pela figura 19. Dado um ponto inicial aleatório nesse espaço, o gradiente seguirá o caminho que leva até um ponto onde a derivada de $J(\boldsymbol{\theta})$ é zero.

Figura 19 – Gradiente Descendente.



Fonte: Elaboração própria.

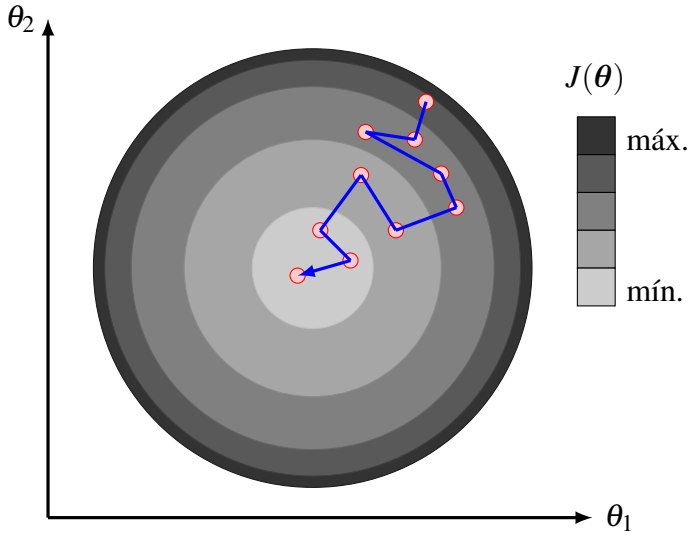
3.4 Gradiente Descendente Estocástico

Pela equação 3.10, vemos que o gradiente descendente usa todos os dados do conjunto de treino para convergir até um valor de custo mínimo. Em conjuntos onde o número de dados é muito grande, esse processamento pode levar muito tempo, especialmente se o número de features também for grande. O método de **Gradiente Descendente Estocástico** (GDE) é essencialmente o mesmo que o da regressão linear. A única diferença é que em vez de usar todas as m amostras do conjunto de treino, esse algoritmo seleciona apenas uma delas aleatoriamente e realiza o mesmo procedimento. Neste caso, podemos reescrever a equação 3.10 como (BOTTOU, 2012)

$$\theta_j = \theta_j - \alpha \left[h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - \boldsymbol{y}^{(i)} \right] \boldsymbol{x}^{(i)}, \tag{3.11}$$

onde consideramos apenas uma i-ésima amostra escolhida aleatoriamente. Isso reduz drasticamente o processamento computacional, mas torna o caminho de convergência tortuoso, como mostra a figura 20.

Figura 20 – Gradiente Descendente Estocástico.



Fonte: Elaboração própria.

O GDE também é bastante útil quando a função custo é irregular, pois seu caráter aleatório faz com que o algoritmo seja capaz de sair de mínimos locais no espaço dos $\boldsymbol{\theta}$. Essa mesma vantagem se torna uma desvantagem, já que o algoritmo não consegue se estabelecer em um mínimo global. Isso, contudo, pode ser remediado reduzindo-se gradualmente a taxa de aprendizagem. Ao final da execução do algoritmo, os parâmetros $\boldsymbol{\theta}$ são bons, mas geralmente não são ótimos.

3.5 Floresta Randômica

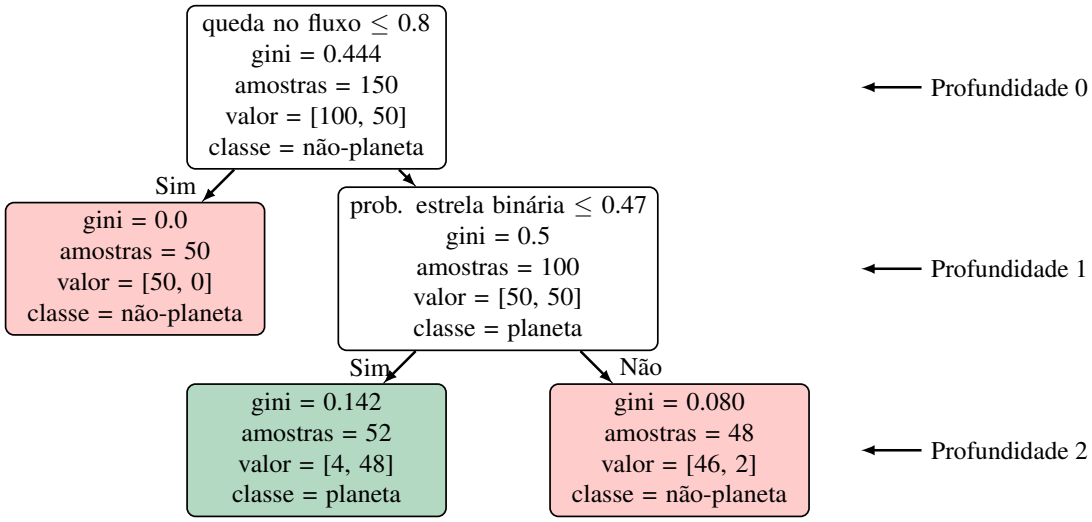
O algoritmo de **floresta randômica** se fundamenta em algo que é conhecido como *ensemble learning method* (ou apenas *ensemble method*), que é um algoritmo composto por um conjunto de algoritmos preditores com pouca ou nenhuma correlação entre si. O que ele normalmente faz é dividir o conjunto de treino em subconjuntos com um dado número de amostras para que cada algoritmo preditor seja treinado com um desses subconjuntos. Depois de treinados, quando o algoritmo de floresta randômica recebe um novo dado, cada um de seus preditores fará a sua classificação particular daquele dado e a classe mais votada será a considerada na classificação final.

O nome “floresta randômica" se dá porque os seus preditores são algoritmos conhecidos como *árvore de decisão* e porque as amostras dos subconjuntos criados no processo são selecionadas aleatoriamente. Para entender bem um algoritmo de floresta randômica, é útil conhecer o funcionamento de uma árvore de decisão e de um ensemble method genérico.

3.5.1 Árvore de Decisão

Mais uma vez, usaremos um exemplo fictício com base no nosso contexto para explorarmos os conceitos deste algoritmo. Neste exemplo, nossos TCEs serão simplificados de tal forma que serão descritos por apenas dois features x_1 e x_2. São eles, respectivamente: a presença de queda no fluxo e a probabilidade de que essa queda seja causada por uma estrela binária. Podemos considerar que x_1 e x_2 assumam valores reais entre 0 e 1, e que x_1 indique a porcentagem de relevância que a queda tem (se houver alguma) para caracterizar um planeta. Suponhamos também que temos 150 amostras de treino. A figura 21 dá um exemplo de uma árvore já treinada para estudarmos.

Figura 21 – Exemplo de árvore de decisão para classificação binária.



Fonte: Elaboração própria.

O processo de classificação que a árvore faz se baseia em perguntas que serão respondidas pelos features de um TCE que ela esteja analisando. Começamos com o nó na profundidade 0, chamado de *nó-raiz*. Ele pergunta se não há uma queda relevante no fluxo na curva de luz, ou seja, se x_1 é menor ou não de 80%. Descemos para a próxima profundidade da árvore e a resposta da pergunta feita pelo nó-raiz nos dirá para qual de seus *nós-filhos* iremos. Em caso positivo, então a curva não apresenta uma queda relevante no fluxo e chegamos no nó esquerdo da profundidade 1 da árvore. Esse nó em particular é chamado de *nó-folha* (ou *nó-terminal*), pois não tem nenhum nó-filho e não faz nenhuma pergunta. Quando chegamos em um nó-folha, a árvore avalia qual a classe predita nesse nó. Neste caso, o TCE seria classificado como não sendo planeta.

Se a resposta para o nó-raiz for negativa, então iremos para o nó direito da profundidade 1. Esse nó faz uma outra pergunta, onde ele avalia a se probabilidade da queda no fluxo ser dada por uma estrela binária é menor que 47% ou não. Vamos então para a profundidade 2 da árvore e em seguida para o nó-folha da esquerda se essa probabilidade for menor que 47%, fazendo com que o TCE seja classificado como um planeta. Caso contrário, vamos para o nó-folha da direita e o planeta é então classificado como não sendo um planeta.

Todos os nós carregam algumas informações, chamadas de *atributos*, acerca dos dados usados no treino do algoritmo. A `classe` é a predição mais provável caso a resposta da pergunta feita pelo nó seja positiva. No caso dos nós-folha, é a própria classificação que a árvore levará em conta. O atributo `amostras` diz para quantas amostras de treino aquele nó se aplica. Por exemplo, o nó raiz se aplica a todos os 150 dados do conjunto de treino, ao passo que o nó-folha da primeira profundidade se aplica a 50 dados desses 150, o nó-filho da direita se aplica a 100 desses 150 e assim por diante. O atributo `valor` informa quantas amostras de cada classe aquele nó se aplica. Por exemplo, o nó da esquerda na profundidade 2 se aplica para 4 não-planetas e 48 planetas. Por fim, o atributo `gini` é uma medida de impureza do nó[21]. Essa impureza é calculada como

$$G_i = 1 - \sum_{j=1}^{k} \left(\frac{a_i}{v_j} \right)^2, \tag{3.12}$$

onde G_i é o gini do i-ésimo nó, k é o número de classes, a_i é a quantidade de amostras para o qual o i-ésimo nó se aplica e v_j é o valor de quantas amostras da j-ésima classe aquele nó se aplica. Quando esse valor é igual a 0, dizemos que o nó é “puro". Isso acontece no caso da folha da profundidade 1, onde todas as amostras daquele nó pertencem à mesma classe (não-planeta). Em termos práticos, o cálculo do gini nesse nó se dá por $1 - (50/50)^2 - (0/50)^2 = 0$. No nó da esquerda na profundidade 2, o gini se dá por $1 - (4/52)^2 - (48/52)^2 \approx 0,142$.

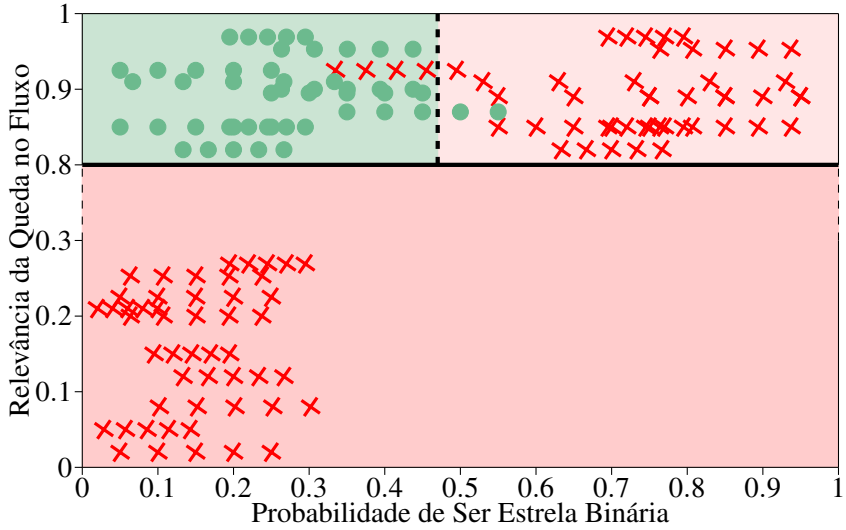
Todo o fluxo de decisão da árvore se baseia em regras “se-então", de forma que é possível acompanhar seu funcionamento manualmente, como acabamos de fazer. Apesar de simples, é um algoritmo bastante versátil e poderoso, sendo capaz de realizar outras tarefas além de classificação. Em particular, a equação 3.12 mostra que a árvore pode fazer classificação com mais de duas classes, como mostra o exemplo da figura 45.

Esse é o procedimento que uma árvore treinada realiza para fazer classificações. Resta agora saber como, de fato, a árvore é treinada. Um algoritmo bem conhecido, e usado

[21] Outra medida de impureza comumente usada em machine learning é a entropia. Em termos práticos, não há diferenças muito relevantes no resultado da árvore ao usar gini ou entropia.

no exemplo acima, é o *Classification and Regression Tree* (CART), que produz apenas árvores binárias. Isso significa que um nó que não é folha só possui dois nós-filhos, que respondem “sim" ou “não" para a sua pergunta. O CART começa separando o conjunto de treino em dois subconjuntos usando um feature x dos n disponíveis e uma condição t_x. Essa separação inicial é o que cria o nó-raiz e já traz a noção de fronteira de decisão discutida anteriormente. A figura 22 mostra a fronteira de decisão para a árvore da figura 21 e a figura 46 mostra o mesmo esquema, mas para a árvore da figura 45.

Figura 22 – Fronteiras de decisão geradas na classificação binária da árvore de decisão do exemplo.



Fonte: Elaboração própria.

No exemplo da figura 21, o feature escolhido foi a queda no fluxo e a condição foi um limite de relevância dessa queda. Para fazer essa escolha, o algoritmo analisa o par feature-condição que faz com que um nó produza filhos com o menor gini, ou seja, os mais puros possíveis. Com isso, temos que a função custo que deve ser minimizada para este caso se dá por

$$J(x,t_x) = \frac{b_\mathrm{e}}{m} G_\mathrm{e} + \frac{b_\mathrm{d}}{m} G_\mathrm{d}, \tag{3.13}$$

onde m é o número total de amostras, b é o número de amostras do novo subconjunto, G é a sua pureza, os índices “e"e “d"indicam os subconjuntos esquerdo e direito, respectivamente. Esses pesos que acompanham os G se dão porque os tamanhos dos subconjuntos também devem ser levados em conta.

Após a primeira divisão, o CART repete o mesmo procedimento para os subconjuntos gerados e vai fazendo isso até chegar em uma profundidade predeterminada ou até que ele não consiga mais achar uma divisão que reduza a impureza dos subconjuntos. A profundidade

da árvore é, então, um hiperparâmetro do algoritmo que podemos estabelecer. Outros hiperparâmetros surgem para impedir um overfitting, como o número mínimo de amostras que cada nó deve ter antes de ser dividido, número mínimo de amostras que uma folha deve ter, dentre outros. Isso porque o algoritmo faz poucas suposições sobre os dados que estão sendo trabalhados e se deixarmos a árvore crescer muito ela se adaptará quase que perfeitamente aos dados de treino, o que resulta em uma má generalização para novos dados.

3.5.2 Ensemble Methods

Como comentado anteriormente, um **ensemble method** é um algoritmo que agrega vários algoritmos preditores (os chamarei de subalgoritmos por praticidade) e leva em consideração todas as predições individuais feitas para dar a sua predição final. No caso de problemas de classificação, ainda é possível chamar esse algoritmo de *ensemble classifier*. Uma forma simples e direta de se fazer essa predição final é considerar a classe que foi mais votada pelos subalgoritmos. Quando isso acontece, o algoritmo se caracteriza como um *Hard Voting Classifier* (HVC). É comum que um HVC tenha uma acurácia maior que o melhor subalgoritmo implementado nele. Em termos matemáticos, isso se dá por

$$\hat{y} = \text{moda}\left\{C_1(\boldsymbol{x}), C_2(\boldsymbol{x}), ..., C_R(\boldsymbol{x})\right\}, \tag{3.14}$$

em que $\hat{y}$ é a classe predita do algoritmo com base na moda das classificações individuais e C_j é a classificação do j-ésimo subalgoritmo dos R usados no ensemble.

Em um HVC, ainda é possível atribuir pesos às classificações dos subalgoritmos. Com isso, a generalização da equação 3.14 se dá por (RASCHKA; MIRJALILI, 2019)

$$\hat{y} = \arg\max_i \sum_{j=1}^{R} w_j \, \chi_A(C_j(\boldsymbol{x}) = i), \tag{3.15}$$

onde w_j é o peso da j-ésima classificação, A é o conjunto de labels ou o conjunto de todas as possíveis classes (“planeta" ou “não-planeta", por exemplo) e χ_A é a *função característica* (ou *função de agregação*) que retorna 1 se a $C_j(\boldsymbol{x})$ corresponder a uma classe i ou 0, do contrário. Essa equação procura o argumento i ($i \in A$) onde essa soma é máxima, ou seja, a classe mais votada.

Em contra partida, um ensemble classifier também pode ser caracterizado como um *Soft Voting Classifier* (SVC). Isso se dá quando todos os subalgoritmos são capazes de estimar a probabilidade de sua predição estar correta. Com base na equação 3.15, essas probabilidades são usadas no lugar dos labels retornados pela função χ_A, de forma que

$$\hat{y} = \arg\max_{i} \sum_{j=1}^{R} w_j \, p_{i,j}, \tag{3.16}$$

em que $p_{i,j}$ é a probabilidade calculada pelo j-ésimo subalgoritmo para uma classe i. É comum que o SVC se sobressaia em termos de performance em relação ao HVC, já que o primeiro dá mais ênfase às classificações mais confiáveis. Naturalmente, estamos considerando que os subalgoritmos estejam devidamente calibrados, bem como os seus dados de treino.

Um ensemble diversificado é importante pois o modelo se torna menos enviesado e menos excessivamente sensível a pequenas variações nos dados de treino. Essa diversificação pode ser obtida usando subalgoritmos diferentes no mesmo ensemble. Poderíamos, por exemplo, usar regressão logística, gradiente descendente estocástico e árvore de decisão como subalgoritmos de um ensemble e avaliar suas predições como comentado acima.

Uma outra forma de se ter uma diversidade no ensemble consiste na técnica de usar subalgoritmos iguais, mas os treinando com diferentes subconjuntos do conjunto de treino completo. Esses subconjuntos são formados por amostras selecionadas aleatoriamente do conjunto de treino e essa seleção pode ser com reposição ou não. Uma seleção com reposição significa que quando uma dada amostra no conjunto de treino é selecionada para um subconjunto, ela poderá ser selecionada novamente para o mesmo subconjunto. Esse método é conhecido como *bootstrap aggregatin*, *bootstrapping* ou apenas ***bagging***. Quando não há reposição, a técnica é conhecida como ***pasting***.

3.5.3 Floresta Randômica: Características Adicionais

Como citado, o algoritmo de floresta randômica é um ensemble method que agrega várias árvores de decisão para fazer sua predição final. Além da aleatoriedade inerente do bagging (que é comumente usado neste algoritmo), o seu caráter randômico é acentuado pelo fato de um subconjunto de features ser selecionado aleatoriamente para ser considerado em cada amostra. Isso aumenta ainda mais a diversidade do ensemble e resulta em um modelo melhor.

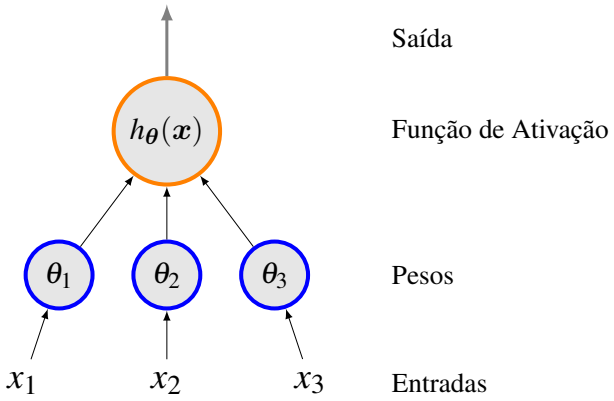
Na discussão sobre o modelo de árvore de decisão, vimos que esse algoritmo busca um par feature-condição que minimizasse o gini dos subconjuntos produzidos. É possível deixar o algoritmo de floresta randômica ainda mais aleatório quebrando essa busca e selecionando valores randômicos das condições para cada feature. Com isso, temos algo conhecido como um *ensemble de árvores extremamente randomizadas*. Como uma vaga analogia ao caso de usar um

Gradiente Descendente Estocástico em vez de uma Regressão Logística, esse algoritmo é mais rápido de ser treinado, uma vez que não é mais necessário computar a melhor condição possível para cada feature em cada nó. Em termos de performance, não é possível saber previamente qual dos modelos se sairá melhor e essa modificação não foi considerada neste trabalho.

3.6 Rede Neural Artificial

Uma **rede neural artificial** é um modelo de aprendizado de máquina que consiste em uma série de operações lógicas inspiradas nas redes neurais biológicas dos nossos cérebros. Nessa comparação, o neurônio biológico é representado por uma *unidade logística*, ou *perceptron*, como mostra a figura 23.

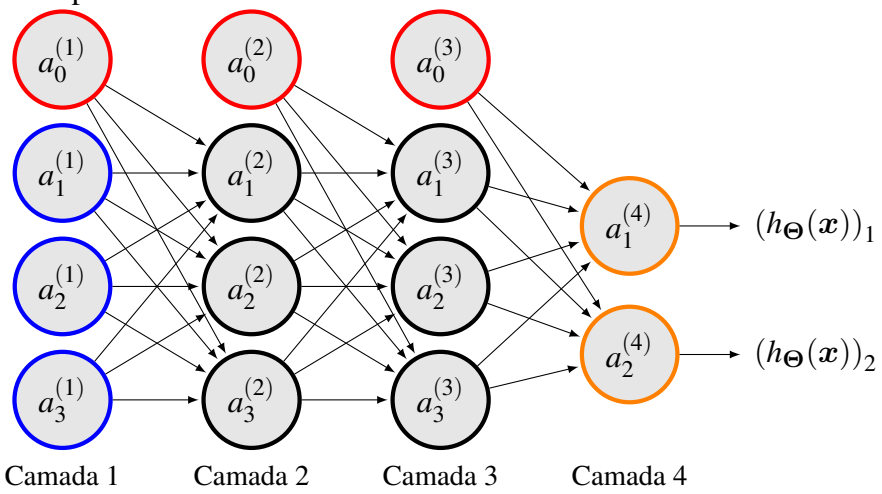
Figura 23 – Esquema de uma unidade logística.



Fonte: Elaboração própria.

Essa unidade recebe dados de entrada, multiplica cada uma deles por um parâmetro θ, soma esses produtos e então calcula um novo valor ao inserir essa soma em uma função chamada de *função de ativação*. Isso é exatamente o que acontece no caso da equação 3.4 para a regressão logística. Uma ressalva importante é que a equação 3.4 leva em consideração um termo de viés (x_0 e θ_0) que não está incluso na figura, mas que deve ser considerado no modelo. O grande diferencial de uma rede neural é a sua capacidade de relacionar essas unidades em uma rede para conseguir descrever dados complexos de forma otimizada. A forma como essas unidades são conectadas é chamada de *arquitetura da rede* e um exemplo se dá pela figura 24.

Figura 24 – Exemplo de rede neural.



Fonte: Elaboração própria.

A notação para as descrições matemáticas das redes neurais é um pouco diferente da usada na regressão logística. Os termos $a_i^{(j)}$ representam a função de ativação da unidade logística i na camada j. Em particular, a camada $j = 1$ é chamada *camada de entrada*, pois é nela onde os dados entram para serem analisados ($a_i^{(1)} = x_i$). Além disso, $a_0^{(j)}$ é o termo de viés da j-ésima camada (que, apesar de mostrado na figura, é usualmente omitido e implícito). Temos então que[22]

$$\boldsymbol{a}^{(j)} = g\left(\boldsymbol{z}^{(j)}\right), \tag{3.17}$$

onde $\boldsymbol{a}^{(j)}$ é o vetor com todas as funções da camada j e $\boldsymbol{z}^{(j)} = \boldsymbol{a}^{(j-1)}\boldsymbol{\Theta}^{(j)}$.

Em vez de um vetor para representar os pesos, agora fazemos uso de uma matriz $\boldsymbol{\Theta}^{(j)}$ que mapeia os parâmetros de uma camada j até uma camada $j+1$. Se uma camada j tem s_j unidades e uma camada $j+1$ tem s_{j+1}, então a dimensão da matriz $\boldsymbol{\Theta}^{(j)}$ é $s_{j+1} \times (1+s_j)$. A última camada é chamada *camada de saída*, pois é nela onde o resultado do processamento sai e nos é exibido. Com isso, temos que $(h_\Theta(x))_i = a_i^{(L)}$, onde L é o número total de camadas (e, portanto, a última nesta notação). As demais camadas entre as de entrada e saída são culturalmente chamadas de *camadas ocultas*, pois não vemos diretamente os seus valores. Todas as camadas, exceto a de saída, possuem uma unidade de viés.

Ao receber um dado, o fluxo das operações de uma rede neural começa na camada de entrada e vai até a camada de saída. Isso é chamado de *forward propagation*. O algoritmo mais popular para treinar redes neurais é o de *backpropagation*. Como o nome sugere, o fluxo de

[22] Termos em negrito representam notações matriciais.

processamento desse algoritmo vai no sentido contrário do que foi visto antes e os cálculos têm muita relação com o algoritmo de gradiente descendente.

Os detalhes matemáticos desse processo não serão abordados aqui, mas uma descrição qualitativa pode ser feita. De forma muito sucinta, ele se dá dividindo o conjunto de treino em pequenos grupos (chamados de *mini-batches*) e os passando para a rede neural, que tem seus parâmetros Θ iniciados aleatoriamente. Dizemos que o algoritmo completou um *epoch* quando ele usa todos esses grupos uma vez. Essa passagem é o forward propagation comentado acima. Todos os resultados de cada unidade logística são guardados para serem usados na etapa de backpropagation.

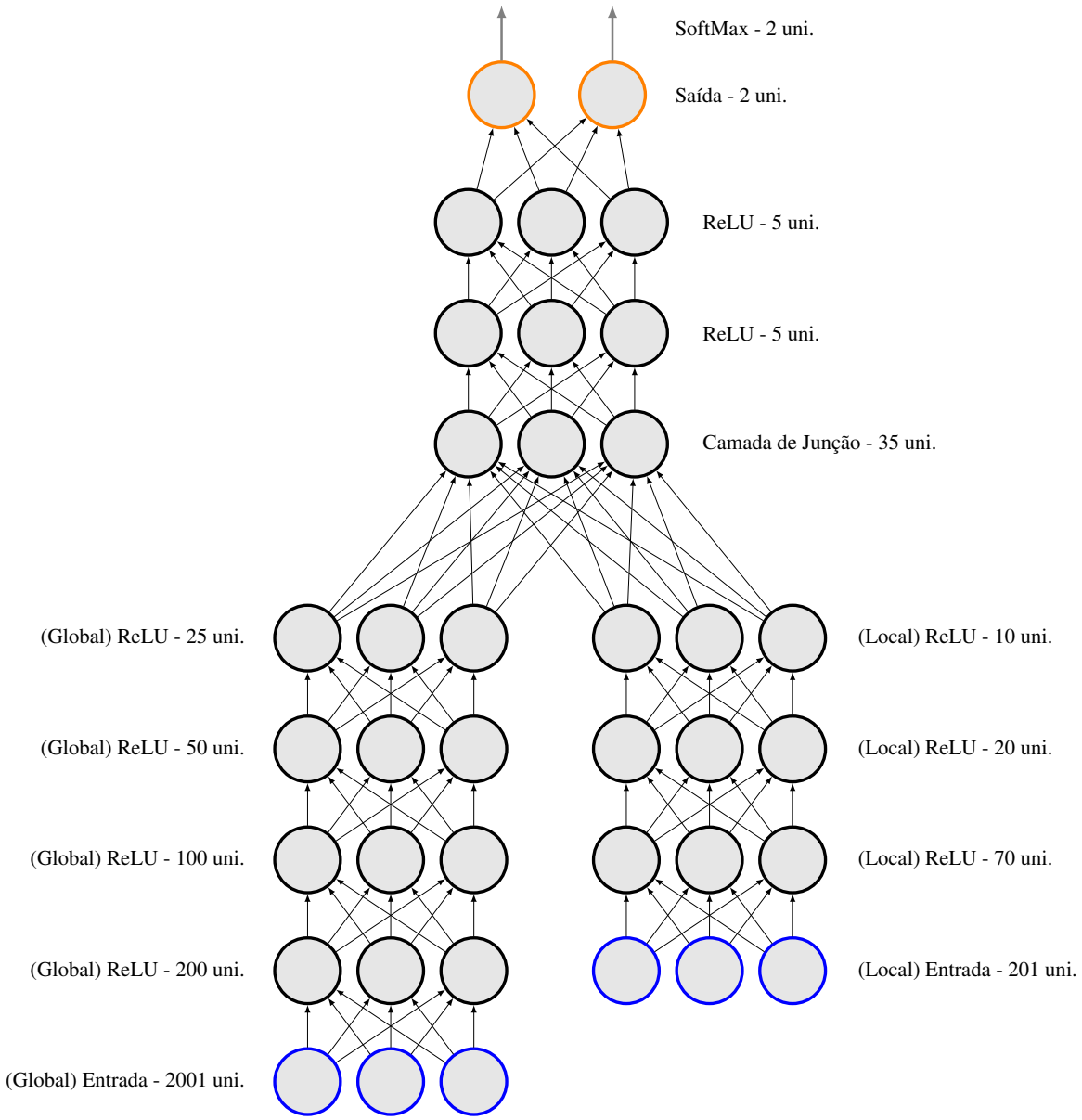
Ao chegar no final da rede com uma amostra, a diferença da predição feita pela última camada com o resultado esperado é avaliada, bem como o quanto cada unidade logística contribuiu para esse erro. Isso é feito computando o gradiente do custo naquela camada. Disso, o algoritmo avalia o quanto cada unidade da camada anterior contribuiu para o erro, também pelo gradiente, e o vai fazendo até chegar no começo da rede. Essa medição dos erros leva em consideração os resultados de cada unidade gerados e guardados no processo do forward propagation. Usando todos esses erros, o algoritmo realiza um gradiente descendente para achar os valores dos parâmetros que minimizam os erros cometidos pelas unidades da rede e os ajusta de acordo. Esse processo é repetido para todas as amostras de um mini-batch, e então para todos os mini-batches do conjunto de treino e repete tudo outra vez em uma quantidade de epochs definidas previamente. O número de epochs é escolhido para que o algoritmo consiga convergir para um resultado otimizado e varia em cada situação. Por conta disso, o epoch é um hiperparâmetro que influencia diretamente a performance do modelo.

Na discussão acerca da equação 3.4, a função sigmoidal foi escolhida para $g(z)$. No entanto, essa não é a única escolha possível. De fato, outras escolhas comuns para a função de ativação são a tangente hiperbólica, *Rectified Linear Unit* (ReLU) e Softplus (ver figura 47). Dessas, a ReLU se mostra particularmente vantajosa em relação às sigmoidal e tangente hiperbólica. Isso porque, utilizando as duas últimas citadas, os gradientes do modelo praticamente se anulam quando o algoritmo de backpropagation se aproxima das primeiras camadas, o que causa um problema conhecido como *problema do gradiente evanescente*. Além de evitar esse problema, as computações feitas com a ReLU são bem mais rápidas dado seu caráter linear.

É possível combinar várias funções de ativação em uma mesma rede. No caso de redes neurais para problemas de classificação, temos mais duas delas que são úteis quando usadas

na camada de saída: a ArgMax e a SoftMax. Tomemos como exemplo uma arquitetura de rede neural usada neste trabalho, dada pela figura 25.

Figura 25 – Arquitetura de rede neural usada.



Fonte: Elaboração própria.

Todas as camadas usam a ReLU como função de ativação, exceto a última. Da forma como está construída, a rede retorna as "probabilidades"[23] de que uma amostra seja `não-planeta` na primeira unidade da camada de saída e `planeta` na segunda. A SoftMax

[23] Está entre aspas, pois esses valores dependem de como os parâmetros da rede foram inicializados. Isso porque o algoritmo pode convergir para um mínimo diferente em cada treino, dependendo do seu ponto de partida no espaço dos parâmetros. Um termo mais apropriado para esse valor talvez seja a "confiança" que um modelo tem, com base no que aprendeu e com base em como aprendeu (dependendo da configuração de seus hiperparâmetros), de que uma amostra pertence a uma dada classe.

recebe todos os valores de cada unidade logística que a camada de saída recebe da camada anterior e os normaliza em valores entre 0 e 1. A equação que a descreve é

$$\boldsymbol{a}^{(L+1)} = \left[\sum_i \exp\left(a_i^{(L)}\right)\right] \exp\left(\boldsymbol{a}^{(L)}\right) \tag{3.18}$$

Nessa expressão, $L+1$ é apenas uma notação conveniente para dizer que a SoftMax realiza seus cálculos com base nos valores recebidos pela camada de saída e retorna algo a mais. De forma didática, podemos pensar em uma camada além da de saída cuja função é tratar os valores "brutos"recebidos dessa última antes de exibir os resultados finais. Essa ressalva se dá porque a representação da rede na figura 25 mostra uma relação de duas unidades logísticas na camada de saída para três unidades na camada anterior, o que seria incompatível com a equação 3.18 considerando que a rede só retorna dois valores.

Como as classes são mutuamente excludentes, a SoftMax garante que a soma dessas probabilidades seja 1. Além disso, essa função pode ser usada na etapa de treino da rede, já que ela tem uma derivada conveniente para ser usada no backpropagation. Esse não é o caso, por exemplo, ao usar a função de ativação ArgMax na camada final. Ela funciona com a mesma analogia do SoftMax, porém retorna o valor 1 para a classe mais provável e 0 para as demais. Por conta dessa característica, ela só é usada depois que a rede é treinada. Com a ArgMax, a rede consegue informar suas classificações de forma mais direta e assertiva em um contexto onde o modelo já esteja em sua fase de aplicação real.

4 RESULTADOS E DISCUSSÕES

Todos os algoritmos discutidos no capítulo anterior tiveram uma função muito específica: classificar padrões em curvas de luz como `planeta` ou `não-planeta`. A performance de cada algoritmo é avaliada com base em seu desempenho com os dados do conjunto de teste. Para isso, avaliamos as suas *medidas de performance*. A próxima seção discute brevemente algumas delas antes de compararmos os modelos.

4.1 Medidas de Performance

Quando um algoritmo faz uma classificação de um dado, podemos fazer quatro avaliações sobre esse resultado. Tomando como exemplo o nosso dataset, se o label de um TCE é `planeta` e o algoritmo o classifica como tal, então esse resultado é um *positivo verdadeiro* (P_V). Se o algoritmo classificasse esse planeta como `não-planeta`, então esse resultado seria um *falso negativo* (F_N). Se o label de um TCE é `não-planeta` e o algoritmo o classificasse da mesma forma, então esse resultado seria um *negativo verdadeiro* (N_V). Finalmente, se esse TCE fosse classificado como `planeta` pelo algoritmo, então o resultado seria um *falso positivo* (F_P). Essas avaliações podem ser sintetizadas em uma *matriz de confusão*, como a seguinte:

Tabela 2 – Matriz de confusão genérica.

		Predito	
		Negativo	Positivo
Original	Negativo	N_V	F_P
	Positivo	F_N	P_V

Fonte: Adaptado de (GÉRON, 2019).

A medida de performance mais imediata na avaliação de um algoritmo classificador é a sua *acurácia*, que é o percentual de acertos que o algoritmo alcançou. Supondo que o teste tenha sido feito em todo o conjunto de teste, com m_t amostras, então

$$\text{acurácia} = \frac{P_V + N_V}{m_t}.$$

Neste caso em particular, temos que m_t é numericamente igual à soma de todos os resultados, ou seja, $m_t = P_V + N_V + F_P + F_N$. O percentual de acertos de um algoritmo não deve ser superestimado, especialmente em situações onde o conjunto de dados é desbalanceado. Um

conjunto desbalanceado é um aquele onde a quantidade de amostras das classes consideradas pelo modelo são bastante desiguais. Um exemplo disso é o nosso próprio conjunto de dados, onde a classe `planeta` representa apenas 23% de todo o conjunto de dados[24]. Isso implica que um algoritmo classificador programado para apenas retornar resultados negativos (`não-planeta`), independentemente das características da curva de luz, estaria correto 77% das vezes.

A ideia da acurácia pode ser refinada em uma nova métrica ao avaliarmos apenas a acurácia das predições positivas. Isso é chamado de *precisão* e ela se dá por

$$\text{precisão} = \frac{P_V}{P_V + F_P}.$$

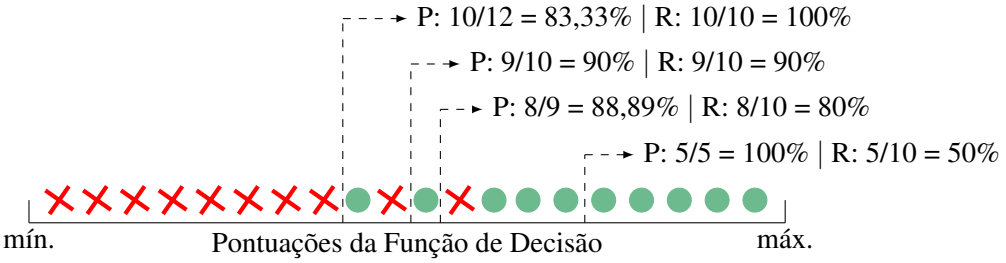
Em outras palavras, a precisão nos diz a porção dos positivos selecionados que são verdadeiros. Uma métrica que sempre é levada em conta quando se fala em precisão é o *recall* (também chamado de revocação ou sensibilidade), dado por

$$\text{recall} = \frac{P_V}{P_V + F_N}.$$

Ou seja, o recall nos diz a porção dos verdadeiros positivos que foram identificados corretamente em relação a todos os positivos que existem no dataset. Ele também é chamado de sensibilidade por avaliar a eficácia que o algoritmo tem em detectar resultados positivos com sucesso. Um valor alto em ambas as medidas é importante para que o algoritmo tenho um bom desempenho, porém aumentar a precisão implica em diminuir o recall e vice-versa.

O algoritmo calcula se um dado pertence à classe positiva ou à negativa com base em uma pontuação que uma *função de decisão* retorna. Essa decisão se dá com base nas fronteiras de decisão discutidas no capítulo anterior. Se o valor retornado pela função de decisão for maior que um limite estabelecido, por exemplo, então o dado é classificado como positivo. A escolha desse limite afeta diretamente os valores de precisão e recall, como mostra a figura 26.

Figura 26 – Exemplo de troca entre Precisão (P) e Recall (R) para diferentes limites da pontuação da função de decisão.



Fonte: Elaboração própria.

[24] Os labels AFP e NTP, descritas na tabela 1, foram mescladas em uma única classe “não-planeta".

A tendência da precisão é aumentar quando o algoritmo fica mais rigoroso com a pontuação, mas em alguns pontos ela pode cair, como é o caso do terceiro limite apontado na figura. Em geral, temos liberdade para escolher esse limite de forma a otimizar a troca precisão-recall que mais atenda às necessidades do trabalho. No caso particular deste, é preferível um recall um tanto maior que a precisão para reduzir a perda de planetas (especialmente em dados nunca explorados). Como podemos escolher esses valores, podemos construir um gráfico da presição em função do recall para avaliarmos as performances de modelos com essas medidas. Bons modelos têm uma *Área Sob a Curva* (AUC)[25] elevada.

Uma outra métrica importante é a *especificidade*. Assim como o recall pode ser entendido como a capacidade que o algoritmo tem de detectar positivos corretamente, a especificidade é a capacidade que algoritmo tem de detectar negativos corretamente. Ela se dá por

$$\text{especificidade} = \frac{N_V}{N_V + F_P}.$$

Dessa forma, a especificidade também pode ser chamada de taxa de negativos verdadeiros. Com ela, podemos ainda derivar a *Taxa de Falsos Positivos* (TFP), dada por

$$\text{TFP} = 1 - \text{especificidade} = \frac{F_P}{F_P + N_V}.$$

Pela figura 26, a TFP aumenta quando o recall também aumenta. Um gráfico do recall, chamado de *Taxa de Positivos Verdadeiros* (TPV) neste contexto, em função da TFP pode ser gerado para comparar os modelos de uma forma semelhante como o da precisão-recall. A curva desse gráfico é conhecida como *Receiver Operating Characteristic* (ROC). Bons modelos também têm a ROC-AUC[26] elevada.

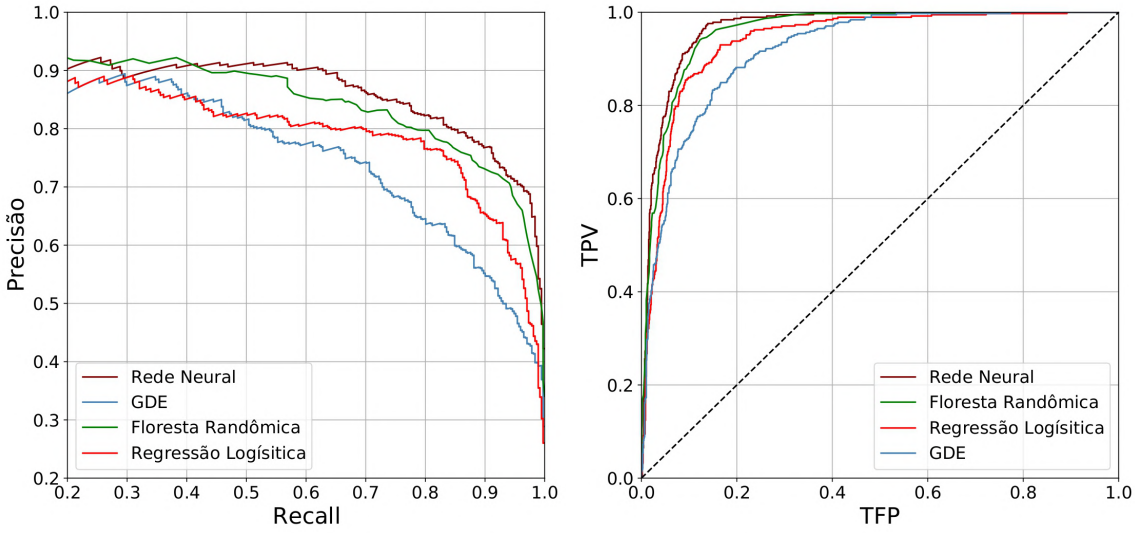
4.2 Comparação dos Modelos

Um gráfico da precisão em função do recall foi gerado para cada modelo, assim como os da taxa de positivos verdadeiro em função da taxa de falsos positivos. A figura 27 mostra os primeiros resultados com os dados de teste.

[25] AUC se dá por *Area Under Curve*.

[26] AUC será designada para a área sobre a curva de precisão em função do recall, ao passo que ROC-AUC designará a área sobre a curva ROC.

Figura 27 – Medidas iniciais de performance dos modelos testados.



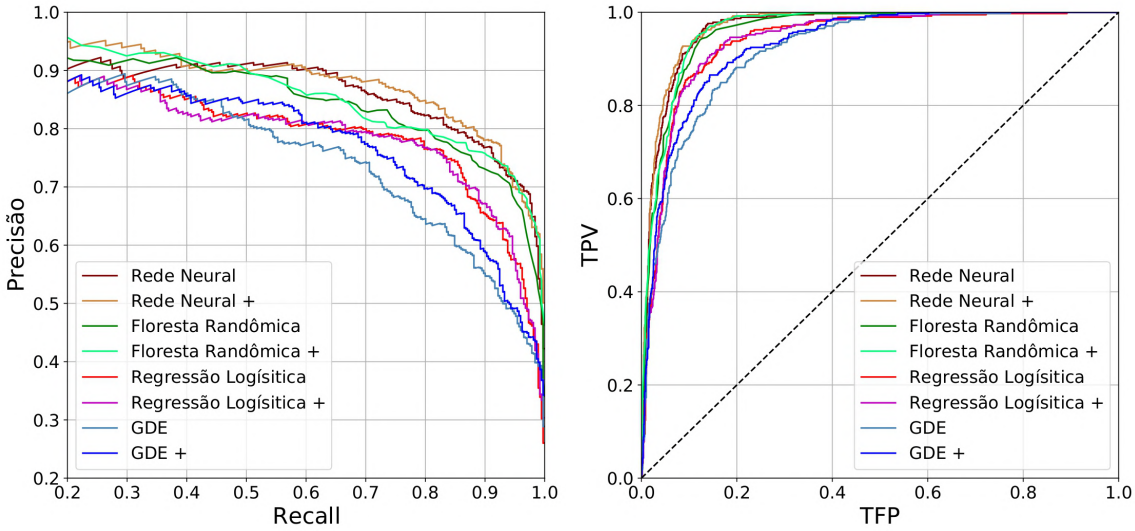
Fonte: Elaboração própria.

Pela discussão da seção 3.4, era esperado que o GDE não oferecesse resultados ótimos em relação à Regressão Logística. Isso foi realmente observado, mas a discrepância entre essas duas curvas mostra que a velocidade de treinamento do GDE não compensa sua performance. Também era esperado que o modelo de floresta randômica se sobressaísse em relação aos dois anteriores, mas o modelo de rede neural se mostrou o mais eficaz dos testados.

Um ponto de preocupação levantado ainda na seção anterior foi o fato de que o conjunto de dados é desbalanceado, onde as curvas que contém planetas representam apenas 23% do total. Uma forma para tentar amenizar isso foi copiando as curvas onde há planetas, refletindo-as horizontalmente e inserindo essas novas curvas de luz no conjunto de dados. Essa é uma das técnicas de *aumento de dados* mais simples e bastante usada em situações onde os dados disponíveis são limitados (SHORTEN; KHOSHGOFTAAR, 2019). Apesar de ser usada em conjuntos de dados constituídos por imagens, é perfeitamente plausível aplicá-la nas curvas de luz. Isso porque o padrão de queda no fluxo luminoso para planetas é preservado nessa reflexão. O mesmo acontece para quaisquer características astrofísicas ou instrumentais de eventos que não caracterizam planetas.

Após esse aumento, os modelos foram treinados mais uma vez e a figura 28 mostra o efeito dos novos dados no dataset (representados por um “+").

Figura 28 – Medidas de performance dos modelos testados com dados aumentados.



Fonte: Elaboração própria.

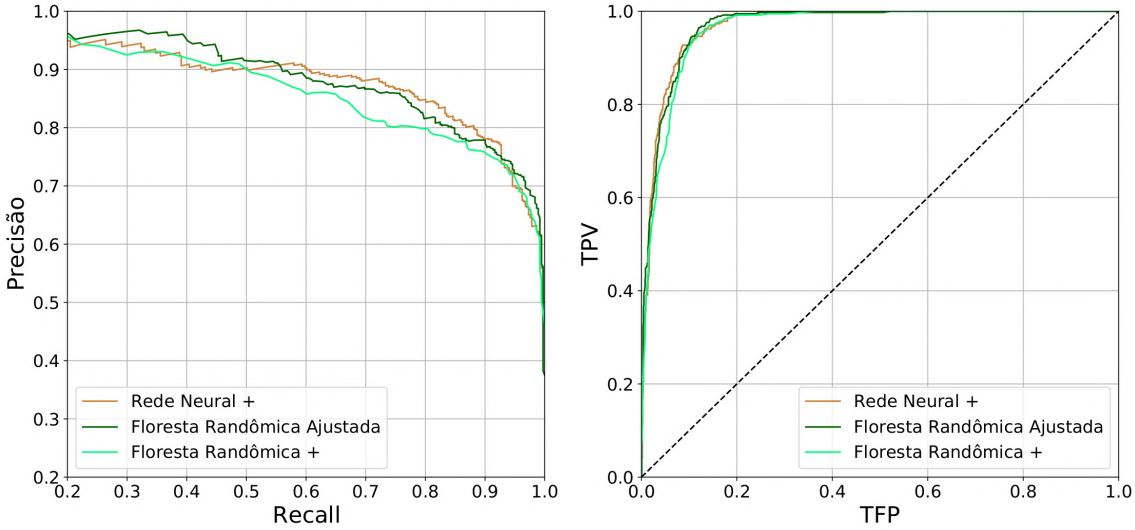
O aumento de performance do GDE foi considerável, apesar de ainda não apresentar vantagem em relação ao modelo de Regressão Logística. A performance desse último se manteve essencialmente inalterada. Ambos os modelos de Floresta Randômica e Rede Neural apresentaram um leve aprimoramento nos resultados.

Mais técnicas de aumento de dados poderiam ter sido aplicadas, mas uma forma mais eficaz de obter melhores resultados com os modelos é fazendo um *ajuste de hiperparâmetros*. A ideia é selecionar os dois mais promissores, testar diferentes combinações dos valores dos seus hiperparâmetros e ver quais dessas combinações retornam os melhores resultados. Para cada configuração de hiperparâmetros, um novo modelo é criado e treinado. Disso, ele é testado com os dados de validação e uma métrica de performance é calculada. O processo é repetido para todas as possíveis combinações de valores estabelecidos. Vale ressaltar que os dados de teste ficam completamente indisponíveis aos modelos nesse processo, pelos motivos discutidos no capítulo anterior.

Um exemplo de ajuste foi feito com o modelo de Floresta Randômica, onde alguns valores de três hiperparâmentros foram testados. O primeiro foi se o método de bagging (como explicado na subseção 3.5.2) seria aplicado ou não. Em caso negativo, todo o conjunto de dados é usado para construir cada árvore do modelo. O segundo hiperparâmetro foi o número de árvores que constituem a floresta. Os valores testados foram 30, 100 e 300. O terceiro foi o número de features que o modelo deve considerar ao fazer as divisões dos nós explicadas na seção 3.5.1. Os valores testados foram 10, 100, 126 (valor inteiro aproximado de $\sqrt{m}$, usado por padrão pelo

Scikit-Learn[27]) e 200. Com isso, a busca tem $2 \times 3 \times 4 = 24$ possíveis configurações para testar. O processo levou aproximadamente 17 horas e a melhor combinação desses hiperparâmetros foi um modelo sem bagging, com 300 árvores de decisão usando 200 features. A figura 29 mostra o modelo com essas configurações de hiperparâmetros comparado com os dois melhores da análise anterior.

Figura 29 – Modelo de Floresta Randômica com hiperparâmetros ajustados.



Fonte: Elaboração própria.

Com o ajuste, o modelo de Floresta Randômica pôde alcançar uma performance comparável com a da Rede Neural. Todos os modelos dos gráficos acima usam o conjunto dados aumentado.

Um cuidado que deve ser tomado ao término de um processo de ajuste de hiperparâmetros é observar qual dos valores disponíveis na procura entregaram os melhores resultados. No caso acima, os maiores valores disponíveis para o número de árvores e de features foram selecionados, o que nos dá margem para testar valores mais elevados para esses hiperparâmetros em busca de uma configuração que retorne um modelo ainda melhor. O mesmo valeria se os valores ótimos fossem os mínimos disponibilizados, onde teríamos espaço para analisar valores ainda menores.

No caso das redes neurais, possíveis hiperparâmetros para testar são a taxa de aprendizagem usada no backpropagation, a quantidade de epochs, a função de ativação usada nas camadas ocultas e a própria arquitetura da rede (número de camadas, número de unidades logísticas da j-ésima camada, entre outros). Inicialmente, uma arquitetura de rede descrita de forma simplificada pela tabela 3 foi utilizada.

[27] (PEDREGOSA *et al.*, 2011)

Tabela 3 – Descrição simplificada de uma das arquiteturas de rede neural usadas.

Camada	Ativação	Num. Unidades
Saída	Sigmoid	2
Oculta 3	ReLU	10
Oculta 2	ReLU	100
Oculta 1	ReLU	Entre 20 e 620
Entrada	ReLU	2202

Fonte: Elaboração própria.

Os hiperparâmetros procurados para ela foram o número de unidades na primeira camada, e a taxa de aprendizagem. Os valores testados para o número de unidades iam de 20 a 620 espaçados em 40 unidades (20, 60, 100, ...). As taxas de aprendizagem testadas foram 1^{-2}, 1^{-3} e 1^{-4}. O processo para avaliar as $15 \times 3 = 45$ possíveis configurações levou pouco mais de 3 horas e o melhor resultado foi a rede tendo uma taxa de aprendizagem de 1^{-2}, com sua primeira camada oculta constituída por 180 unidades logísticas.

A arquitetura mostrada na figura 25 foi a rede testada que entregou os melhores resultados (mostrados na figura 28). Ela foi construída inicialmente de forma arbitrária e os hiperparâmetros relativos à sua arquitetura (número de unidades por camada, apenas) foram parcialmente ajustados com auxílio do KerasTuner (O'MALLEY *et al.*, 2019). Os demais hiperparâmetros não foram ajustados em tempo viável para o período de conclusão da escrita deste trabalho (tanto os da Floresta Randômica quanto os da Rede Neural), dada a longa cadeia de processamentos e os baixos recursos computacionais disponíveis para esta tarefa. Resultados mais atualizados poderão ser encontrados na página do GitHub deste projeto à medida em que eles forem sendo obtidos. Com base em toda esta discussão, a tabela 4 resume os resultados de performance obtidos nos testes de todos os modelos (com os dados aumentados).

Tabela 4 – Desempenho alcançado com os modelos testados.

Modelo	AUC	ROC-AUC	Acurácia
Regressão Logística	0.79	0.94	0.89
GDE	0.78	0.93	0.86
Floresta Randômica	0.87	0.96	0.91
Floresta Randômica Ajustada	0.89	0.97	0.91
Rede Neural 1 (tab. 3)	0.85	0.96	0.90
Rede Neural 2 (fig. 25)	0.88	0.97	0.90

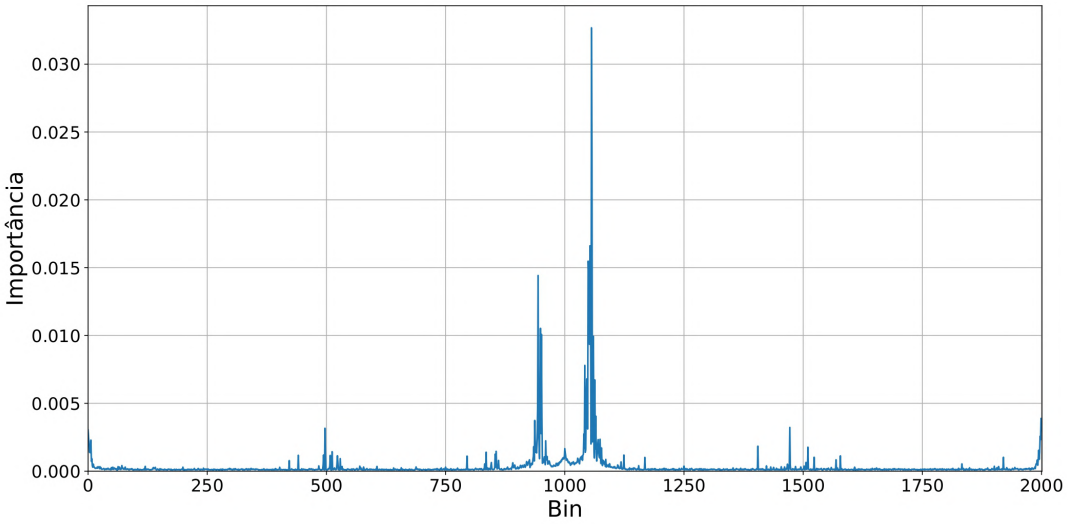
Fonte: Elaboração própria.

4.3 Análises dos Melhores Modelos

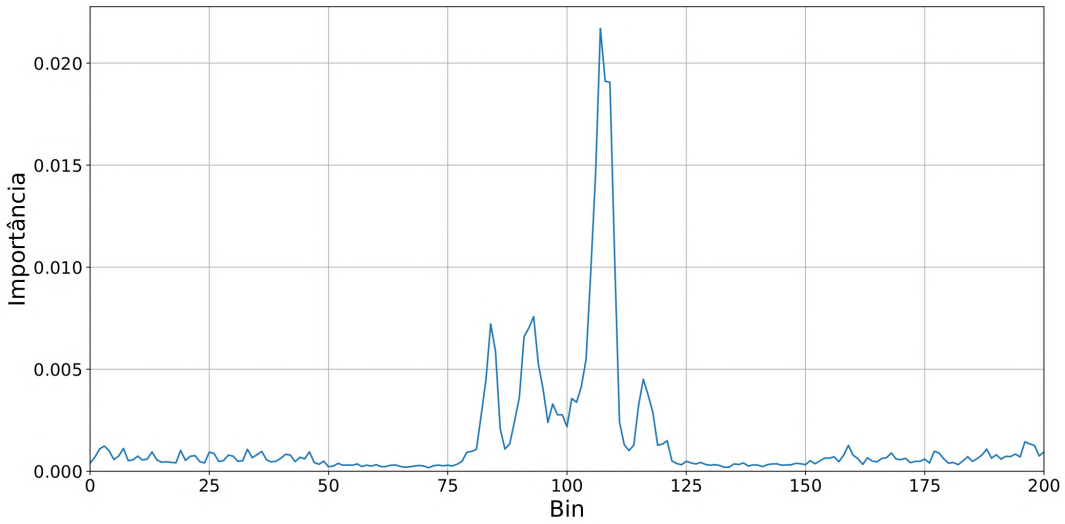
Como mostra a tabela 4, os melhores modelos testados foram os de Floresta Randômica (com os hiperparâmetros ajustados) e o da Rede Neural com a arquitetura conforme a figura 25. Contudo, suas respectivas acurácias de 91% e 90%, bem como quaisquer outras medidas de performance, nem sempre são suficientes para caracterizar completamente todas as vantagens e desvantagens dos modelos. Além de boas métricas, bons modelos devem acertar os seus resultados pelos motivos certos. Com base nisso, podemos avaliar quais foram as informações mais relevantes que eles precisaram usar para fazer as suas classificações. Os gráficos da figura 30 mostram os features mais importantes para a decisão do modelo de Floresta Randômica nas curvas globais e locais.

Figura 30 – Relevância geral dos features para a classificação.

(30.a) Features da Curva Global



(30.b) Features da Curva Local



Fonte: Elaboração própria.

Com o que foi discutido na seção 3.1, era esperado que os bins centrais das curvas tivessem forte impacto nas classificações, o que é observado. Isso porque todas as quedas no fluxo foram centralizadas. No caso das curvas globais, também era esperado que features mais afastados do centro tivessem alguma contribuição mais relevante, como no caso de estrelas binárias mostrado na figura 14. Além disso, vemos dois picos ao redor dos features mais centrais, o que sugere que não só a queda no fluxo é importante, mas a forma como esse fluxo cai também é. Isso é plausível, uma vez há uma diferença recorrente no formato da queda gerado por planetas e por eclipses binários, por exemplo[28]. Foi observado que o processo de binning dos dados desloca os dados um pouco para a direita do centro, o que pode explicar diferença de altura dos picos.

Vale ressaltar que ambos os modelos de Floresta Randômica e de Rede Neural são modelos tidos como “caixas-pretas". Isso porque entender os motivos que os levam a tomar as suas decisões não é uma tarefa trivial, dada a complexidade dos mesmos (KRAUSE *et al.*, 2016). Apesar disso, a análise breve feita pelos gráficos da figura 30 já indica que a modelagem feita por eles é coerente com a natureza do problema abordado.

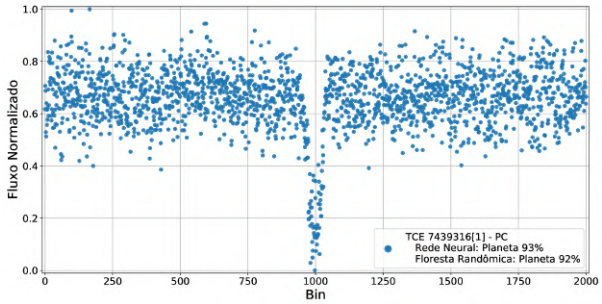
Pudemos avaliar como esses features afetam a decisão dos modelos criando um novo conjunto de dados para testes, mas com amostras das curvas globais modificadas. Alguns planetas tiveram seus trânsitos omitidos, outros tiveram trânsitos secundários implantadas em suas curvas de luz para simular eclipses e alguns eclipses verdadeiros tiveram seus trânsitos secundários omitidos. Exemplos dos resultados obtidos podem ser vistos na figura 31. Todos os testes foram repetidos com mais exemplos semelhantes, de forma que as análises qualitativas a seguir são representativas e se mantém para a maioria dos dados do conjunto de teste.

No primeiro teste, as figuras 31.a e 31.b mostram o trânsito de um planeta real que foi omitido e substituído por cópias dos dados dos seus arredores. Com o trânsito, o modelo de Rede Neural (RN) classificou esse TCE como `planeta` com 93% de confiança, ao passo que o de Floresta Randômica (FR) o classificou como `planeta` com uma confiança de 92%. Ambas as porcentagens aumentaram para 95% ao omitir o trânsito na curva global. Isso indica que ambos os modelos dão mais atenção para as curvas locais, especialmente quando não há algum sinal relevante nas curvas globais. Isso era esperado, já que algumas curvas tiveram seus trânsitos parcial ou totalmente obstruídos no processo do tratamento e geração das curvas globais. Esse foi, a propósito, um dos maiores motivos para se gerar duas representações do mesmo TCE.

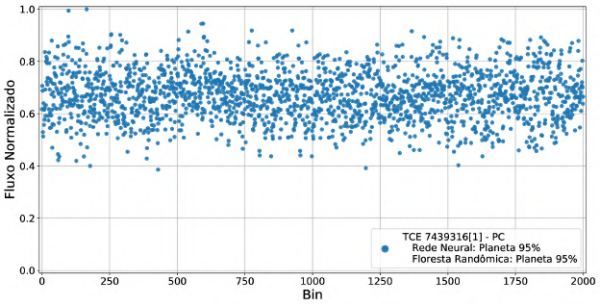
[28] Em geral, as curvas que representam planetas têm um formato semelhante a um “U", ao passo que as dos eclipses têm um formato semelhante a um “V". Exemplos disso podem ser vistos nas figuras 42 e 43

Figura 31 – Predições dos modelos para diferentes modificações em curvas de luz.

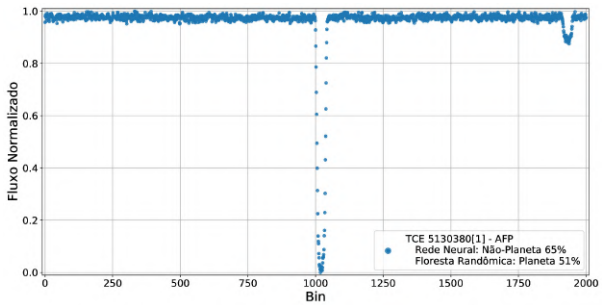
(31.a)



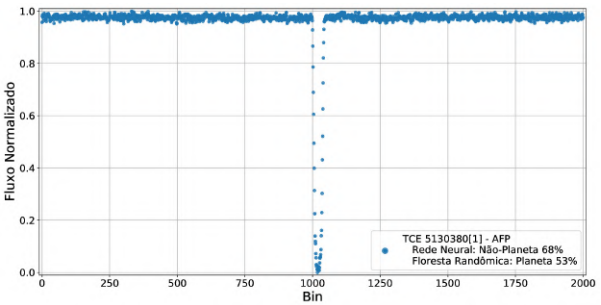
(31.b)



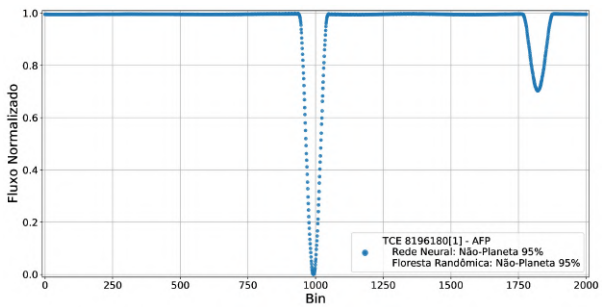
(31.c)



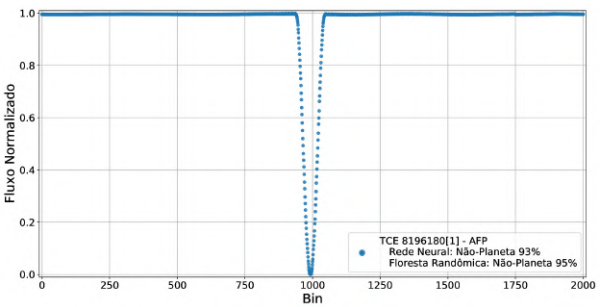
(31.d)



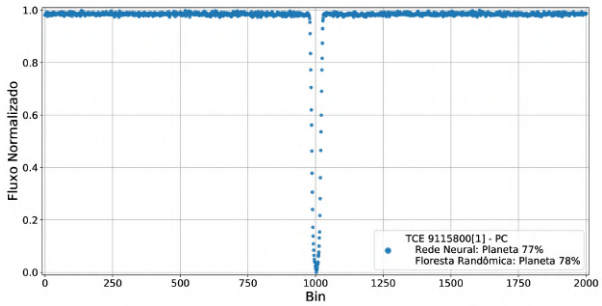
(31.e)



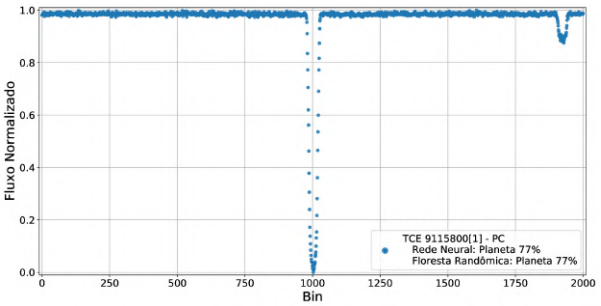
(31.f)



(31.g)



(31.h)



Fonte: Elaboração própria.

O outro motivo que justifica a criação de ambas as visualizações foi a verificação de trânsitos secundários mais distantes do trânsito principal nas curvas globais. As figuras 31.c e 31.d mostram um falso positivo do catálogo usado com e sem seu trânsito secundário, respectivamente. Esse é um exemplo propositalmente mais difícil para os modelos, pois o trânsito central se assemelha bastante com o formato de “U" de um trânsito gerado por um planeta. Ainda

assim, os modelos conseguiram o classificar como `não-planeta`. A RN tem certa vantagem em perceber essa nuance, com mais de 10% de assertividade em relação à FR. Ainda assim, ambos os modelos de alguma forma conseguiram dar um tanto mais de certeza em suas classificações quando o trânsito secundário foi removido.

As figuras 31.e e 31.f mostram um outro caso de eclipse binário, mas com ambas as quedas em um formato mais nítido de “V". Em ambos os casos, a FR classificou esse TCE como `não-planeta` com 95% de certeza, ao passo que o resultado da RN caiu de 95% para 93% ao classificá-lo como `não-planeta` após a remoção do trânsito secundário. Isso sugere que a RN é mais sensível em relação a esses trânsitos e que a FR dá mais ênfase no formato da curva do que no que há em seus arredores.

Por fim, as figuras 31.g e 31.h mostram a curva de um planeta que teve um trânsito secundário inserido. Em ambos os casos, a RN a classificou como `planeta` com 77% de certeza. A FR também a classificou como `planeta`, mas a porcentagem caiu de 78% para 77% com a modificação na curva. Isso mostra que a FR não é totalmente alheia a outros eventos associados ao TCE e que a sensibilidade da RN para com eles não é tão alta ao ponto de alterar sua classificação.

As tabelas 5 e 6 mostram as matrizes de confusão desses dois modelos. Por meio delas, consta-se que a RN tem um recall maior e, portanto, consegue detectar mais planetas. Além disso, a especificidade da FR é maior, o que implica que esse modelo consegue detectar melhor os casos negativos. As diferenças entre um modelo e outro não são tão significativas até agora a ponto de declarar um deles como o melhor. Apesar disso, a Rede Neural obteve seus resultados com poucos ajustes de hiperparâmetros comparado com a Floresta Randômica, o que sugere que o modelo tem potencial para alcançar resultados de performance ainda melhores.

Tabela 5 – Matriz de confusão do modelo de Rede Neural.

		Predito	
		Negativo	Positivo
Original	Negativo	1049	132
	Positivo	22	349

Fonte: Elaboração própria.

Tabela 6 – Matriz de confusão do modelo de Floresta Randômica.

		Predito	
		Negativo	Positivo
Original	Negativo	1092	89
	Positivo	52	319

Fonte: Elaboração própria.

4.4 Predições Feitas com os Melhores Modelos

Os modelos passaram por uma última etapa de teste, onde eles analisaram TCEs encontrados em sistemas multiplanetários selecionados e estudados por (SHALLUE; VANDERBURG, 2018) em seu trabalho. Tais sistemas possuem grande potencial para a descoberta de novos trânsitos planetários reais. A procura por candidatos é feita como descrito na seção 2.3 e no apêndice A. Após esse processo, as informações essenciais como `período` e t_0 do trânsito detectado são inseridas no pipeline de tratamento dos dados explicado na seção 3.1 e as curvas globais e locais são geradas. Essas novas buscas por trânsitos se diferenciam das que são feitas no pipeline padrão do Kepler pelo fato de serem mais lenientes com o limite imposto no SNR para que um evento seja considerado um TCE[29]. Isso é importante, pois os sinais de alguns planetas menores podem estar abaixo desse limite adotado por padrão e não serem descobertos (STUMPE *et al.*, 2012).

Os TCEs testados foram aqueles que o algoritmo do (SHALLUE; VANDERBURG, 2018) classificou como planetas com 80% ou mais de confiança[30]. Uma análise visual na figura 48 mostra que esses TCEs são candidatos plausíveis com o que vínhamos discutindo nos capítulos anteriores. Os resultados do artigo estão na tabela 7 e os resultados dos melhores modelos deste trabalho estão destacados em cinza.

Tabela 7 – Comparação dos melhores modelos com os do artigo de base.

KIC ID	KOI ID	Nome Kepler	Período (Dias)	SNR	Predição Shallue	Predição RN	Predição FR	Nota
11442793	351	Kepler-90	14,4	8,7	0,942	0,893	0,397	Kepler-90 i
8480285	691	Kepler-647	10,9	8,4	0,941	0,936	0,397	-
11568987	354	Kepler-534	27,1	9,8	0,920	0,848	0,530	-
4852528	500	Kepler-80	14,6	8,6	0,916	0,886	0,260	Kepler-80 g
5972334	191	Kepler-487	6,02	10,2	0,896	0,959	0,430	-
10337517	1165	Kepler-783	11,1	9,6	0,860	0,391	0,443	-
11030475	2248	-	4,75	8,7	0,858	0,649	0,317	-
4548011	4288	Kepler-1581	14,0	9,7	0,852	0,079	0,377	-
3832474	806	Kepler-30	29,5	14,1	0,847	0,014	0,227	-

Fonte: Adaptado de (SHALLUE; VANDERBURG, 2018). Resultados dos modelos autorais inseridos para comparação.

Os quatro primeiros TCEs foram inspecionados pelos autores com mais profundidade,

[29] Para os dados da missão Kepler, esse limite é de 7,1 (JENKINS *et al.*, 2002) e para os dados da missão K2 esse limite é de 9 (VANDERBURG *et al.*, 2016).

[30] Esse algoritmo consiste em uma Rede Neural Convolucional.

dada a confiança acima de 90% que o seu modelo apresentou. Dois deles foram validados como planetas reais, agora conhecidos como Kepler-90 i e Kepler-80 g. Nosso modelo de Rede Neural classificou esses quatro TCEs como `planeta` com uma confiança acima de 84%. A confiança do modelo para os planetas confirmados se deu acima 88%. No entanto, o modelo de Floresta Randômica só classificou um desses nove TCEs como `planeta`, com apenas 53% de confiança, e não conseguiu identificar os dois planetas confirmados. A performance exibida na tabela 4 da FR mostra que o modelo é bom, mas esses novos resultados deixam claro que a RN é bastante superior quando se trata de dados com um SNR baixo.

De fato, vimos nas análises da seção 4.3 que a FR é mais suscetível ao formato da curva. Em análises mais ruidosas, esse formato é bastante deturpado, o que pode explicar a baixa performance da FR neste último teste. Com isso, constatamos que o modelo de Rede Neural foi o modelo testado que melhor generalizou as suas predições, até em situações mais extremas.

5 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho, revisamos as principais características da coleta e tratamento de dados da missão Kepler para fins de detecção de exoplanetas. Em muitos casos, planetas pequenos como os tipo-Terra são facilmente perdidos pelo ruído inerente ao processo de observação de uma estrela pelo telescópio. Analisar potenciais eventos de interesse com um SNR baixo se faz necessário para a descoberta de planetas desse porte, porém a taxa de falsos positivos aumenta consideravelmente, inviabilizando a inspeção manual desses eventos por astrônomos.

Diante disso, revisamos também como algoritmos de Aprendizado de Máquina podem ajudar a otimizar essa busca em meio aos falsos positivos. Para esse fim, estudamos alguns dos algoritmos mais comuns de inteligência artificial usados em ciência de dados e os implementamos para classificar curvas de luz de TCEs previamente detectados. O processo de treino, validação e teste teve como base os dados do catálogo *Autovetter Planet Candidate Catalog for Q1-Q17 DR24*, onde seus TCEs foram analisados e classificados de antemão pelo algoritmo ***autovetter***.

O trabalho vai além de uma revisão da literatura, visto que todos os algoritmos abordados foram implementados independentemente e entregam resultados relevantes. O melhor modelo testado e apresentado aqui usa uma arquitetura de Rede Neural e ele se mostrou capaz de generalizar bem os seus resultados para além dos dados de validação e teste, como mostra a tabela 7, mesmo sem os seus hiperparâmetros devidamente ajustados. Com isso, todos os objetivos específicos propostos foram alcançados, salvo o que diz respeito ao aprimoramento de hiperparâmetros. Esse foi alcançado apenas de forma parcial por motivos de limitação computacional. Ainda assim, o objetivo geral do trabalho de criar um algoritmo para classificar TCEs como planetas ou não com os dados do Kepler foi alcançado.

Apesar disso, o trabalho já está tomando continuidade e seu desenvolvimento poderá ser encontrado no GitHub à medida que novas atualizações estáveis forem sendo implementadas. As perspectivas para esta nova etapa incluem:

- A criação de um banco de dados personalizado, com dados das missões Kepler, K2 e TESS com características semelhantes aos usados aqui;
- Testes com curvas de luz simuladas, onde poderemos ter um maior controle sobre os parâmetros astrofísicos presentes nas mesmas;
- A implementação de modelos mais sofisticados não apresentados aqui, como os que usam Rede Neural Convolucional, um algoritmo de Deep Learning amplamente usado em

processamento de imagens e também usado no artigo base deste trabalho;

– Em condições adequadas, um ajuste completo de hiperparâmetros do melhor modelo.

Além dos aprimoramentos citados, a principal perspectiva do trabalho para o futuro próximo é uma exploração massiva nos dados disponíveis das missões já finalizadas ou em andamento, em uma busca real por exoplanetas ainda não descobertos. Uma vez que o algoritmo detecte candidatos com uma boa confiança, análises astrofísicas mais avançadas para validação ou descarte desses candidatos, sejam estatísticas ou empíricas, deverão ser empregadas. Em caso de análises estatísticas, técnicas aprendidas ao longo da pesquisa para este trabalho serão fundamentais. Em caso de análises empíricas, trabalhos futuros estarão sujeitos a contribuições e colaborações de observatórios e outros centros de pesquisa.

No caso particular de missões em andamento e futuras, dados são ou serão disponibilizados continuamente. Eles podem contribuir ainda mais para a robustez dos modelos de machine learning, mas todo o processo de treinamento deveria ser repetido. Com base nisso, uma última perspectiva a ser avaliada é a utilização do que se conhece por *online machine learning* (FONTENLA-ROMERO *et al.*, 2013). Esse é um tipo de aprendizagem ativa, onde os modelos podem ser aprimorados com novos dados em tempo real. Desta forma, os aperfeiçoamentos computacionais e das técnicas de busca certamente trarão impactos positivos para o trabalho.

REFERÊNCIAS

ARMITAGE, P. **Astrophysics of Planet Formation**. [*S. l.*]: Cambridge University Press, 2010. ISBN 9780521887458.

ASTROPY COLLABORATION *et al.* **The Astropy Project**: Building an Open-science Project and Status of the v2.0 Core Package. The Astrophysical Journal, v. 156, n. 3, p. 123, 09 2018.

BATALHA, N. M. *et al.* **Pre-spectroscopic false-positive elimination of Kepler planet candidates**. The Astrophysical Journal, v. 713, n. 2, p. L103–L108, 03 2010.

BORUCKI, W. J. *et al.* **Kepler Planet-Detection Mission**: introduction and first results. Science, v. 327, n. 5968, p. 977–980, 2010.

BORUCKI, W. J.; SUMMERS, A. L. **The photometric method of detecting other planetary systems**. Icarus, v. 58, n. 1, p. 121–134, 1984. ISSN 0019-1035.

BOTTOU, L. **Stochastic gradient descent tricks**. In: Neural networks: tricks of the trade. [*S. l.*]: Springer, 2012. p. 421–436.

BREIMAN, L. **Random forests**. Machine learning, Springer, v. 45, n. 1, p. 5–32, 2001.

BRENNAN, P. **Discoveries Dashboard**. NASA, 2020. Disponível em: https://exoplanets.nasa.gov/discovery/discoveries-dashboard/. Acesso em: 29 nov. 2021.

BRYSON, S. T. *et al.* **Selecting pixels for Kepler downlink**. In: Software and Cyberinfrastructure for Astronomy. [*S. l.*]: SPIE, 2010. v. 7740, p. 526 – 537.

CAMPBELL, B.; WALKER, G. A. H.; YANG, S. **A Search for Substellar Companions to Solar-type Stars**. The Astrophysical Journal, v. 331, p. 902, ago. 1988.

CATANZARITE, J. H. **Autovetter Planet Candidate Catalog for Q1-Q17 Data Release 24**. Astronomy & Astrophysics, 2015.

CLAESEN, M.; MOOR, B. D. **Hyperparameter Search in Machine Learning**. 2015.

CLEVE, J. E. V.; CALDWELL, D. A. **Kepler Instrument Handbook**. 2016. 1 p. Kepler Science Document KSCI-19033-002.

CLEVE, J. E. V. *et al.* **That's How We Roll**: the NASA K2 mission science products and their performance metrics. Publications of the Astronomical Society of the Pacific, IOP Publishing, v. 128, n. 965, p. 075002, 06 2016.

COUGHLIN, J. *et al.* **Humans Need Not Apply**: robotization of Kepler Planet Candidate Vetting. In: American Astronomical Society Meeting Abstracts #225. [*S. l.*: *s. n.*], 2015. (American Astronomical Society Meeting Abstracts, v. 225), p. 202.05.

FONTENLA-ROMERO, Ó. *et al.* **Online machine learning**. In: Efficiency and Scalability Methods for Computational Intellect. [*S. l.*]: IGI Global, 2013. p. 27–54.

GÉRON, A. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow**: concepts, tools, and techniques to Build Intelligent Systems. [*S. l.*]: O'Reilly Media, 2019. ISBN 9781492032618.

GILLILAND, R. L. *et al.* **Initial characteristics of Kepler short cadence data**. The Astrophysical Journal Letters, v. 713, n. 2, p. L160, 2010.

GONZALEZ, G.; BROWNLEE, D.; WARD, P. **The Galactic Habitable Zone**: galactic chemical evolution. Icarus, v. 152, n. 1, p. 185–200, 2001. ISSN 0019-1035. Disponível em: https://www.sciencedirect.com/science/article/pii/S0019103501966175.

GREGORY, P. C.; LOREDO, T. J. **A New Method for the Detection of a Periodic Signal of Unknown Shape and Period**. The Astrophysical Journal, v. 398, p. 146, 10 1992.

GREICIUS, T. **Overlooked Treasure**: The First Evidence of Exoplanets. NASA, 2017. Disponível em: https://www.nasa.gov/feature/jpl/overlooked-treasure-the-first-evidence-of-exoplanets. Acesso em: 29 nov. 2021.

GRUBBS, F. E. **Procedures for Detecting Outlying Observations in Samples**. Technometrics, v. 11, n. 1, p. 1–21, 1969.

HARTMAN, J. D.; BAKOS, G. **VARTOOLS**: A Program for Analyzing Astronomical Time-Series Data. 2016.

HATZES, A. P. *et al.* **A Planetary Companion to γ Cephei A**. The Astrophysical Journal, American Astronomical Society, v. 599, n. 2, p. 1383–1394, 12 2003. ISSN 1538-4357.

HELLIER, C. **WASP Planets**. 2019. Disponível em: https://wasp-planets.net/about/. Acesso em: 29 nov. 2021.

HIPPKE, M.; ANGERHAUSEN, D. **Photometry's bright future**: detecting solar system analogs with future space telescopes. The Astrophysical Journal, American Astronomical Society, v. 810, n. 1, p. 29, 08 2015.

JENKINS, J. M.; CALDWELL, D. A.; BORUCKI, W. J. **Some Tests to Establish Confidence in Planets Discovered by Transit Photometry**. The Astrophysical Journal, American Astronomical Society, v. 564, n. 1, p. 495–507, 1 2002.

JENKINS, J. M. *et al.* **Initial characteristics of Kepler long cadence data for detecting transiting planets**. The Astrophysical Journal Letters, IOP Publishing, v. 713, n. 2, p. L120, 2010.

JENKINS, J. M. *et al.* **Overview of the Kepler science processing pipeline**. The Astrophysical Journal, American Astronomical Society, v. 713, n. 2, p. L87–L91, 03 2010.

JENKINS, J. M. *et al.* **Transiting planet search in the Kepler pipeline**. In: Software and Cyberinfrastructure for Astronomy. [*S. l.*: *s. n.*], 2010. v. 7740, p. 77400D.

JOHNSON, D. H. **Signal-to-noise ratio**. Scholarpedia, v. 1, n. 12, p. 2088, 2006.

KE, G. *et al.* **LightGBM**: A highly efficient gradient boosting decision tree. In: Advances in Neural Information Processing Systems. [*S. l.*]: Curran Associates, Inc., 2017. v. 30.

KOVÁCS, G.; ZUCKER, S.; MAZEH, T. **A box-fitting algorithm in the search for periodic transits**. Astronomy & Astrophysics, v. 391, n. 1, p. 369–377, 07 2002. ISSN 1432-0746.

KRAUSE, J.; PERER, A.; NG, K. **Interacting with predictions**: visual inspection of black-box machine learning models. In: Proceedings of the 2016 CHI conference on human factors in computing systems. [*S. l.*: *s. n.*], 2016. p. 5686–5697.

LIGHTKURVE COLLABORATION *et al.* **Lightkurve**: Kepler and TESS time series analysis in Python. 2018. Astrophysics Source Code Library.

LOMB, N. R. **Least-squares frequency analysis of unequally spaced data**. Astrophysics and space science, v. 39, n. 2, p. 447–462, 1976.

MANDEL, K.; AGOL, E. **Analytic Light Curves for Planetary Transit Searches**. The Astrophysical Journal, v. 580, n. 2, p. L171–L175, 12 2002.

MAYOR, M.; QUELOZ, D. **A Jupiter-mass companion to a solar-type star**. Nature, v. 378, n. 6555, p. 355–359, nov. 1995.

MCCAULIFF, S. D. *et al.* **Automatic classification of Kepler planetary transit candidates**. The Astrophysical Journal, v. 806, n. 1, p. 6, 06 2015.

MCCULLOCH, W. S.; PITTS, W. **A logical calculus of the ideas immanent in nervous activity**. The bulletin of mathematical biophysics, v. 5, n. 4, p. 115–133, 1943.

MITCHELL, T. **Machine Learning**. [*S. l.*]: McGraw-Hill, 1997. (McGraw-Hill International Editions). ISBN 9780071154673.

MONTGOMERY, J. **The AI revolution in science**: applications and new research directions. 2019. Disponível em: https://royalsociety.org/blog/2019/08/the-ai-revolution-in-science/. Acesso em: 30 nov. 2021.

MORRIS, R. L. *et al.* **Kepler Data Processing Handbook**: photometric analysis. Kepler Science Document KSCI-19081-003, p. 6, 2020.

O'MALLEY, T. *et al.* **KerasTuner**. 2019. Disponível em: https://github.com/keras-team/keras-tuner. Acesso em: 29 nov. 2021.

PASCUAL-GRANADO, J.; GARRIDO, R.; SUÁREZ, J. **MIARMA**: A minimal-loss information method for filling gaps in time series-application to corot light curves. Astronomy & Astrophysics, v. 575, p. A78, 2015.

PEDREGOSA, F. *et al.* **Scikit-learn**: Machine learning in Python. Journal of Machine Learning Research, v. 12, p. 2825–2830, 2011.

PRŠA, A.; ZWITTER, T. **A computational guide to physics of eclipsing binaries**. i. demonstrations and perspectives. The Astrophysical Journal, v. 628, n. 1, p. 426, 2005.

QUEMY, A. **Data Pipeline Selection and Optimization**. In: DOLAP. [*S. l.*: *s. n.*], 2019.

RASCHKA, S.; MIRJALILI, V. **Python Machine Learning**: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2, 3rd Edition. [*S. l.*]: Packt Publishing, 2019. 228 p. ISBN 9781789958294.

SAMUEL, A. L. **Some studies in machine learning using the game of checkers**. IBM Journal of research and development, v. 3, n. 3, p. 210–229, 1959.

SAVITZKY, A.; GOLAY, M. J. **Smoothing and differentiation of data by simplified least squares procedures**. Analytical chemistry, ACS Publications, v. 36, n. 8, p. 1627–1639, 1964.

SCARGLE, J. D. **Studies in astronomical time series analysis**. ii-statistical aspects of spectral analysis of unevenly spaced data. The Astrophysical Journal, v. 263, p. 835–853, 1982.

SCHMIDHUBER, J. **Deep Learning in Neural Networks**: An overview. CoRR, abs/1404.7828, 2014.

SHALLUE, C. J.; VANDERBURG, A. **Identifying Exoplanets with Deep Learning**: A five-planet resonant chain around kepler-80 and an eighth planet around kepler-90. The Astronomical Journal, American Astronomical Society, v. 155, n. 2, p. 94, 01 2018.

SHORTEN, C.; KHOSHGOFTAAR, T. M. **A survey on image data augmentation for deep learning**. Journal of Big Data, Springer, v. 6, n. 1, p. 1–48, 2019.

SING, D. K. **Stellar limb-darkening coefficients for CoRot and Kepler**. Astronomy & Astrophysics, EDP Sciences, v. 510, p. A21, 2010.

SOUTHWORTH, J. **Homogeneous studies of transiting extrasolar planets**–i. light-curve analyses. Monthly Notices of the Royal Astronomical Society, Blackwell Publishing Ltd Oxford, UK, v. 386, n. 3, p. 1644–1666, 2008.

SOUTHWORTH, J. **Homogeneous studies of transiting extrasolar** planets–ii. physical properties. Monthly Notices of the Royal Astronomical Society, Blackwell Publishing Ltd Oxford, UK, v. 394, n. 1, p. 272–294, 2009.

STUMPE, M. C. *et al.* **Kepler Presearch Data Conditioning** i—architecture and algorithms for error correction in kepler light curves. IOP Publishing, v. 124, n. 919, p. 985–999, 09 2012.

THOMPSON, S. E. *et al.* **Kepler Archive Manual**. [*S. l.*], 2016.

TWICKEN, J. D. *et al.* **Presearch data conditioning in the Kepler Science Operations Center pipeline**. In: Software and Cyberinfrastructure for Astronomy. [*S. l.*]: SPIE, 2010. v. 7740, p. 673 – 684.

VANDERBURG, A. *et al.* Planetary candidates from the first year of the k2 mission. The Astrophysical Journal Supplement Series, American Astronomical Society, v. 222, n. 1, p. 14, 1 2016.

WENZ, J.; POWELL, C. **The Lost Planets**: Peter van de Kamp and the Vanishing Exoplanets around Barnard's Star. [*S. l.*]: MIT Press, 2019. ISBN 9780262042864.

WOLSZCZAN, A.; FRAIL, D. A. **A planetary system around the millisecond pulsar PSR1257 + 12**. Nature, v. 355, n. 6356, p. 145–147, jan. 1992.

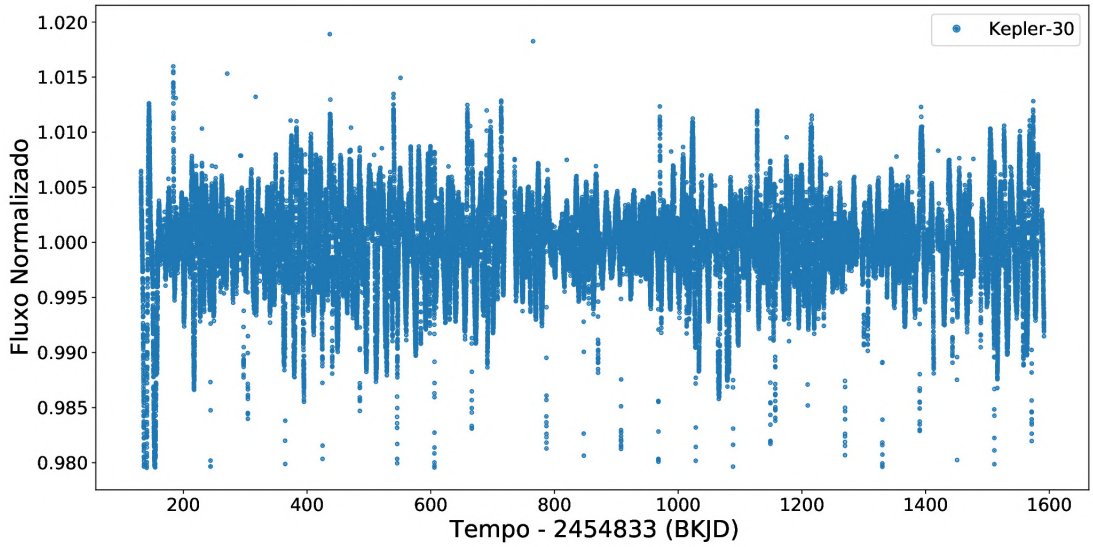
ZECHMEISTER, M.; KÜRSTER, M. **The generalised Lomb-Scargle periodogram**. Astronomy & Astrophysics, EDP Sciences, v. 496, n. 2, p. 577–584, 01 2009. ISSN 1432-0746.

APÊNDICE A – ENCONTRANDO SINAIS PERIÓDICOS EM SÉRIES TEMPORAIS

Para gerar as curvas de luz pelo tratamento descrito na seção 3.1, é preciso previamente fazer uma análise nas curvas de luz em busca de TCEs. Nela, devemos encontrar sinais periódicos que tenham características de trânsito. Isso implica que esse sinal deve indicar uma queda no fluxo de radiação fotométrica de uma estrela e ele, naturalmente, tem um `período`, uma `duração`, uma `profundidade` e um tempo de referência t_0. Existem muitos programas que já fazem isso de forma automatizada, como é o caso do VARTOOLS (HARTMAN; BAKOS, 2016) e do AstroPy (ASTROPY COLLABORATION *et al.*, 2018). Vamos aqui entender a ideia geral desse processo.

Tomemos como exemplo a curva de luz da estrela Kepler-30, tratada como explicado no capítulo 2 e representada pela figura 32.

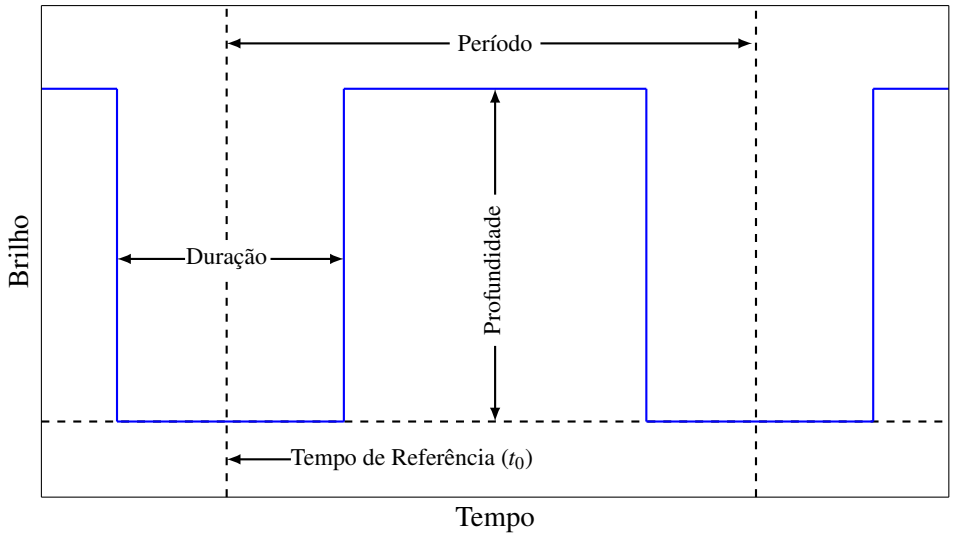
Figura 32 – Curva de luz tratada da estrela Kepler-30.



Fonte: Elaboração própria.

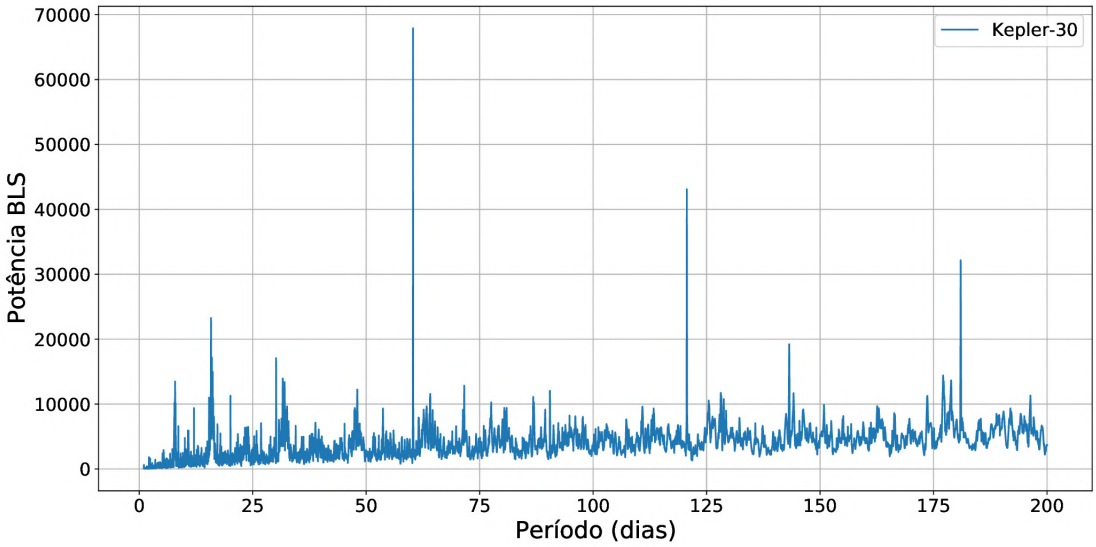
Para encontrar sinais periódicos em uma série temporal como essa, precisamos gerar um espectro da mesma, de forma que seja possível avaliar a o quão significante são os sinais em função das suas frequências ou períodos. Para isso, é comum aplicar uma transformada de Fourier, gerar um periodograma Lomb-Scargle (LOMB, 1976; SCARGLE, 1982; ZECHMEISTER; KÜRSTER, 2009) ou gerar esse espectro pelo método de *Box Least Squares* (BLS) (KOVÁCS *et al.*, 2002). No caso desse último, os sinais periódicos procurados na curva são aqueles podem ser aproximados por um formato retangular, como na figura 33.

Figura 33 – Modelo de trânsito procurado pelo método BLS.



Fonte: Elaboração própria.

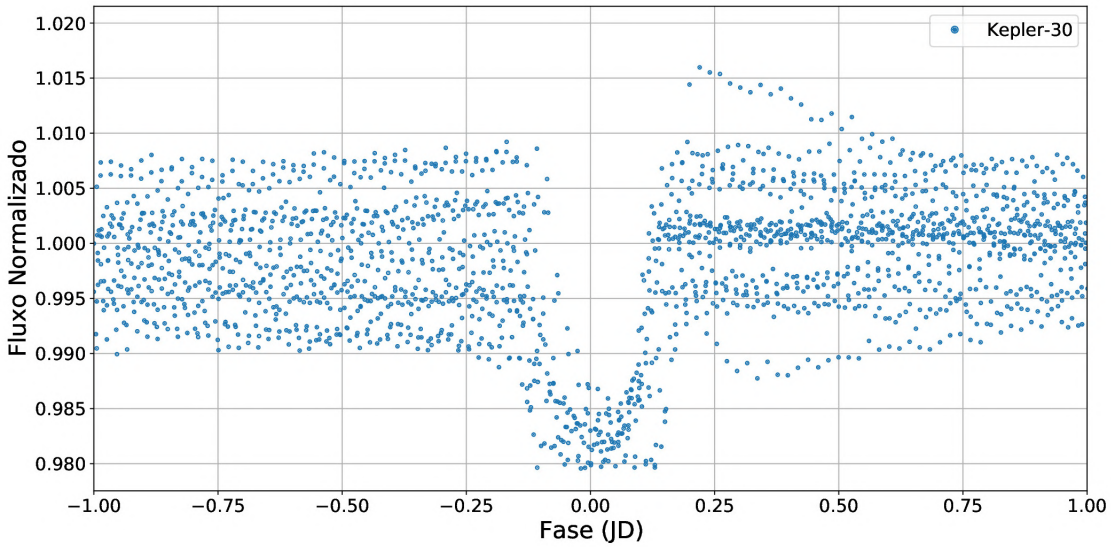
Figura 34 – Periodograma gerado para a estrela Kepler-30.



Fonte: Elaboração própria.

Há um pico muito significativo em torno de 60 dias. Mais precisamente, o pico se dá em 60,3 dias. Podemos usar esse valor para dobrar a curva de luz original em fase (como explicado na seção 3.1) e avaliar se o sinal é coerente com um trânsito planetário. A figura 35 mostra o resultado desse processo.

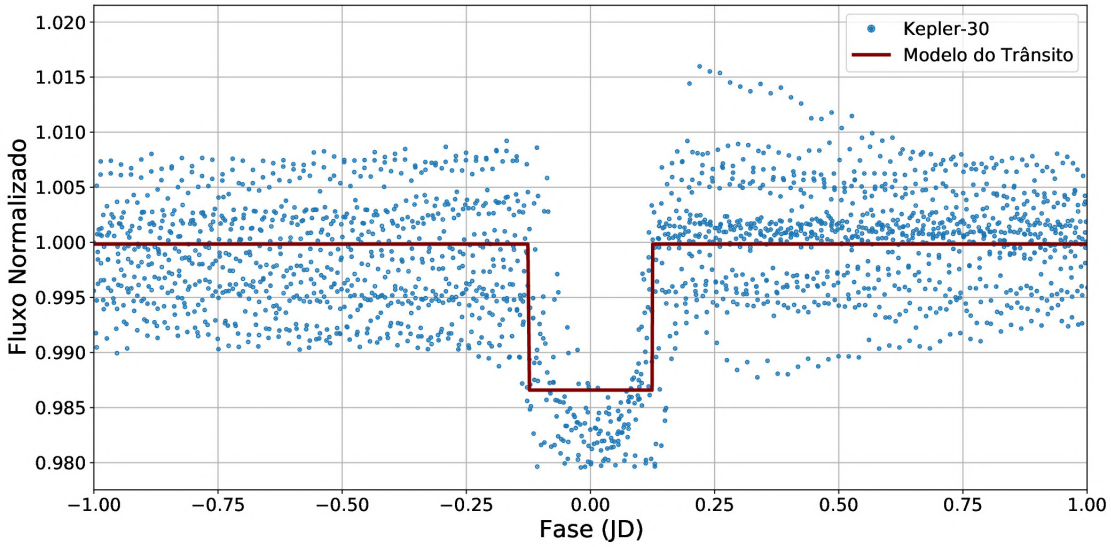
Figura 35 – Curva de luz da Kepler-30.



Fonte: Elaboração própria.

Esse, de fato, é um forte candidato a planeta. Com o BLS, podemos criar um modelo para esse trânsito. Grosso modo, ele ajusta os dados em linhas horizontais até o ponto em que os mesmos mudam abruptamente, como é o caso em uma queda no fluxo. Os dados do trânsito são ajustados separadamente. A figura 36 mostra o resultado desse ajuste.

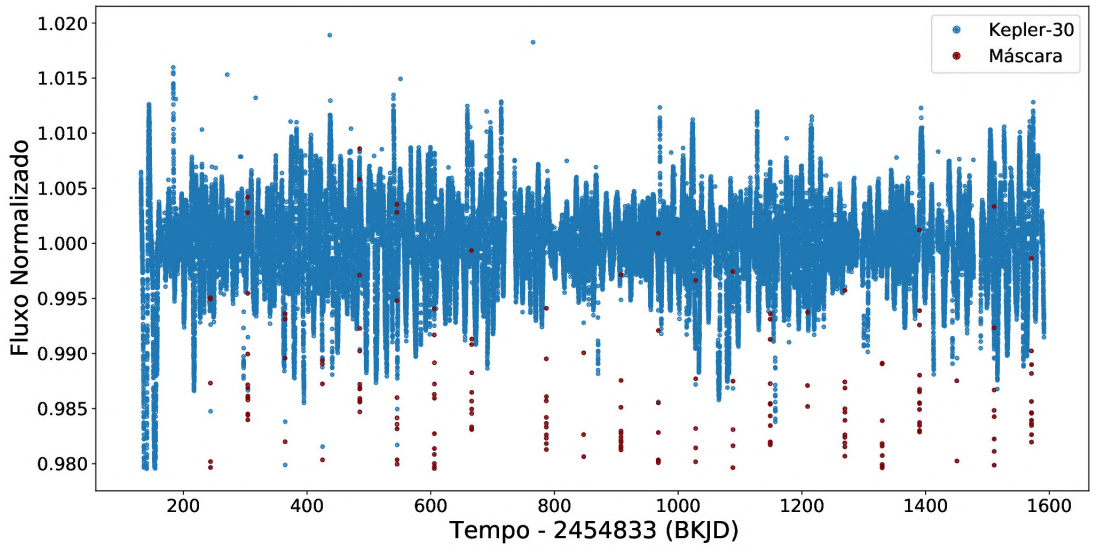
Figura 36 – Curva de luz da Kepler-30 com o modelo do trânsito.



Fonte: Elaboração própria.

Com base nesse modelo, conseguimos determinar que a `duração` do trânsito é de 0,25 dias. Combinando essa informação com o período encontrado na análise do gráfico 34, podemos demarcar todos os pontos na curva de luz original que representam o trânsito com uma “máscara", como exibido na figura 37.

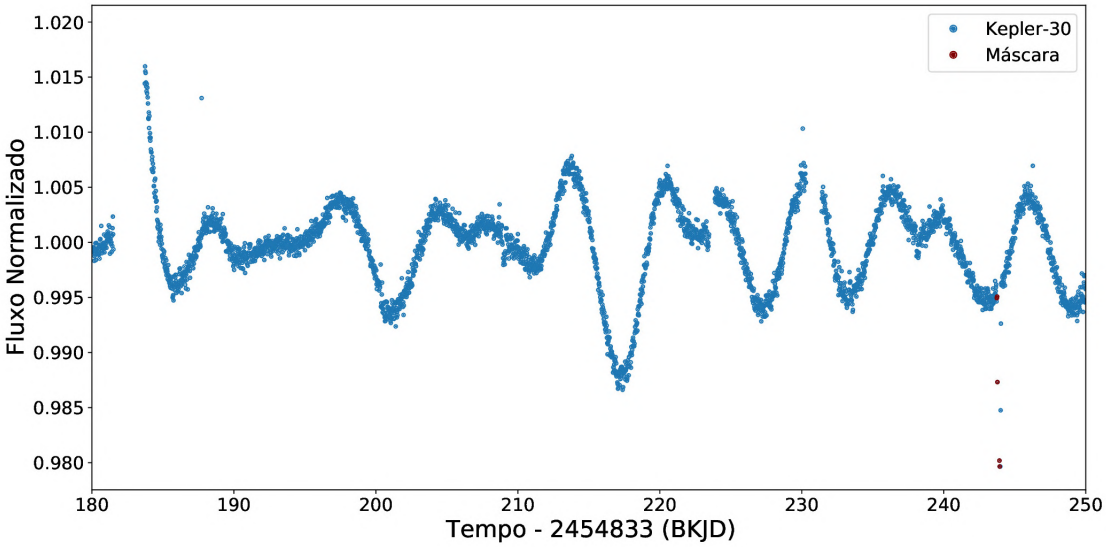
Figura 37 – Curva de luz da Kepler-30 com a máscara do trânsito.



Fonte: Elaboração própria.

À primeira vista, poderíamos dizer que t_0 se dá um pouco depois de 240 BKJD, mas essa análise deve considerar também o período do trânsito encontrado e o tempo do começo da curva no gráfico. Isso porque é possível que alguns dos trânsitos se percam em *gaps*, inclusive o primeiro, como é o que mostra a figura 38

Figura 38 – Começo da curva de luz da Kepler-30.



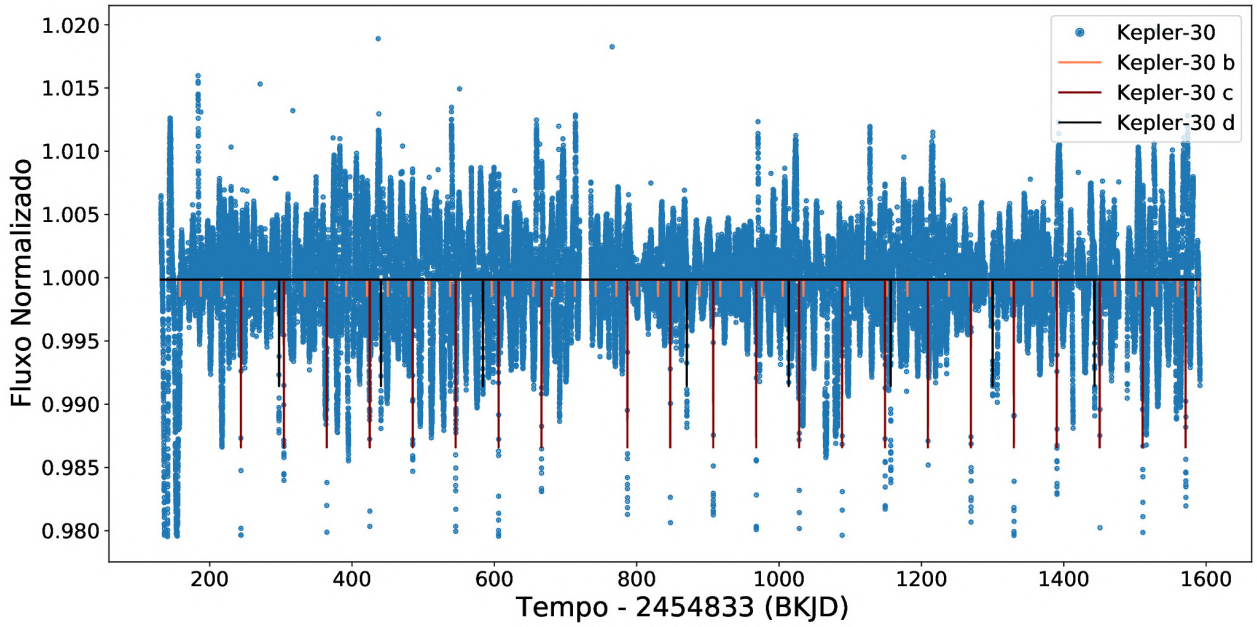
Fonte: Elaboração própria.

O t_0 neste caso está em 183.5 BKJD, precisamente dentro do gap observado mais à esquerda do gráfico. Com isso, obtemos todos os parâmetros necessários para gerar as curvas global e local deste TCE explanado. Ele, por sinal, se trata do planeta Kepler-30 c. Outros TCEs também podem ser encontrados na mesma curva da figura 32. Para isso, é necessário analisar os

demais sinais relevantes no gráfico da figura 34 ou gerar outros semelhantes, mas com intervalos de períodos diferentes.

Seguindo a mesma metodologia, é possível encontrar o planeta Kepler-30 d, cujo período é de 143,3 dias. Também é possível encontrar o planeta Kepler-30 b, porém seu sinal é muito fraco e, portanto, bastante propenso a ser confundido com ruído. Em situações como essa, uma lista de TCEs pode ser gerada como nos passos anteriores. A maioria deles seguramente não são sinais de interesse, mas todos podem ser analisados com machine learning, como foi feito na seção 4.4. O período do Kepler-30 b é de 29,3 dias. Os trânsitos destacados de todos os planetas podem ser vistos na figura 39.

Figura 39 – Curva de luz com os trânsitos dos planetas da Kepler-30 destacados.

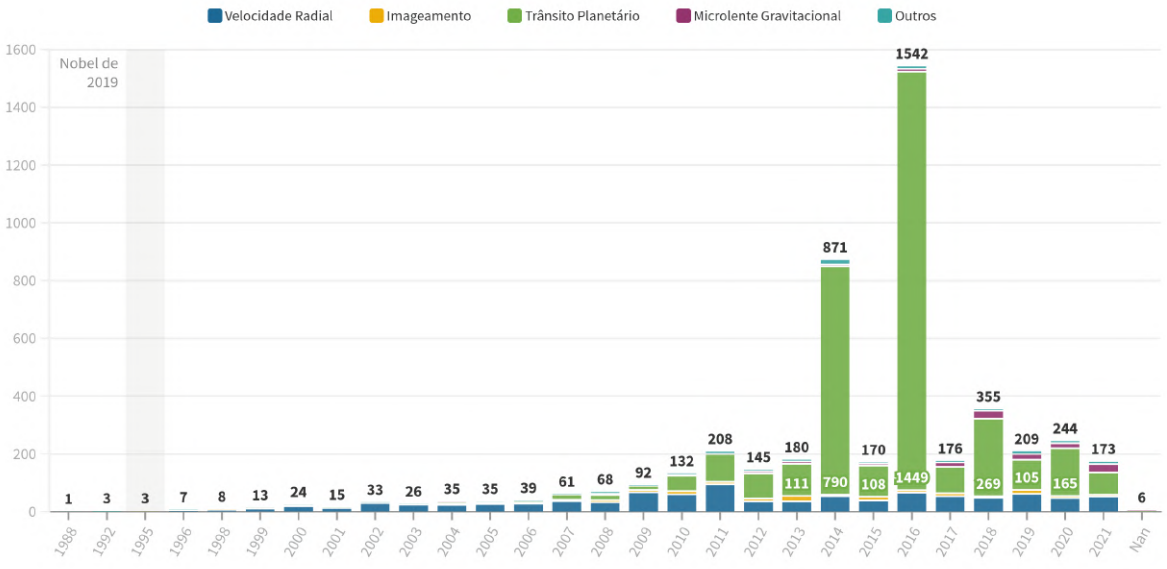


Fonte: Elaboração própria.

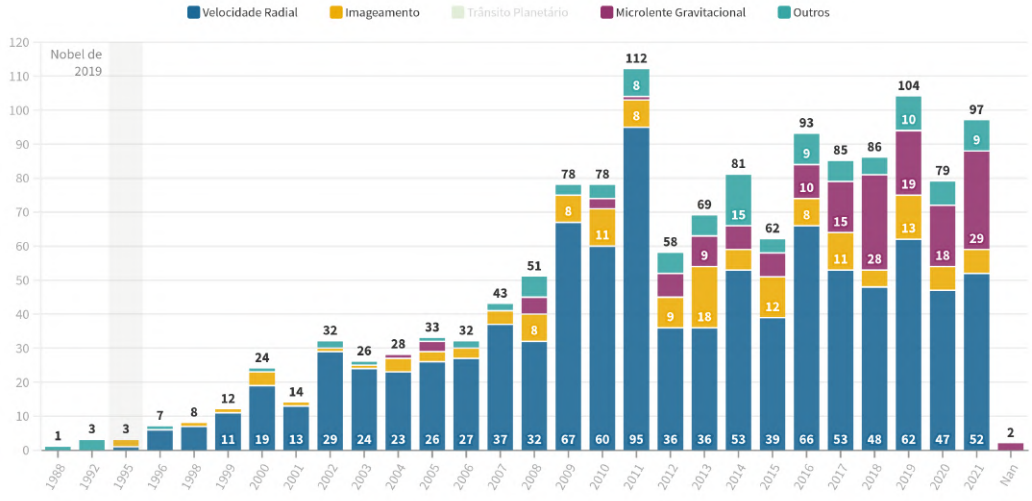
APÊNDICE B – FIGURAS AUXILIARES

Figura 40 – Número de exoplanetas descobertos ao longo dos anos e seus respectivos métodos de detecção. A contagem “Nan" no eixo horizontal representa os planetas cujo ano de descoberta não estava disponível no dataset da fonte.

(40.a) Dados gerais.

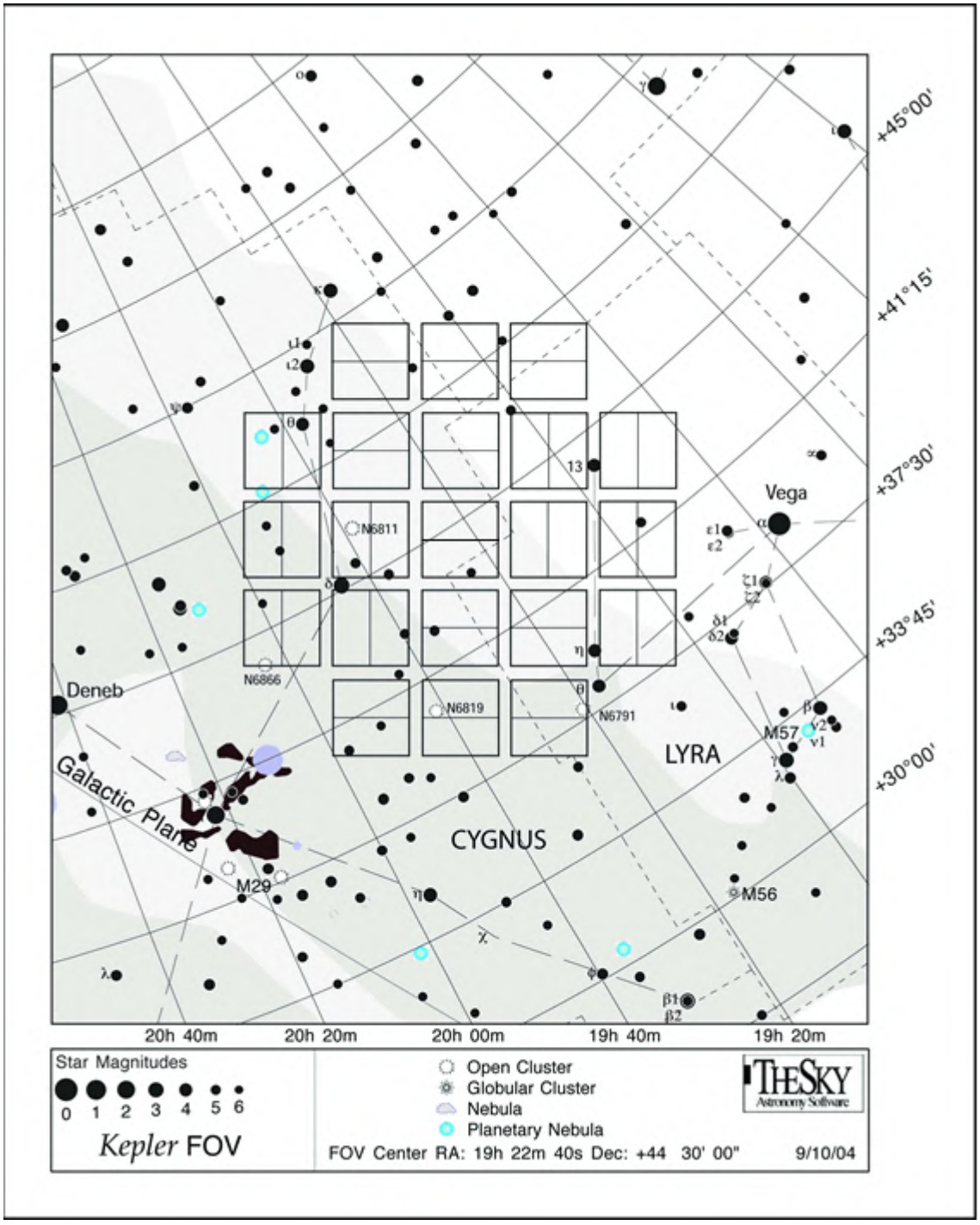


(40.b) Dados sem o método de trânsito planetário.



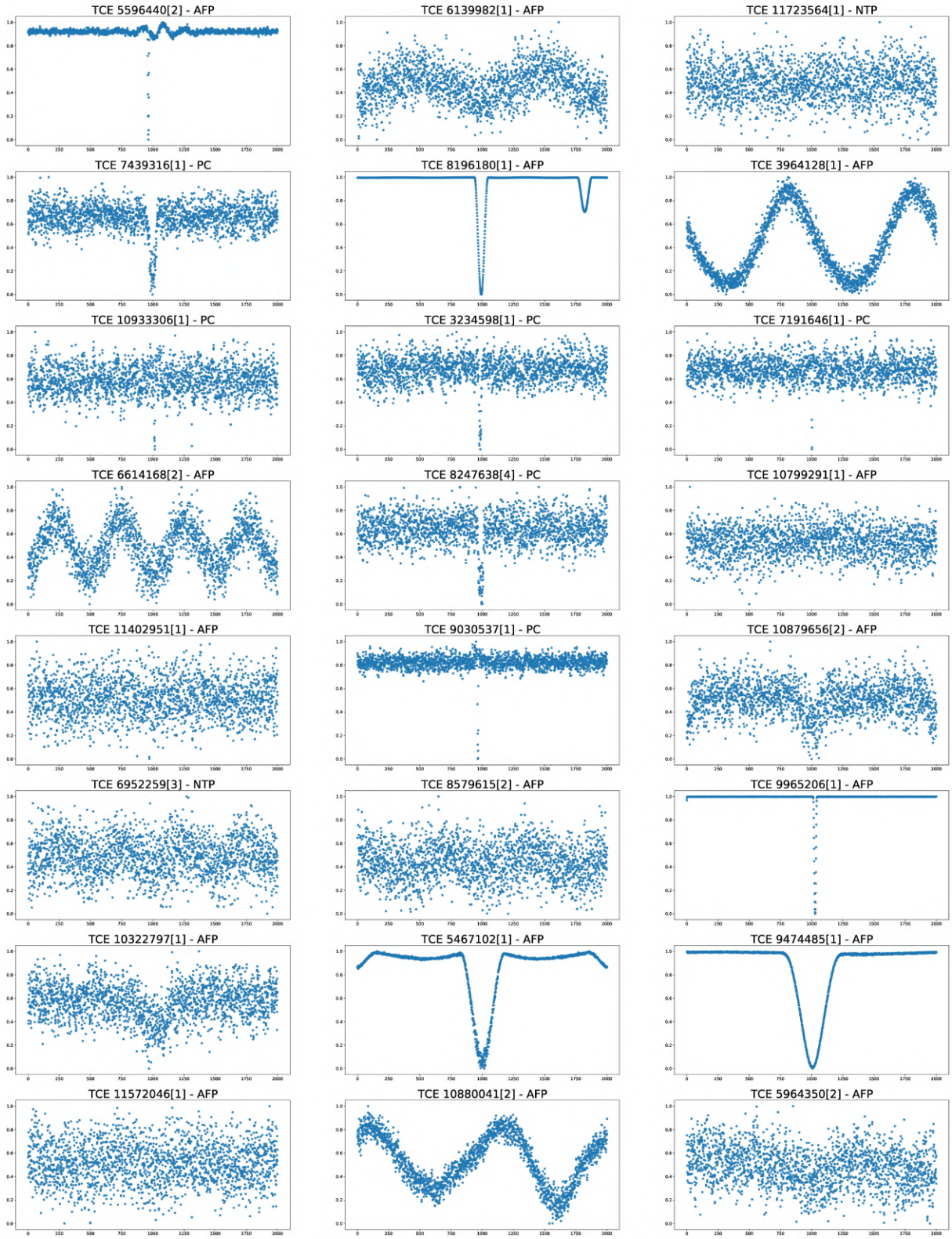
Fonte: Elaboração própria. Dados do *The Exoplanet Encyclopaedia*, disponível em http://exoplanet.eu. Acesso em 29 de novembro de 2021.

Figura 41 – Ilustração do campo de visão do Kepler.



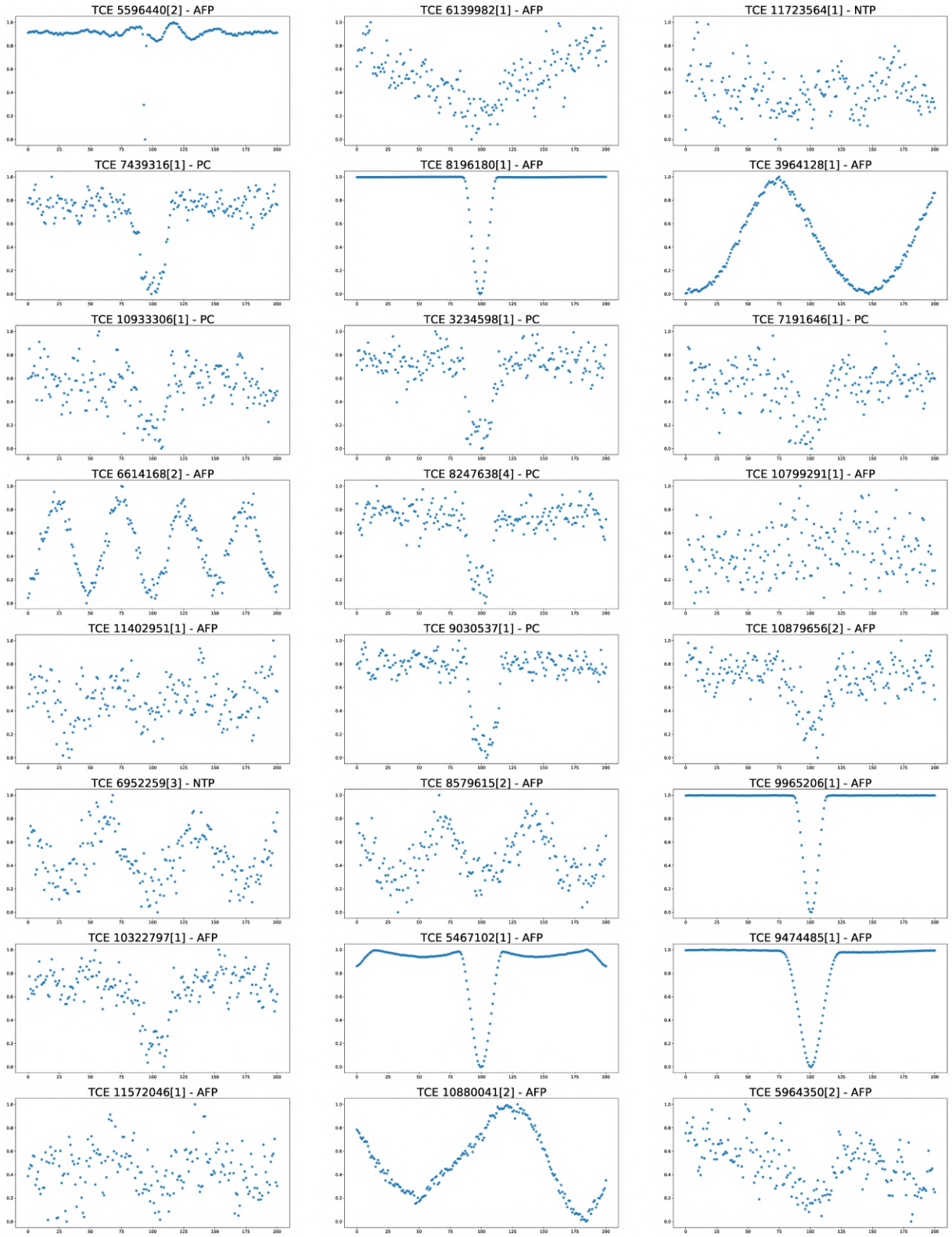
Fonte: MAST, disponível em https://archive.stsci.edu. Acesso em 30 de novembro de 2021.

Figura 42 – Exemplos de curvas globais geradas pelo pipeline desenvolvido.



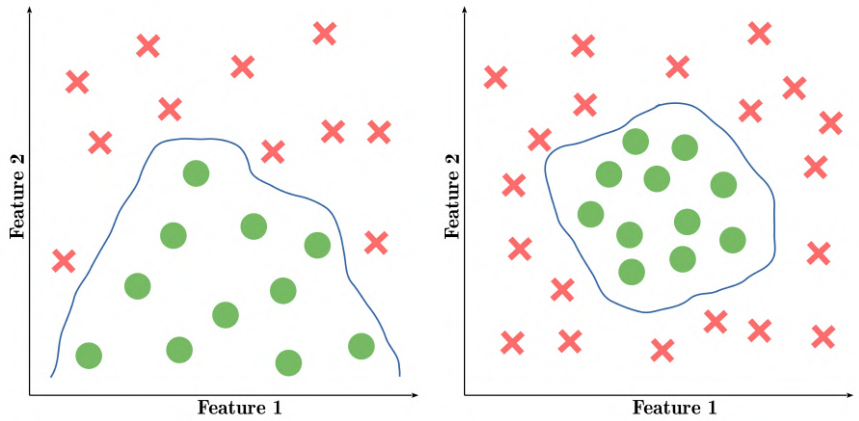
Fonte: Elaboração própria.

Figura 43 – Exemplos de curvas locais geradas pelo pipeline desenvolvido.



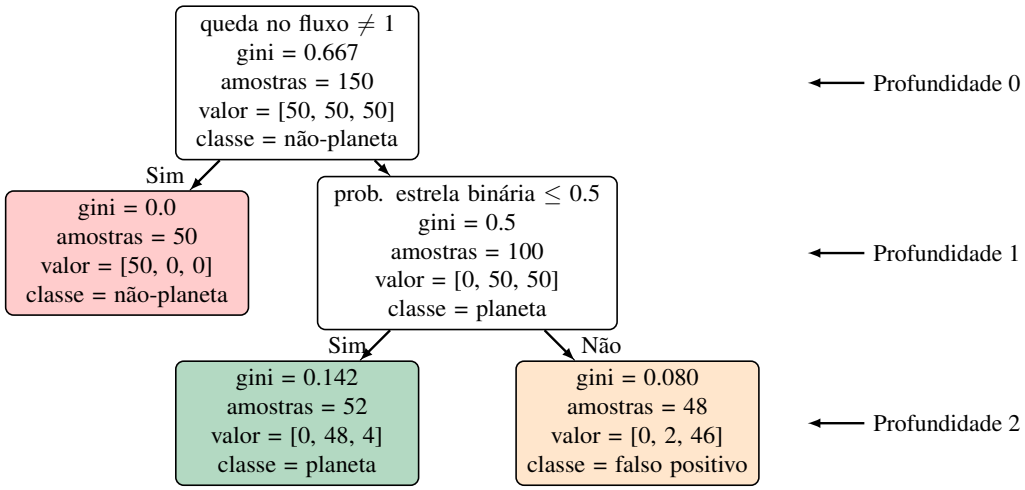
Fonte: Elaboração própria.

Figura 44 – Exemplos auxiliares de fronteiras de decisão para classificar os TCEs.



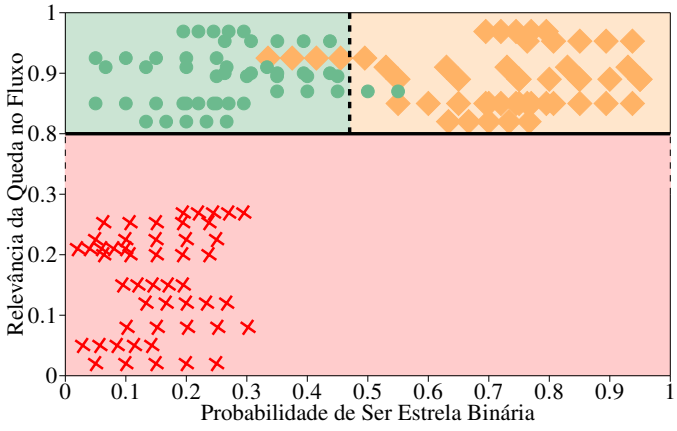
Fonte: Elaboração própria.

Figura 45 – Exemplo de árvore de decisão para multi-classificação.



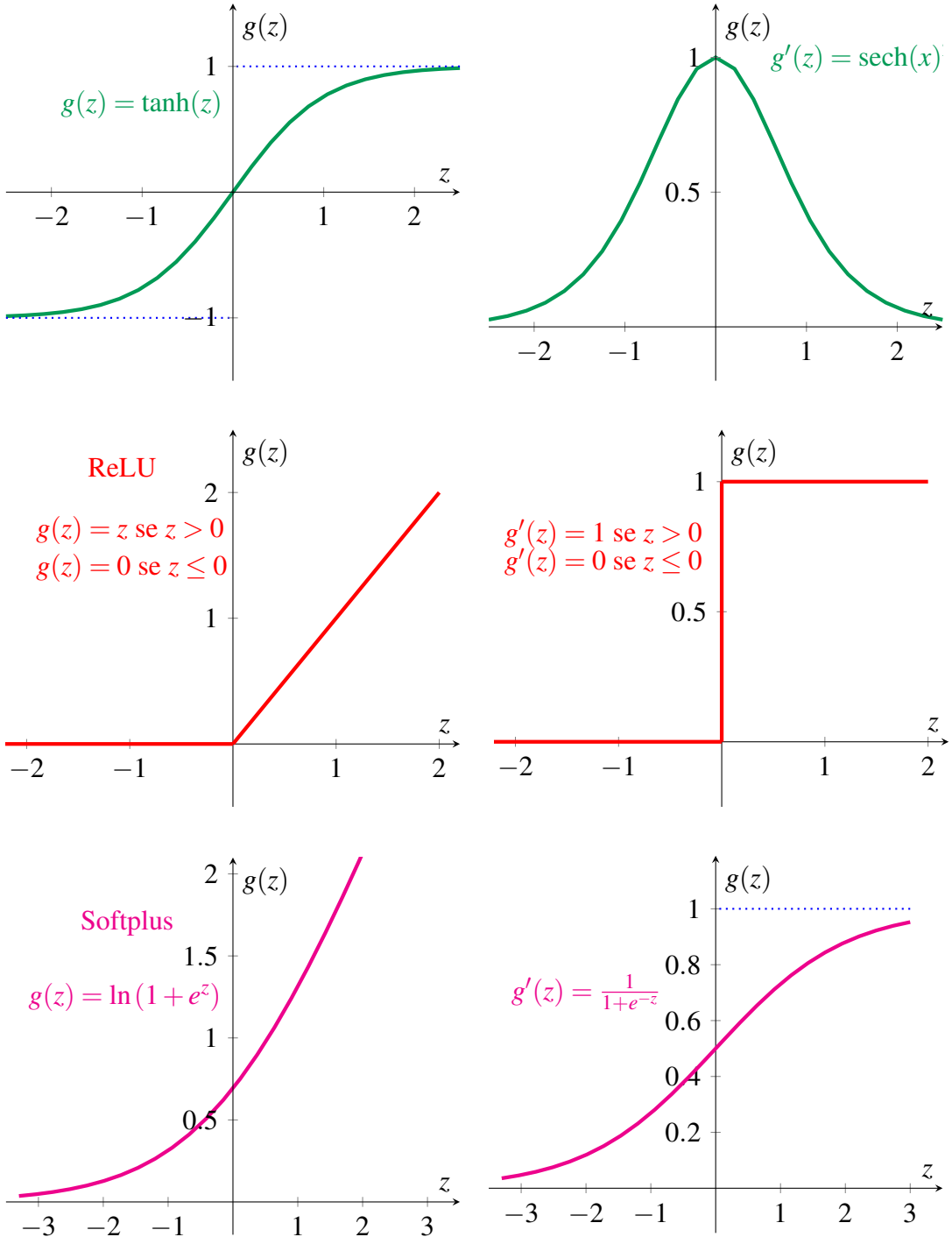
Fonte: Elaboração própria.

Figura 46 – Fronteiras de decisão geradas na multiclassificação da árvore de decisão do exemplo.



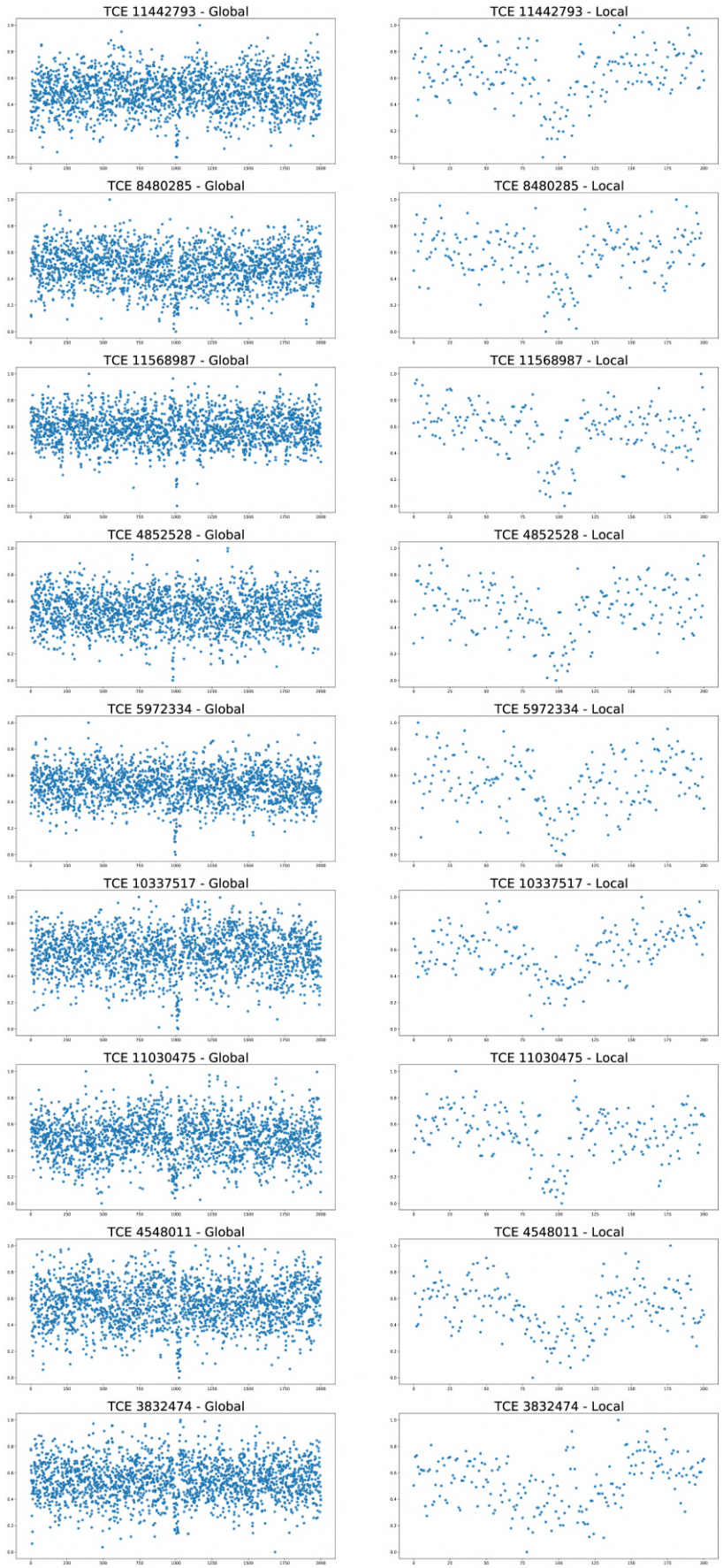
Fonte: Elaboração própria.

Figura 47 – Outras funções de ativação e suas derivadas.



Fonte: Elaboração própria.

Figura 48 – Curvas analisadas no último teste.



Fonte: Elaboração própria.