UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI
PROGRAMA DE PÓS-GRADUAÇÃO EM
CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Análise e comparação de algoritmos ensemble de classificação na descoberta de exoplanetas

Thiago Sales Freire Luz

Itajubá, 23 de outubro de 2023

UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI
PROGRAMA DE PÓS-GRADUAÇÃO EM
CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Thiago Sales Freire Luz

Análise e comparação de algoritmos ensemble de classificação na descoberta de exoplanetas

Dissertação submetida ao Programa de Pós-Graduação em Ciência e Tecnologia da Computação como parte dos requisitos para obtenção do Título de Mestre em Ciência e Tecnologia da Computação.

Área de Concentração: Inteligência Artificial

Orientador: Prof. Dr. Enio Roberto Ribeiro
Coorientador: Prof. Dr. Rodrigo Aparecido da Silva Braga

23 de outubro de 2023
Itajubá

Agradecimentos

Agradeço a Deus por me iluminar e me guiar nesta caminhada, me ajudando a superar os desafios encontrados.

Agradeço a minha esposa por sempre ter me apoiado na realização deste trabalho. Uma grande mulher diante de mim.

Agradeço meus pais e irmão por sempre estarem do meu lado me guiando e me ajudando enfrentar as dificuldades.

Agradeço ao meu orientador e co-orientador por me guiarem durante todo o mestrado. Sempre me orientaram e auxiliaram no desenvolvimento para chegar a este trabalho final.

Agradeço a todos os meus professores do POSCOMP que contribuíram para meu conhecimento.

"A Astronomia é grandiosa pelo que é e belíssima pelo que sabe fazer: fazer-nos maravilhar, também, a cada dia, a cada descoberta, com muitos dos seus mistérios... "

(Calvalcanti)

Resumo

Exoplanetas são planetas encontrados fora do sistema solar. A descoberta dos exoplanetas ocorre devido ao trabalho científico envolvendo o uso de telescópios, entre eles, o Kepler. Os dados coletados por este telescópio são chamados de *Kepler Object of Interest*. Para a tarefa de identificação de padrões nestes dados são utilizados algoritmos de Aprendizado de Máquina. Estes algoritmos são treinados para classificar estes dados em exoplanetas ou em falso-exoplaneta, isto é, falso-positivo. Dentre os algoritmos de classificação têm-se os denominados algoritmos Ensemble. Estes algoritmos combinam o desempenho de predição de dois ou mais algoritmos visando aperfeiçoar o desempenho preditivo final. Na literatura são utilizados algoritmos tradicionais em pesquisas relacionadas a detecção de exoplanetas. Constata-se, dessa forma, a carência de trabalhos que utilizam algoritmos Ensemble com este propósito. Esta dissertação realiza uma comparação de desempenho entre algoritmos Ensemble no processo de identificação de exoplanetas. Cada algoritmo é implementado com um conjunto de diferentes valores de parâmetros e executado várias vezes por um processo de validação cruzada. Uma matriz de confusão é gerada em cada execução, a qual é usada para análise das seguintes métricas de desempenho do algoritmo: exatidão, sensibilidade, especificidade, precisão e nota F1. Os algoritmos Ensemble atingiram um desempenho maior que 80% de acerto na maioria das métricas. Com a alteração dos valores dos parâmetros das funções observa-se um melhor resultado na predição. O algoritmo com o melhor desempenho foi o Stacking. Em síntese, verifica que os algoritmos Ensemble possuem um grande potencial para melhorar o resultado da predição de exoplanetas. O algoritmo Stacking se mostrou superior aos demais algoritmos e este aspecto é discutido no artigo. Os resultados desta dissertação indicam ser relevante aumentar o uso destes algoritmos, por possuírem um alto desempenho preditivo, favorecendo a detecção de exoplanetas.

Palavras-chave: Algoritmos Ensemble. KOI. Aprendizado de Máquina. Matriz de confusão. Stacking.

Abstract

Exoplanets are planets discovered outside our solar system. Their discovery happens because of scientific work with telescopes such as the Kepler. The data collected by Kepler is known as *Kepler Object of Interest*. Machine Learning algorithms are trained to classify these data into exoplanets or non-exoplanets. An Ensemble Algorithm is a type of Machine Learning technique that combines the prediction performance of two or more algorithms to gain an improved final prediction. The current works on exoplanet identification use mostly traditional non-Ensemble algorithms. Therefore, research that uses Ensemble algorithms for exoplanet identification is scarce. This paper performs a comparison among some Ensemble algorithms on the exoplanet identification process. Each algorithm is implemented with a set of different values for its parameters and executed multiple times. All executions are performed with the cross-validation method. A confusion matrix is created for each algorithm implementation. The results of each confusion matrix provided data to evaluate the following algorithm's performance metrics: accuracy, sensitivity, specificity, precision, and F1 score. The Ensemble algorithms achieved an average performance of more than 80% in all metrics. Changing the default values of the Ensemble algorithms parameters improved their predictive performance. The algorithm with the best performance is Stacking. In summary, the Ensemble algorithms have great potential to improve exoplanet prediction. The Stacking algorithm achieved a higher performance than the other algorithms. This aspect is discussed in the text. The results of this work show that it is reasonable to increase the use of Ensemble algorithms. The reason is their high prediction performance to improve exoplanet identification.

Key-words: Exoplanet detection. Ensemble algorithms. KOI. Machine Learning. Confusion matrix.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Exoplanetas são planetas orbitando outras estrelas similares ao Sol. A descoberta de exoplanetas tem sido um campo de pesquisa ativo nas últimas décadas e estimula o desenvolvimento de várias técnicas para detectar e caracterizar esses planetas distantes. O primeiro exoplaneta descoberto foi o 51 Pegasi B, em 1995, por Reid e Metchev (2008). A sua descoberta foi realizada por uma técnica que detecta a presença de um planeta através da velocidade radial entre ele e a sua estrela. Entre os métodos de detecção de exoplanetas tem-se: detecção por velocidade radial, influência da estrela hospedeira (astrometria) e método de trânsito. A técnica de velocidade radial detecta um planeta através da velocidade radial da estrela em relação à rotação do sistema planetário no seu centro de massa. Na astrometria foi descoberto que seria possível detectar o exoplaneta através da influência de sua estrela hospedeira em sua órbita e sua luminosidade. O método de trânsito detecta um exoplaneta através da identificação das variações periódicas na luz da estrela quando o planeta passa em frente a ela durante sua órbita (REID; METCHEV, 2008).

Com o objetivo de detectar exoplanetas, em uma região do espaço com mais de 200.000 estrelas, a NASA, em 2009, lançou o telescópio Kepler. Com os dados oriundos dele foram detectados mais de 1.000 possíveis exoplanetas utilizando o método de trânsito. Contudo, estes planetas passaram por análise para verificação se seriam de fato exoplanetas confirmados ou seriam dados falsos-positivo originados de diferentes fontes, como eclipse binário, eclipse binário de fundo, entre outros. O conjunto de dados desta descoberta foi armazenado em um banco de dados chamado de *Kepler Object of Interest* (KOI) (OFMAN et al., 2022).

No passado eram utilizados métodos manuais na descoberta de exoplanetas, mas com o avanço da Inteligência Artificial, essa descoberta é realizada através da utilização de algoritmos de Aprendizado de Máquina (PRIYADARSHINI; PURI, 2021). Estes algoritmos automatizam tarefas de análises em conjunto de dados, principalmente aqueles de grande porte e as fazem em menor tempo quando comparada à análise manual (MITCHELL, 1997).

A classificação supervisionada é uma técnica comumente utilizada de Aprendizado de Máquina, para análises em conjunto de dados, que consegue identificar e classificar dados de acordo com sua respectiva classe (SOOFI; AWAN, 2017). Existem diversos algoritmos de classificação e dentre estes temos os chamados algoritmos Ensemble. Os algoritmos Ensemble são abordagens que combinam as previsões de vários modelos de Aprendizado de Máquina para melhorar o desempenho e a robustez dos resultados (SAGI; ROKACH,

2018).

Essa abordagem tem sido amplamente utilizada em diversas áreas, como medicina, finanças e reconhecimento de padrões. No entanto, quando se trata da descoberta de exoplanetas, a aplicação de algoritmos Ensemble ainda é escassa. Várias fontes científicas apoiam essa observação.

Em um estudo realizado por Nigri e Arandjelovic (2017) sobre métodos de detecção de exoplanetas foram utilizados algoritmos não Ensemble e somente um Ensemble. O algoritmo Ensemble apresentou um melhor desempenho de predição quando comparado aos algoritmos não Ensemble, porém não foi feita nenhuma menção sobre o uso de algoritmos Ensemble como uma abordagem viável para detecção de exoplanetas. Além disso, uma revisão abrangente da literatura sobre Inteligência Artificial utilizada na Astronomia, conduzida por Fluke e Jacobs (2020), mencionou os cinco algoritmos mais utilizados na detecção de exoplanetas, mas entre eles não foram mencionados os algoritmos Ensemble.

Outro estudo relevante é o trabalho de Schanche et all (2019), que revisa e compara diferentes métodos de Aprendizado de Máquina aplicados à busca de exoplanetas. Embora forneça uma visão geral e abrangente das técnicas utilizadas, o artigo não mencionou algoritmos Ensemble como uma abordagem explorada na procura de exoplanetas.

A escassez de trabalhos utilizando algoritmos Ensemble na descoberta de exoplanetas é uma questão relevante, pois essa abordagem pode, potencialmente, melhorar a precisão e a confiabilidade dos resultados obtidos. Além disso, considerando a complexidade dos dados astronômicos, a combinação de múltiplos modelos, uma característica dos algoritmos Ensemble, pode ajudar a reduzir os desafios enfrentados pelos métodos tradicionais. Portanto, é evidente que há uma necessidade de pesquisas mais aprofundadas e estudos que explorem o uso de algoritmos Ensemble na descoberta de exoplanetas.

Recentemente, apesar da escassez mencionada, alguns trabalhos que utilizam algoritmos Ensemble tem mostrado que seu desempenho preditivo é superior ao dos algoritmos tradicionais de Aprendizado de Máquina quando usados em bases de dados de exoplanetas. Priyadarshini e Puri (2021) utilizam um modelo Ensemble de rede neural para unir a predição de vários algoritmos. Os autores demonstram que o algoritmo Ensemble tem um desempenho preditivo superior ao de outros algoritmos. De maneira similar Bhamare, Baral e Agarwal (2021) realizaram uma comparação com quatro algoritmos, dos quais dois eram algoritmos Ensemble. O resultado do artigo demonstrou que o algoritmo Ensemble Adaboost teve o melhor desempenho de acerto de exoplanetas.

Esta dissertação apresenta um estudo comparativo dos seguintes algoritmos Ensemble: Random Forest, Adaboost, Stacking, Random Subspace Method e Extremely Randomized Trees na classificação do conjunto de dados KOI. A comparação produz resultados que permitem identificar qual algoritmo possui o melhor desempenho para a

detecção de exoplanetas. Uma matriz de confusão com os resultados de cada execução dos algoritmos é criada com os acertos e erros dos modelos. Com base nestes resultados serão calculadas as seguintes métricas de desempenho dos algoritmos: exatidão, sensibilidade, especificidade, precisão e nota F1.

2 Revisão teórica

Esta seção faz uma revisão na literatura do conceito de aprendizado de máquina, do processo de classificação dos algoritmos Ensemble e do conjunto de dados de exoplanetas.

2.1 Aprendizado de máquina

O aprendizado de máquina existe para responder a duas perguntas correlacionadas: como é possível construir sistemas de computação que aprendam por meio de experiências? Quais são as leis fundamentais da estatística que governam todos os sistemas de aprendizado, incluindo computadores, humanos e organizações (MITCHELL, 1997)?

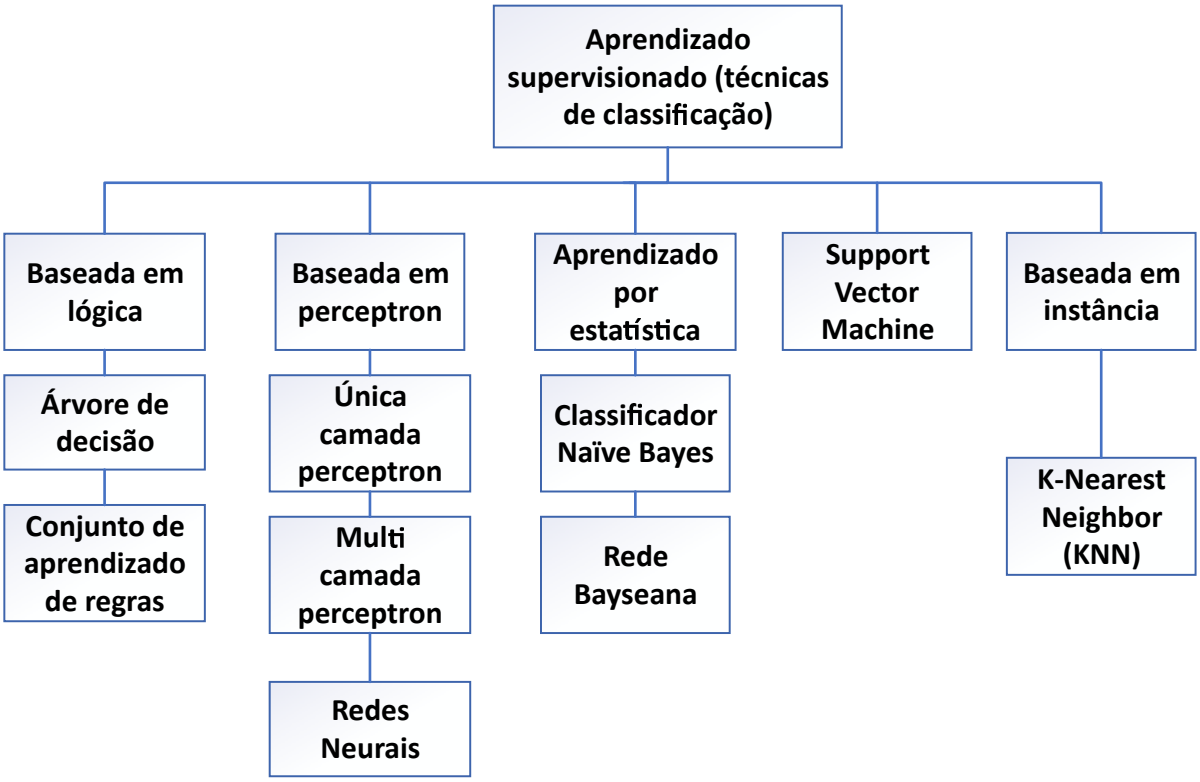
O aprendizado de máquina se provou capaz de resolver problemas de Ciência dos Dados, unindo o conceito de estatística, análise de dados e seus métodos para entender e analisar os dados. Antes de resolver problemas, deve-se categorizá-lo de modo a se utilizar o algoritmo de aprendizado de máquina mais apropriado. Qualquer problema em ciência de dados pode ser categorizado em cinco situações (ALZUBI; NAYYAR; KUMAR, 2018):

- **`De classificação:`** problema no qual o resultado da classificação poderá ser somente um número fixo de classes conhecidas anteriormente, como: sim ou não, verdadeiro ou falso, ou dependendo do número de classes poderá ser binário, ou de multi-classe.
- **`De detecção de anomalia:`** problema que analisa um certo padrão e detecta mudanças ou anomalias neste padrão. Por exemplo: empresas de cartão de crédito utilizam algoritmos de detecção de anomalia para encontrar desvios nos comportamentos de transação usual dos clientes e alertam sempre quando houver transações incomuns.
- **`De regressão:`** lidam com números contínuos e não discretos como a classificação. Como exemplo, temos um conjunto de dados com informação sobre altura e peso de várias pessoas. A regressão irá predizer, por exemplo, o peso médio para uma pessoa de determinada altura com base nos valores contínuo de peso.
- **`De agrupamento:`** problemas que lidam com o aprendizado mediante estruturas encontradas nos dados e agrupando as com base na semelhança encontrada.
- **`De aprendizado por reforço:`** a decisão é feita baseada em experiência de aprendizado.

O aprendizado de máquina pode ser divido em duas categorias: aprendizado não supervisionado e supervisionado. Aprendizado não supervisionado pode ser utilizado para predizer sobre um conjunto de dados constituído de dados de entrada sem identificação pré-definida, ou pode-se assumir que o resultado desejado não é considerado uma premissa. Aprendizado supervisionado tenta encontrar uma relação entre variáveis dependentes e um valor alvo (variável independente). Pode-se classificar aprendizado supervisionado em duas categorias: regressão que lida com dados numéricos contínuos e classificação com dados discretos (SOOFI; AWAN, 2017).

O modelo geral de aprendizado supervisionado pode ser representado conforme Figura 1 (SOOFI; AWAN, 2017).

Figura 1 – Técnicas de classificação de aprendizado de máquina.



Fonte - Soofi e Awan (2017).

Um dos problemas enfrentados durante a classificação é lidar com dados faltantes no conjunto de dados, ocasionando dificuldades durante a fase de treinamento e classificação do modelo. Algumas razões de potenciais dados faltantes seriam dados reconhecidos como irrelevantes durante a entrada, dados removidos devido ao desvio ou mau funcionamento do equipamento (SOOFI; AWAN, 2017).

2.2 Conjunto de Dados de exoplanetas

O Conjunto de dados utilizado nesta dissertação foi extraído do site eletrônico *Exoplanet Archive*, sendo um conjunto de ferramentas e banco de dados criados e disponibilizados pela NASA (AKESON et al., 2013).

O número de exoplanetas descobertos vem aumentando desde a primeira descoberta em 1995. Até o momento as técnicas predominantes de descoberta são: método de velocidade radial e método de trânsito. Os cientistas têm disponibilizado estes dados de exoplanetas descobertos para o público (AKESON et al., 2013). Estas técnicas são descritas a seguir:

- Método de velocidade radial: um planeta de massa M_p em uma órbita circular de raio α sobre uma estrela de Massa M_*, a estrela e o planeta irão ambos orbitar seus centros de massa, situado em uma distância conforme Equação (2.1) da estrela.

$$2\alpha M_p/(M_p + M_*) \tag{2.1}$$

  Equacionando as forças gravitacionais e centrípetas atuando na estrela, e assumindo que $M_* > M_p$, a velocidade máxima da estrela υ na linha de visão do observador poderá satisfazer a Equação (2.2), na qual i é a inclinação da órbita do planeta em relação ao observador. A velocidade radial pode determinar ambos $M_{\sin i}$ e também, da forma da variação de υ com o tempo e a excentricidade e da órbita do planeta (REID; METCHEV, 2008).

$$\upsilon = G(M_p \sin i)^2/2M_*a \tag{2.2}$$

- Método de trânsito: Os exoplanetas periodicamente podem orbitar suas estrelas e existe uma chance de que a inclinação de sua órbita em relação a sua estrela seja de 90°. Com isso, se o planeta for suficientemente grande, haverá uma diminuição na intensidade de luz da estrela, podendo esta ser utilizada para detectar a inclinação e o raio do planeta (REID; METCHEV, 2008).

O conjunto de dados de exoplanetas utilizado nesta dissertação, foi desenvolvido pelo Instituto de Ciência de Exoplanetas (NExScI) da NASA. Ele acumula informações de exoplanetas confirmados, planetas candidatos para análises futuras e dados falso-positivos, que se assemelham a exoplanetas, mas possuem origens em outras fontes, como exemplo, eclipses. O conjunto de dados está disponível ao público para pesquisas e também provê algumas ferramentas para análise de dados (AKESON et al., 2013). Ele pode ser encontrado e acessado em (CALTECH, 2023b) e está sendo constantemente atualizado. Nele é possível encontrar informações relevantes para a comunidade pesquisadora de exoplanetas,

como parâmetros estelares (posição, magnitude e temperaturas), parâmetros de exoplanetas (massa e parâmetros orbitais), e descoberta / categorização dos dados (curvas de velocidade radial, curvas de luz fotométrica e espectro)(AKESON et al., 2013).

O estudo da classificação de exoplanetas utiliza o conjunto de dados acumulativo denominado *Kepler Object of Interest* (KOI). O objetivo deste conjunto de dados acumulativo é fornecer os dados mais precisos, manter a informação estelar e planetária de todos os KOIs em um único lugar (CALTECH, 2023a).

O KOI como mostrado na Figura 2, contém 50 colunas. As colunas podem ser separadas em conjuntos, cada qual com sua definição, as quais são: identificação do dado, classificação do arquivo de exoplanetas, colunas de disposição do projeto, propriedades de trânsito do exoplaneta, eventos de ponto de encontro (TCE), parâmetros estelares, parâmetros de entrada do Kepler (KIC), entre outras (CALTECH, 2023a).

A coluna na qual contém a informação a ser predita no estudo é a "Exoplanet Archive Disposition", esta possui os seguintes valores:

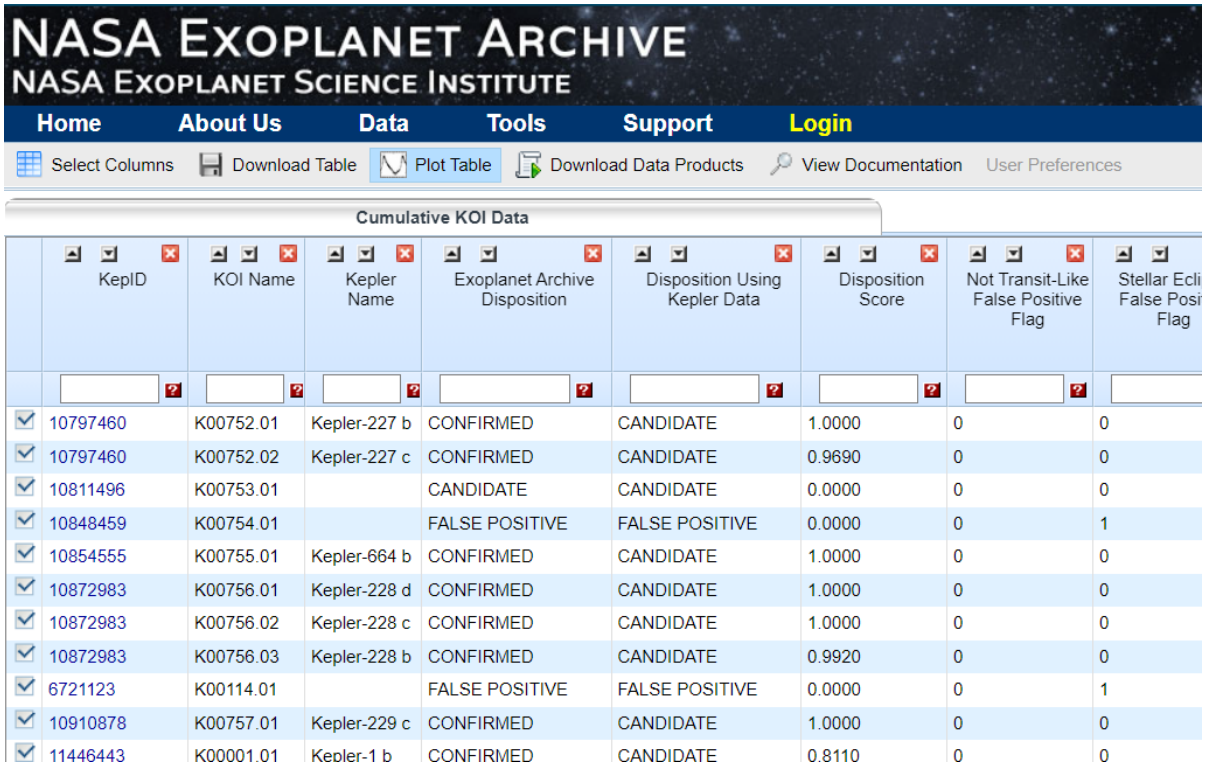
- Candidato: o dado é de um candidato que pode ser de outra fonte além de planeta, por exemplo, de um eclipse binário.
- Não disposto: dado não foi analisado e processado por especialistas.
- Confirmado: o dado de exoplaneta foi confirmado e também será listado na tabela do arquivo de exoplanetas confirmados.

A documentação completa e a explicação de cada coluna pode ser acessada em (CALTECH, 2023a) e para mais informações, consulte o anexo I.

O KOI inclui documentação exclusiva para explicar os conteúdos dos dados e detalhar como utilizar as ferramentas. Cada tabela interativa tem uma lista de todas as colunas com unidades e descrições. Cada ferramenta tem um guia do usuário com exemplos e capturas de tela e problemas mais conhecidos (CALTECH, 2023a).

Estes dados são avaliados por astrônomos com base na literatura atual (AKESON et al., 2013). A missão do Kepler visa procurar exoplanetas do tamanho da terra em mais de 150.000 estrelas hospedeiras utilizando o método de trânsito. O conjunto de dados do Kepler examina as curvas de luz das estrelas, com o intuito de identificar possíveis eventos de trânsito e realizar complexas modelagens. Cada evento de trânsito detectado com uma relação sinal/ruído maior que 7, constitui um evento de limite de cruzamento (*Threshold Cross Event* TCE). Os TCEs são mais estudados e caracterizados para identificar planetas candidatos, eclipse binários e falso positivos. Os eventos de trânsito ainda não examinados serão submetidos a observações e análises posteriores, para confirmar ou validar a situação planetária (AKESON et al., 2013).

Figura 2 – Conjunto de dados KOI.



NASA EXOPLANET ARCHIVE
NASA EXOPLANET SCIENCE INSTITUTE

Home | About Us | Data | Tools | Support | Login

Select Columns | Download Table | Plot Table | Download Data Products | View Documentation | User Preferences

Cumulative KOI Data

KepID	KOI Name	Kepler Name	Exoplanet Archive Disposition	Disposition Using Kepler Data	Disposition Score	Not Transit-Like False Positive Flag	Stellar Ecli False Posi Flag
10797460	K00752.01	Kepler-227 b	CONFIRMED	CANDIDATE	1.0000	0	0
10797460	K00752.02	Kepler-227 c	CONFIRMED	CANDIDATE	0.9690	0	0
10811496	K00753.01		CANDIDATE	CANDIDATE	0.0000	0	0
10848459	K00754.01		FALSE POSITIVE	FALSE POSITIVE	0.0000	0	1
10854555	K00755.01	Kepler-664 b	CONFIRMED	CANDIDATE	1.0000	0	0
10872983	K00756.01	Kepler-228 d	CONFIRMED	CANDIDATE	1.0000	0	0
10872983	K00756.02	Kepler-228 c	CONFIRMED	CANDIDATE	1.0000	0	0
10872983	K00756.03	Kepler-228 b	CONFIRMED	CANDIDATE	0.9920	0	0
6721123	K00114.01		FALSE POSITIVE	FALSE POSITIVE	0.0000	0	1
10910878	K00757.01	Kepler-229 c	CONFIRMED	CANDIDATE	1.0000	0	0
11446443	K00001.01	Kepler-1 b	CONFIRMED	CANDIDATE	0.8110	0	0

Fonte - Caltech (2023a).

2.3 Algoritmos de classificação Ensemble

O conceito de algoritmo Ensemble pode ser explicado através da interpretação da sabedoria da multidão, ilustrada por meio de uma história do matemático Francis Galton (1822 – 1911). Galton conduziu uma competição em uma feira livre, onde solicitou aos participantes que adivinhassem o peso de um machado. Ninguém adivinhou corretamente, contudo ele mostrou que a média do peso de todas as adivinhações estava próxima do peso exato do machado. Com este experimento, Galton mostrou que a combinação de diversas predições resultam em um resultado mais próximo do real (SAGI; ROKACH, 2018).

Os algoritmos de classificação utilizados no modelo Ensemble são denominados indutores, estes com o propósito de classificar os dados. Para unir o resultado dos algoritmos no processo de classificação é utilizado um sistema denominado votação balanceada, que visa escolher o resultado mais exato entre todos no conjunto de resultados gerados pelos indutores.

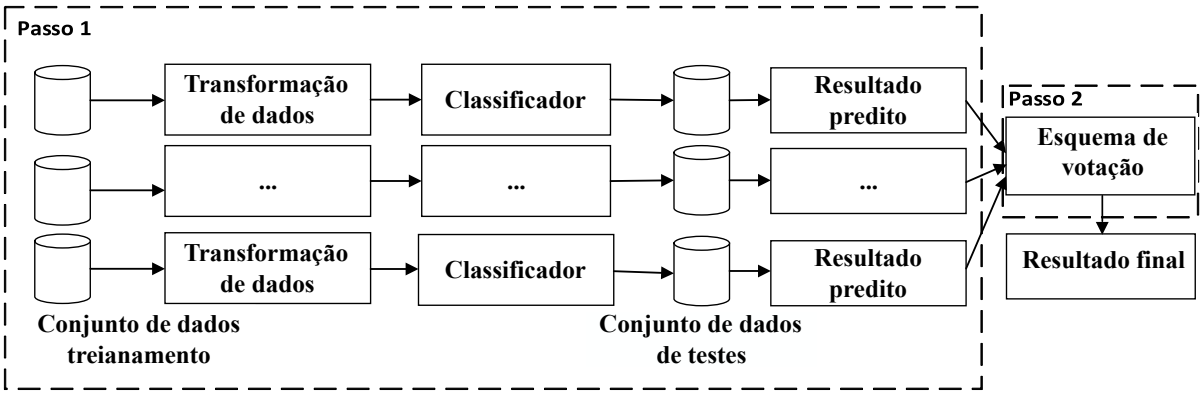
Um indutor utiliza dados de entrada pré-identificados e produz com estes um modelo que irá predizer com novos dados de entrada não identificados. Um indutor pode ser qualquer tipo de algoritmo de aprendizado de máquina, como exemplo: Árvore de Decisão, Rede Neural, Modelo de Regressão Linear, entre outros. O principal conceito de algoritmo Ensemble é combinar múltiplos algoritmos. Se um predizer um dado errado,

outros compensarão o resultado, portanto o desempenho da predição final do algoritmo Ensemble será melhor do que apenas de um único indutor (SAGI; ROKACH, 2018).

A Figura 3 ilustra o conceito de um modelo típico ensemble para classificação que consiste em dois passos (DONG et al., 2020):

1. Gerar resultados de classificação utilizando algoritmos de classificação.
2. Integrar múltiplos resultados de predição dos classificadores em uma única função, obtendo assim o resultado com um esquema de votação.

Figura 3 – Método ensemble para classificação.



Fonte - Dong et all (2020).

Os seguintes aspectos explicam porque algoritmos Ensemble aperfeiçoam o desempenho preditivo (SAGI; ROKACH, 2018):

• Evitam *overfitting* — quando apenas uma quantidade pequena de dados de treinamento está disponível, um algoritmo de aprendizado será propenso a encontrar diferentes hipóteses que predizem o dado alvo corretamente. Contudo, a predição com novos dados de teste de entrada, ainda desconhecidos, será realizada com menos precisão. Os algoritmos Ensemble combinam diferentes hipóteses, reduzindo o risco de escolher uma hipótese errada e melhorando o desempenho preditivo final.

• Vantagem computacional — indutores únicos que realizam buscas locais, podem ficar parados em um estado no qual não consigam avançar na predição. Os algoritmos Ensemble utilizam vários indutores que diminuem este risco.

• Representação — uma hipótese ótima provavelmente não será identificada com um único algoritmo, pois o espaço de busca é pequeno. Quando vários algoritmos são combinados, o espaço de busca aumenta e assim um melhor resultado para os dados poderá ser alcançado.

Existem diversos aspectos com dados que criam desafios para os algoritmos de Aprendizado de Máquina. Os métodos Ensemble podem ser utilizados para mitigar os seguintes desafios (SAGI; ROKACH, 2018):

1. Desbalanceamento de classe: alguns problemas de aprendizado de máquina possuem classes com diferentes números de dados, por exemplo, em um conjunto de dados com dez exoplanetas confirmados e mil candidatos, a proporção das classes está cem vezes maior para o padrão de planetas candidatos. Este desbalanceamento pode levar o modelo a desenvolver uma preferência por classes com maiores quantidade de dados.

2. Mudança de conceito: problemas de aprendizado de máquina, em tempo real, tem suas características e identificações sendo alteradas constantemente. Estas podem influenciar na predição final do algoritmo.

3. Curso da dimensionalidade: quanto mais características, ou seja, mais colunas em uma tabela de conjunto de dados existir, o espaço de busca do algoritmo irá aumentar exponencialmente. Portanto, a probabilidade de obter modelos que não podem ser generalizados irá aumentar (SAGI; ROKACH, 2018).

Durante o desenvolvimento de um algoritmo Ensemble é necessário considerar diversidade e desempenho preditivo. Diversidade significa utilizar vários modelos de indutores de modo a atingir um alto desempenho preditivo. Com este propósito, a inclusão de indutores no modelo Ensemble deve abordar os seguintes pontos (SAGI; ROKACH, 2018):

• Manipulação de entrada — cada indutor base é ajustado (*fitted*) usando subconjuntos diferentes dos dados, portanto muitos dados de entradas utilizam indutores diferentes.

• Algoritmos de aprendizado manipulado — nesta abordagem a utilização de cada indutor é alterada constantemente. Para isso, a forma que o indutor base busca a solução no espaço das hipóteses será manipulada. Consequentemente, o indutor terá vários caminhos para convergir. Como exemplo, ao construir um algoritmo Ensemble de Árvores de Decisão é possível inserir aleatoriedade nas árvores, selecionando no mínimo um, entre os diversos atributos da divisão de cada árvore.

• Particionamento — o conjunto de dados pode ser divido em subconjuntos menores, no qual cada um é utilizado no treinamento de um indutor específico. Logo, com a inclusão de diferentes subconjuntos para treinar os indutores é gerado uma diversidade em todo conjunto de dados.

• Hibridização ensemble — união de no mínimo duas estratégias durante a construção de um algoritmo Ensemble. Por exemplo, o algoritmo Random Forest, manipula

os dados na construção de suas árvores e também manipula o algoritmo de aprendizado de máquina, onde em cada nó das árvores, é escolhido aleatoriamente um subconjunto de colunas do conjunto de dados.

Com o propósito de integrar os resultados de predição dos indutores em apenas um único resultado, ou seja, a criação do modelo Ensemble, existe o processo denominado fusão de sáida. As abordagens mais utilizadas na literatura para esta finalidade, são as seguintes:

• Método de pesagem (*weighting method*): as saídas dos indutores podem ser unidas atribuindo-se pesos para os respectivos indutores. Este método é adequado em casos em que o desempenho destes indutores são semelhantes, podendo assim serem comparados. O método de pesagem mais simples e conhecido em classificação é denominado voto majoritário. E seleciona o indutor que obtiver a maioria dos votos. O conceito do método de somatório de distribuição nos algoritmos Ensemble, é a adição de um vetor de probabilidade condicional obtido de cada indutor. Assim, a classe selecionada será baseada no maior valor do vetor total segundo a Equação (2.3):

$$\hat{y} = argmax(\varphi(x_i)) \text{ onde } \varphi(x_i) = \sum_{k=1}^{K} f_k(x_i) \tag{2.3}$$

Na Equação (2.3), $\hat{y}$ representa a probabilidade de uma instância pertencer a uma determinada classe. A Equação $argmax()$ transforma um valor real em um valor no intervalo de 0 a 1. Utilizada para problemas de classificação, onde o resultado pode ser interpretado como uma probabilidade.

O parâmetro $\varphi(x_i)$ é uma combinação linear de funções de base, onde xi representa uma instância de dados e $\varphi(x_i)$ é o resultado dessa combinação linear. A combinação linear é calculada da seguinte maneira:

- K: representa o número de funções de base usadas na combinação.
- $\sum_{k=1} f_k(x_i)$: representa a soma das saídas das funções de base f_k aplicadas à instância de dados xi. No contexto de aprendizado de máquina, essas funções de base podem ser qualquer função matemática, como funções polinomiais, funções de base radial (RBFs), ou até mesmo funções mais complexas, dependendo do modelo.

• Método meta-aprendizado: se refere ao aprendizado de indutores. Estes modelos de aprendizado de máquina se diferem de um modelo comum por incluírem mais de uma etapa de aprendizado. O resultado do indutor individual serve como uma entrada, que será utilizada pelo modelo de meta-aprendizado, para gerar o resultado geral do modelo.

Os métodos Ensemble podem se diferenciar em relação à dependência dos algoritmos que fazem parte do modelo. Esses algoritmos podem ser dependentes ou independen-

tes. Ensemble dependente, a saída de cada indutor influencia a construção do próximo indutor. Neste conceito o conhecimento adquirido em iterações passadas irá guiar o processo de aprendizado da próxima geração. No Ensemble independente, cada indutor é construído independentemente da saída de outros indutores. Na Tabela 1, cada algoritmo Ensemble foi classificado com base nas categorias: método de fusão, dependência e abordagem de treinamento.

Tabela 1 – Categoria dos algoritmos Ensemble

Algoritmo Ensemble	Método de fusão	Dependência	Abordagem de treinamento
Adaboost	Pesagem	Dependente	Manipulação de entrada
Bagging	Pesagem	Independente	Manipulação de entrada
Random Forest	Pesagem	Independente	Hibridação Ensemble
Random Subspace Methods	Pesagem	Independente	Hibridação Ensemble
Gradient Boosting Machine	Pesagem	Dependente	Manipulação de saída
Error-correcting output codes	Pesagem	Independente	Manipulação de saída
Rotation Forest	Pesagem	Independente	Aprendizado manipulado
Extremely Randomized Trees	Pesagem	Independente	Particionamento
Stacking	Meta-aprendizado	Independente	Aprendizado manipulado

Fonte - Sagi e Rokach (2018).

Os algoritmos Ensemble escolhidos nesta dissertação para análise e comparação foram: Adaboost, Random Forest, Random Subspace Method, Extremely Randomized Trees e Stacking no qual na seção 3.1 será explicado o motivo. Estes algoritmos são descritos teoricamente a seguir.

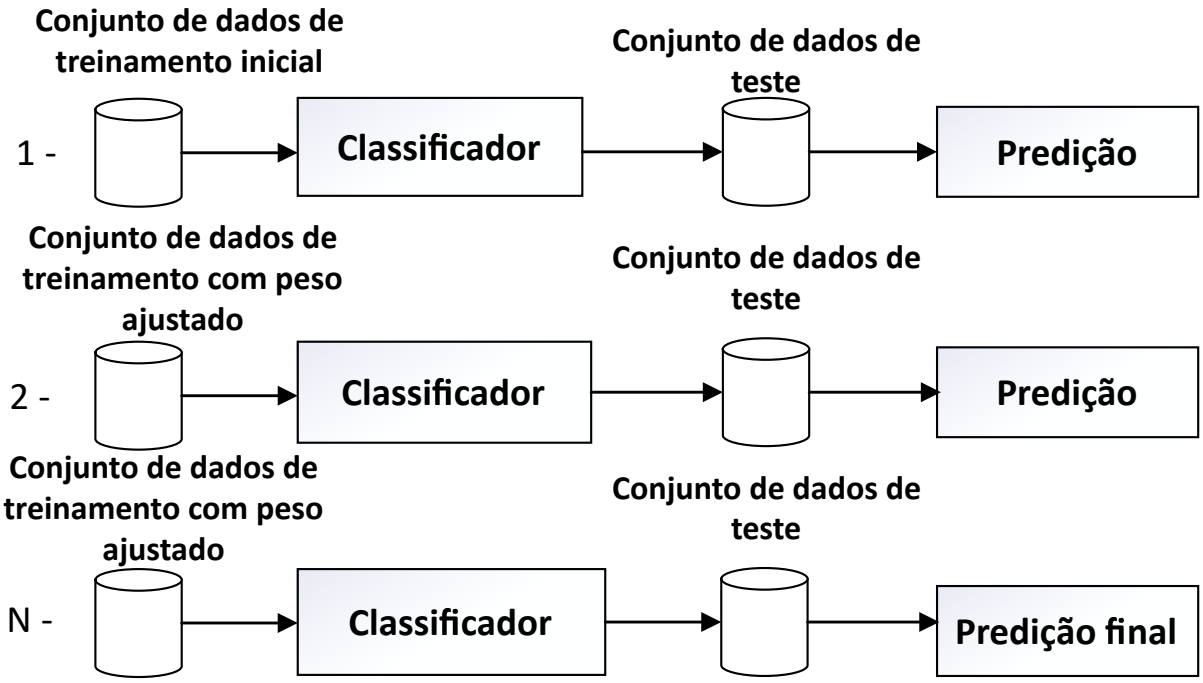
2.3.1 Adaboost

É o algoritmo de classificação mais conhecido com desenvolvimento mediante framework dependente. Deste modo, o resultado de cada predição de um indutor irá consequentemente influenciar a construção do próximo. O principal conceito do Adaboost é manter um foco maior nas instâncias classificadas anteriormente erradas, durante o treinamento de um novo indutor. Na primeira iteração do algoritmo é atribuído um valor de peso igual para toda instância. Após cada iteração, os pesos das instâncias classificadas erradas serão aumentados, mas os pesos nas instâncias classificadas corretamente, serão reduzidos. Ademais, pesos são atribuídos aos indutores de acordo com sua atuação preditiva geral (SAGI; ROKACH, 2018). O treinamento dos indutores no Adaboost é conduzido de uma maneira conjunta e não paralela (DONG et al., 2020).

Um único algoritmo de classificação pode não conseguir classificar corretamente a classe de um objeto, ou seja, predizer se uma instância do conjunto de dados será ou não exoplaneta. Contudo, quando são agrupados vários algoritmos de classificação também conhecidos como indutores ou classificadores fracos, estes progressivamente aprenderão com os objetos classificados erroneamente. Deste modo é possível construir um modelo

forte. O classificador fraco irá predizer melhor do que uma adivinhação aleatória, mas ainda terá um desempenho baixo para atribuir a classe correta para cada objeto. Como exemplo: um classificador pode predizer que todos acima da idade de 40 anos não podem correr uma maratona inteira, mas pessoas abaixo desta idade podem. Com essa informação a probabilidade de uma pessoa com idade maior que 40 correr uma maratona é pequena, ou seja, a porcentagem de acertos do classificador será alta. Contudo, ainda muitos dados serão classificados errados, devido à existência de pessoas acima de 40 anos que irão completar uma maratona e pessoas abaixo que não (KURAMA, 2020). Na Figura 4 pode-se ver o framework do Adaboost, no qual é alterado constantemente o peso no conjunto de dados para a classificação correta.

Figura 4 – Framework de Adaboost representando o processo de ajuste de peso para as instâncias do conjunto de dados.



Fonte - Dong et al (2020).

O algoritmo Adaboost utiliza em sua estrutura árvores de decisão "firmes", sendo chamadas de *decision stump*. O algoritmo *decision stump* utiliza apenas uma variável para realizar uma ação. As árvores do Adaboost se assemelham as árvores do algoritmo Random Forest, mas não crescidas totalmente, ou seja, possui apenas um nó e duas folhas. O algoritmo do Adaboost utiliza um conjunto de *decision stump* ao invés de árvores como Random Forest (KURAMA, 2020).

2.3.2 Etapas de funcionamento do Adaboost

A seguir serão apresentadas as etapas que descrevem o processo de classificação do algoritmo Adaboost.

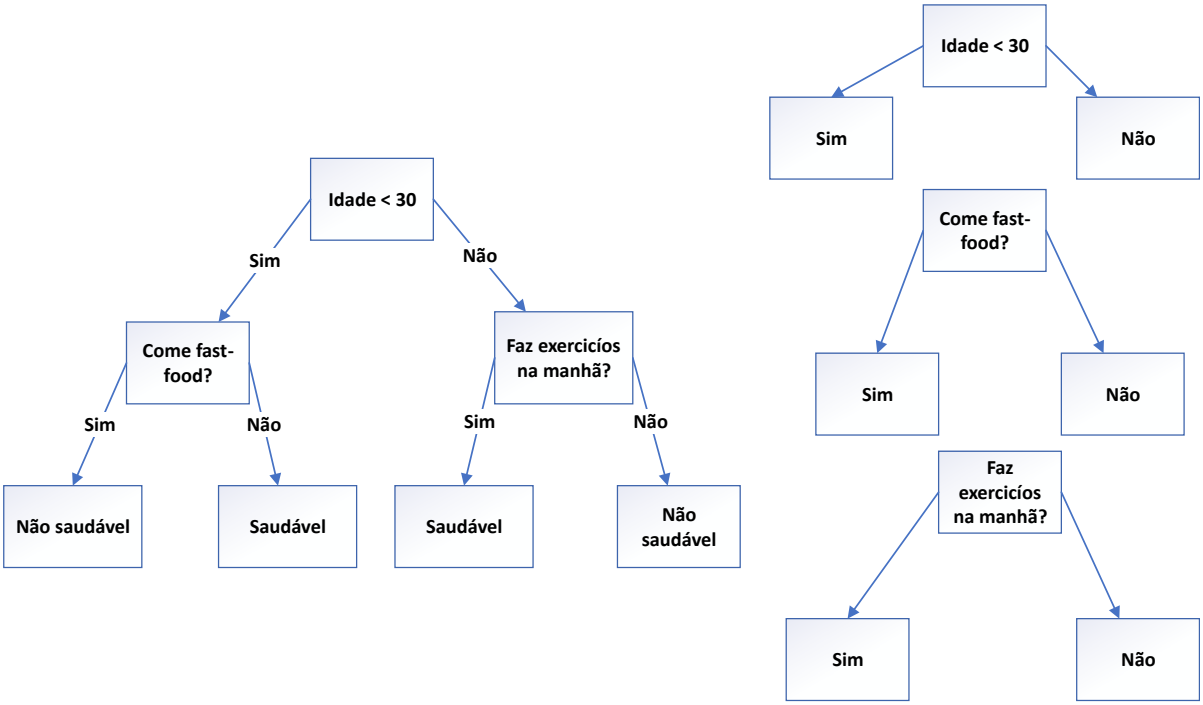
1: Um classificador fraco (Decision Stump) é construído sobre dados de treinamento utilizando as amostras (instâncias do conjunto de dados) com peso. O peso de cada amostra indicará a importância de ser corretamente classificada. Inicialmente, para o primeiro Decision Stump, são fornecidas amostras de pesos iguais.

2: São criados Decision Stumps para cada variável e estes tentam classificar as amostras para a classe correta. Por exemplo, na Figura 5, temos no lado esquerdo uma árvore de decisão já crescida e no lado direito um Decision Stump (apenas nó e folhas) que realiza a conferência por idade, se come fast-food e exercita-se. Serão validadas quais amostras classificaram correta ou incorretamente, como saudável ou não saudável cada stump individual.

3: Mais peso é atribuído para as amostras incorretamente classificadas, assim serão classificadas corretamente no próximo decision stump. Quanto maior a exatidão de um classificador, maior será seu peso.

4: Volta-se a etapa 2 até que todos os dados sejam corretamente classificados, ou o nível máximo de iterações seja atingido (KURAMA, 2020).

Figura 5 – Árvores de decisão x Decision Stumps.



Fonte - Kurama (2020).

2.3.3 O conceito matemático do Adaboost

Seja um conjunto de dados com N pontos ou linhas conforme a Equação (2.4):

$$x \in \mathbb{R}^n, y_i \in \{-1, 1\} \tag{2.4}$$

n é a dimensão de números reais ou o número de atributos em nosso conjunto de dados;

x é o conjunto de pontos de dados;

y é a variável alvo que poderá ser 1 ou -1, como um problema de classificação binário (exemplo: exoplaneta ou não).

Cada variável no conjunto de dados tem seu peso calculado. O Adaboost atribui um peso para as variáveis do conjunto de dados de treinamento a fim de determinar suas relevâncias no conjunto. Quando os pesos atribuídos forem grandes, estas variáveis também caracterizadas como um conjunto de pontos de dados no conjunto de dados treinamento, terão provavelmente maior valor. Da mesma maneira, quando os pesos forem pequenos, estes irão possuir influência miníma no conjunto de dados de treinamento.

Inicialmente, todos os pontos de dados terão o mesmo valor de peso w conforme a Equação (2.5):

$$w = 1/N \in [0, 1] \tag{2.5}$$

N é o número total de pontos de dados.

As amostras com peso sempre terão soma até 1, assim o valor do peso de cada indivíduo sempre estará entre 0 e 1. Após, calcula-se a influência real para este classificador na classificação dos pontos de dados usando a Equação (2.6):

$$\alpha = \frac{1}{2} \ln \frac{(1 - ErroTotal)}{ErroTotal} \tag{2.6}$$

Alfa (α) representa a quantidade de influência que o classificador terá na classificação final. O erro total é o número de classificações incorretas para o conjunto de treinamento divido pelo tamanho total do conjunto de treinamento.

Quando um Decision Stump não tem classificações erradas (um perfeito stump) resulta em uma taxa de erro de 0 e um grande e positivo valor alfa.

Se o Decision Stump classificar metade corretamente e metade incorretamente (uma taxa de erro de 0,5, não melhor que uma adivinhação aleatória), o valor de alfa será 0. Finalmente, quando o Decison Stump não parar de fornecer resultados classificados incorretamente, o valor de alfa será um valor alto negativo.

Após atualizar os valores atuais do erro total para cada stump, as amostras de peso que foram inicialmente tomadas como 1/N para cada ponto de dados são também atualizadas. Utiliza-se a Equação (2.7) para alteração do peso:

$$w_i = w_{i-1} * e^{\pm\alpha} \tag{2.7}$$

O novo peso da amostra será igual ao peso da amostra antiga multiplicado pelo número de Euler, elevado a mais ou menos alfa (KURAMA, 2020).

O Algoritmo 1 a seguir representa o pseudocódigo do Adaboost:

Algoritmo 1: O pseudocódigo do Adaboost

Inicialmente set uniform exemplos de peso
For each base learner do:
Train base learner with uma amostra com peso;
Test base learner on all dados;
Set o peso do learner with um valor destinado como sendo o peso da varável classificada incorretamente;
Set exemplos de pesos baseados on predições ensemble;
End for

Para implementação do Adaboost em linguagem Python é utilizado a biblioteca *scikit-learn* que disponibiliza ferramentas simples e eficientes para análise preditiva de dados, acessível a todos (SCIKIT-LEARN, 2023). Na biblioteca, temos o módulo *sklearn.ensemble* que possui o algoritmo Adaboost nomeado de AdaBoostClassifier.

O algoritmo Adaboost possui cinco parâmetros, para a avaliação do seu desempenho na predição dos dados de exoplanetas:

n_estimators: inteiro, valor default igual a 50. O número máximo de estimadores (algoritmos) no qual o boosting será finalizado. No caso de haver um ajuste (*fit*) perfeito, o processo de aprendizado irá parar antecipadamente. Os valores devem estar na faixa [1, infinito].

learning_rate: float, valor default igual a 1,0. Peso atribuído a cada classificador a cada iteração de boosting. Uma alta taxa aprendizado contribui com cada classificador. Existe um trade-off entre os parâmetros n_estimators e learning_rate. Os valores devem estar no range [0,0, infinito].

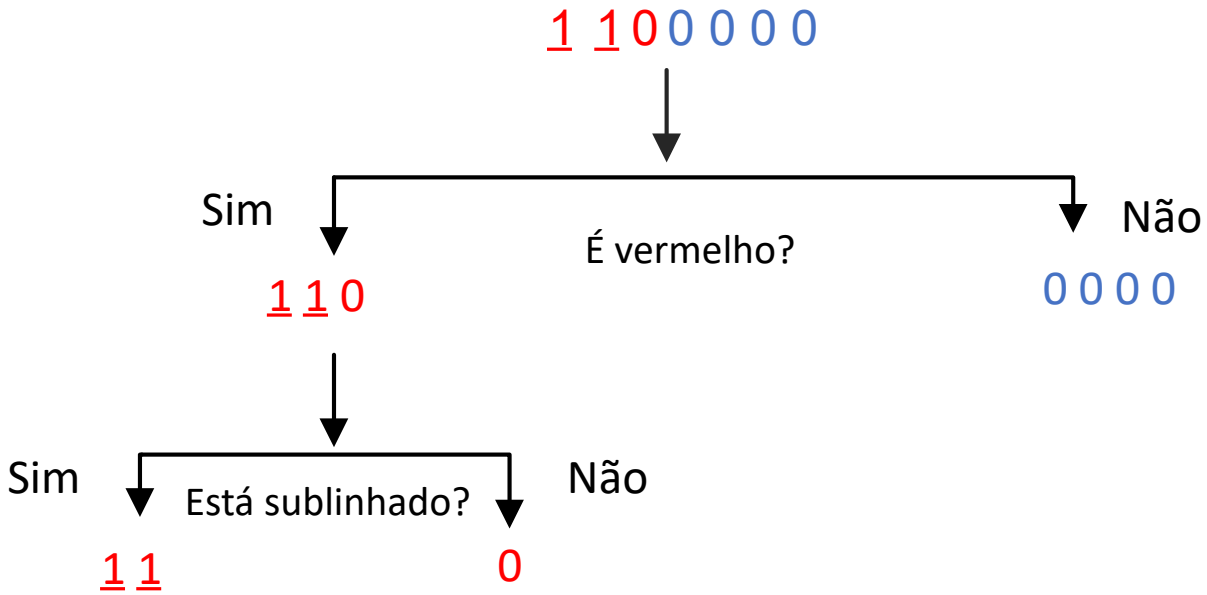
Para mais informações o manual da biblioteca pode ser consultado em (SCIKITLEARN, 2023a).

2.3.4 Random Forest

O algoritmo ensemble mais popular foi apresentado e desenvolvido independentemente por HO (1995) e Amit e Geman (1997), sendo nomeado de Random Forest. Anos mais tarde, esse algoritmo foi modificado e popularizado por Breiman (2021) em um artigo citado mais de cento e dez mil vezes segundo o Google Scholar em junho de 2023. A popularidade do Random Forest continua a aumentar, devido a sua simplicidade e desempenho preditivo. Random Forest é também considerado um método fácil de ser modificado comparado a outros métodos (SAGI; ROKACH, 2018).

Árvores de decisão (*decision trees*) são a base de construção do modelo Random Forest. Como exemplo, temos um conjunto de dados constituído dos valores 1 e 0, assim a classe pode assumir estes dois valores e ter duas variáveis: **cor**, na qual um número pode assumir vermelho ou azul e **sublinhado**, possibilitando um número ser sublinhado ou não. A partir desta premissa inicial, uma árvore de decisão será criada conforme a Figura 6. Para separar as classes, se faz necessário utilizar um nó com separação por cor. Sendo assim, serão geradas duas folhas na árvore, porém nota-se que o nó da esquerda ainda possui classes diferentes. Uma nova divisão será realizada separando as classes sublinhadas e assim definir os demais nós. O conceito de árvore de decisão irá a cada divisão de nó fazer a pergunta: qual variável permitirá dividir as observações de modo que os grupos resultantes sejam distintos e os membros resultantes sejam similares (YIU, 2021)?

Figura 6 – Exemplo de árvore de decisão.



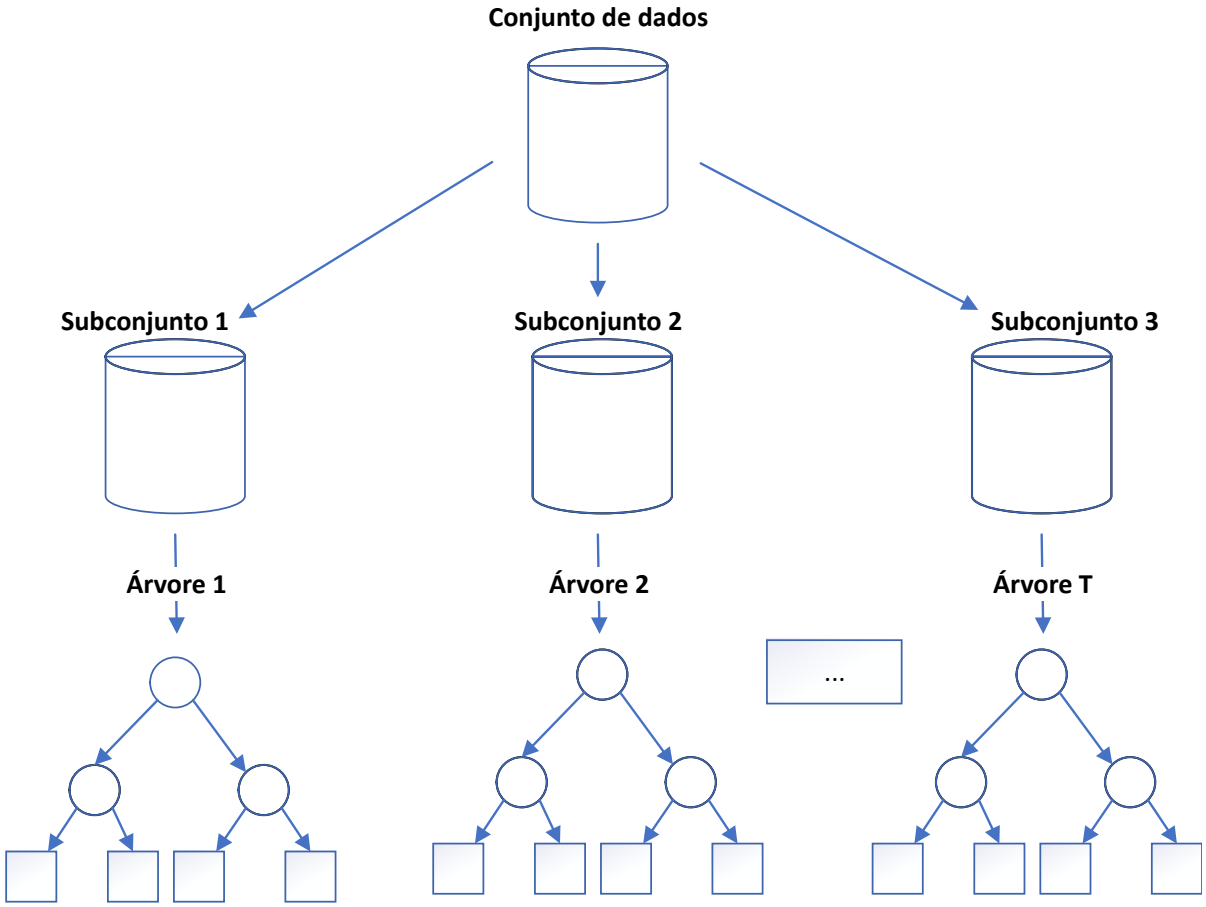
Fonte - Yiu (2021).

O Random Forest é um algoritmo ensemble que constrói múltiplas árvores de decisão conforme a Figura 7. Deste modo evita o problema de *overfitting* selecionando aleatoriamente as amostras de dados de treinamento para a construção de cada árvore,

resultando em um classificador com uma estrutura contra ruídos nos dados, ou seja, informações irrelevantes. Ademais, a seleção aleatória de variáveis para serem usadas na divisão do nó da árvore, permite um treinamento mais rápido do modelo, mesmo se a dimensionalidade do vetor da variável for grande (MISHINA et al., 2015).

As árvores individuais são construídas usando o procedimento apresentado no Algoritmo 2. O parâmetro de entrada N representa o número de variáveis de entrada utilizadas para determinar a decisão em um nó da árvore. Para a árvore do algoritmo aumentar, é inserido um processo aleatório nos indutores da árvore de decisão usando as seguintes abordagens: (a) treinar cada árvore em diferentes amostras de instâncias e (b) ao invés de escolher a melhor divisão em cada nó, o indutor utiliza amostras de um subconjunto de atributos e escolhe a melhor divisão entre eles. O segundo processo de randomização pode ser implementado diferentemente. Serão selecionados atributos de forma aleatória, ao invés do melhor em cada nó (uma informação de medida de ganho)(SAGI; ROKACH, 2018). Cada árvore de decisão no Random Forest pode crescer somente a partir de um subconjunto aleatório de variáveis, permitindo uma variação nos modelos das árvores e resultando em uma correlação baixa e aumentando a diversidade entre elas (YIU, 2021).

Figura 7 – Estrutura do Random Forest. O algoritmo cria aleatoriamente subconjuntos dos dados de treinamento. A duplicação e omissão das amostras selecionadas é permitida.



Fonte - Yiu (2021).

2.3.5 Etapas de funcionamento do Random Forest

O treinamento do conjunto de dados é realizado no algoritmo Random Forest através do algoritmo Bagging que cria amostras de subconjuntos. Com isso, os dados do conjunto de treinamento das amostras são removidos aleatoriamente e utilizados para construir uma árvore de decisão. No nó de divisão da árvore n (*splitting node*), o conjunto de amostra S_n é dividido em conjuntos de novas amostras S_t e S_r. Essa operação é realizada comparando o valor da quantidade da variável x_i com o valor máximo τ. A função de divisão dos nós seleciona combinações que podem particionar a maioria das amostras de variáveis selecionadas conforme a Equação (2.8) e o valor máximo representado na Equação (2.9) para cada classe.

$$f_k{}_{k=1}^{K} \tag{2.8}$$

$$\tau_{k\,k=1}^{H} \tag{2.9}$$

O número recomendado de seleção das variáveis é K, a raiz quadrada da dimensionalidade da variável. A função de avaliação usada para selecionar a combinação ótima é o ganho da informação ΔG. O processo de divisão é repetido recursivamente até que uma certa profundidade seja alcançada, ou até que o ganho de informação seja 0. Um nó folha é então criado e a classe de probabilidade $P(c|l)$ é armazenada (MISHINA et al., 2015).

A classificação dos dados utiliza uma amostra desconhecida inserida em todas as árvores de decisão, e as probabilidades de classificação correta dos dados através das classes dos nós folhas são calculadas. O resultado da classificação, é representado pela Equação (2.10), na qual, a classe escolhida será a que obtiver a melhor média de probabilidade entre todas as classes da árvore de decisão.

$$P(c|x) = \frac{1}{2}\sum_{t=1}^{T} P_t(c|x) \tag{2.10}$$

O conceito fundamental do modelo Random Forest é classificar os dados com utilização de muitas árvores de decisão não correlacionadas devido à aleatoriedade. Essa união do resultado da predição das árvores de decisão terá um desempenho superior ao modelo individual e estas, além de serem treinadas em diferentes conjuntos de dados, utilizarão diferentes variáveis para a tomada de decisão (YIU, 2021). Dificilmente é possível obter um número ideal de árvores para treinamento. Deste modo, existem no algoritmo muitas árvores de decisão redundantes que consomem muita memória (MISHINA et al., 2015).

O Algoritmo 2 a seguir se refere ao processo de classificação do Random Forest.

Algoritmo 2: O algoritmo Random Forest

`Entrada:` IDT(indutor de árvore de decisão), T(número de iterações), S(conjunto de treinamento), μ(o tamanho da sub amostra), N (número de atributos utilizados em cada nó)

`Saída:` M_t**:**$\forall_t$ **= 1,...., T**

For each T em 1, ...,T do

$S_t \leftarrow$ amostra de μ instâncias de S com substituição.

Construa um classificador M_t usando IDT(N) em S_t

t++

Para implementação do algoritmo tem-se a função RandomForestClassifier para ser utilizado na pesquisa.

O algoritmo Random Forest possui dezoito parâmetros para a avaliação do desempenho do algoritmo na predição dos dados de exoplanetas. Na seção 4.2 serão testados várias combinações de valores para os seguintes parâmetros:

n_estimators: inteiro, valor padrão igual a 100. O número de árvores de decisão no algoritmo.

criterion: float, valor padrão igual à função "gini". Função para medir a qualidade de uma divisão do nó da árvore. Os outros valores possíveis são ["entropy", "log_loss"].

max_depth: float, valor padrão igual à None. A profundidade máxima da árvore será None, os nós serão expandidos até todas as folhas estiverem puras ou até todas as folhas conterem menos que a quantidade mínima de amostras de divisão.

max_features: inteiro ou float, valor dpadrãoigual à "sqrt". O número de variáveis consideradas quando procurando pela melhor divisão do nó:

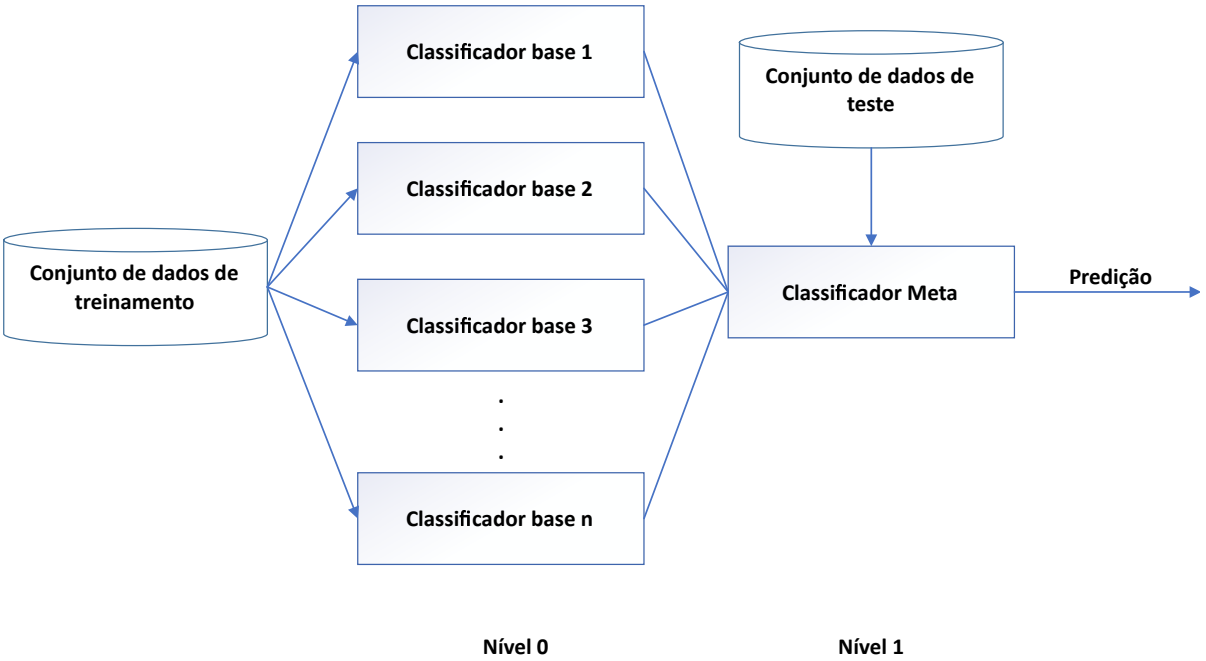
Para mais informações, o manual da biblioteca pode ser consultado em (SCIKITLEARN, 2023d).

2.3.6 Stacking

Stacking ou *stacked generalization* utiliza um algoritmo de meta-aprendizado visando aprender a melhor forma de unir dois ou mais algoritmos de aprendizado de máquina. O benefício da implementação do Stacking é devido a este algoritmo, utilizar as principais capacidades de vários modelos, fazendo predições que terão um desempenho superior a qualquer outro modelo Ensemble (BROWNLEE, 2021). Os resultados das predições de cada algoritmo individual no Stacking são tratados como um novo conjunto de dados de treinamento e serão adicionadas em uma nova camada chamada de modelo-meta, conforme a Figura 8.

O algoritmo Stacking pode ser visualizado como uma camada que contém vários algoritmos de aprendizado de máquina empilhados. Os algoritmos pertencentes a esta camada, possuem conhecimento destes dados. Por exemplo, em um jogo de perguntas e respostas relacionado à história é possível obter ajuda de duas pessoas, uma com graduação em Ciência da Computação e outra em História. O conhecimento permite que a escolha mais adequada seja de obter ajuda da pessoa com graduação em História. Deste modo, os algoritmos na camada do Stacking terão um conhecimento melhor para predizer os dados (KUMAR; MAYANK, 2020).

Figura 8 – Estrutura conceitual do Stacking.



Fonte - Chanamarn e Sitidech (2016).

Ademais, o algoritmo Stacking aborda a seguinte pergunta: com a possibilidade de utilizar vários algoritmos, estes possuindo um ótimo desempenho na resolução de problemas, mas de maneiras diferentes, como e qual algoritmo será escolhido para integrar o modelo Ensemble? O algoritmo Stacking para responder essa questão, irá utilizar a abordagem de escolha de um algoritmo que consiga aprender quando utilizar e confiar em cada algoritmo no modelo Ensemble (BROWNLEE, 2021).

Ao contrário do algoritmo Bagging, no Stacking os algoritmos são tipicamente diferentes, ou seja, os algoritmos no modelo não serão somente constituídos de árvores de decisão como acontece no Random Forest. Também, os algoritmos serão treinados no mesmo conjunto de dados ao invés de amostras destes dados. Ao contrário do algoritmo Boosting, o Stacking utiliza um único modelo para aprender como combinar melhor as predições dos modelos contribuintes, ao invés de uma sequência de modelos que corrigem as predições dos modelos anteriores, como acontece no algoritmo Adaboost. A arquitetura do modelo Stacking é constituída de dois ou mais modelos-base, frequentemente referidos como nível 0, e um modelo de meta-aprendizado que combina as predições dos modelos-base, denominado modelo de nível 1 (BROWNLEE, 2021).

- **Modelo de nível 0 (modelos-base)**: modelos treinados nos dados de treinamento na qual as predições são compiladas.
- **Modelo de nível 1 (modelo-meta)**: modelos que aprendem como melhor combinar as predições dos modelos-base.

O modelo-meta será treinado com os dados resultantes das predições feitas pelos modelos-base. Os dados não utilizados para treinamento dos modelos-base (dados de teste), também serão utilizados para realizar as predições e servirão como dados de entrada e saída para o treinamento do modelo-meta. A maneira mais comum de preparar o conjunto de dados de treinamento para o modelo-meta é através da técnica *cross-validation k-fold* dos modelos-base, no qual, as predições denominadas *out-of-fold* (constituídas de dados de teste) são utilizadas como base para o conjunto de dados. Quando o conjunto de dados de treinamento é preparado para o modelo-meta, este pode ser treinado isoladamente, enquanto os modelos-base são treinados no conjunto de dados original e por completo (BROWNLEE, 2021).

Os modelos-base são geralmente complexos e diversos, sendo recomendado o uso de vários modelos de aprendizado de máquina que efetuam diferentes suposições em como resolver a tarefa de predição, como os algoritmos lineares: Decision Trees, Support Vector Machines e Neural Networks. Outros algoritmos ensemble podem ser usados também como modelos-base como o Random Forest. O modelo-meta é mais simples, pois apenas interpreta as predições feitas pelos modelos-base. Um exemplo de modelo-linear utilizado no modelo-meta é o algoritmo Logistic Regression (BROWNLEE, 2021).

O Stacking foi desenvolvido para melhorar o desempenho do modelo Ensemble, contudo não é garantido para todos os casos. Este dependerá da complexidade do problema e se está bem representado pelos dados de treinamento. Também, dependerá da escolha dos modelos-base e se eles possuem capacidade de aprendizado e não correlação entre as suas predições (BROWNLEE, 2021).

O Algoritmo 3, adaptado de Smolyakov (2017), mostra o pseudocódigo do Stacking.

Algoritmo 3: O algoritmo do Stacking

`Entrada:` dados de treinamento D = $\{x_i, y_i\}_{i=1}^{m}$
`Saída:` classificador ensemble H
Passo 1: classifcadores de nível-base aprendem
For T = 1 para T do
    Aprenda h_t baseado em D
End for
Passo 2: construa novo conjunto de dados de predições
For i = 1 para m do
    $D_h = \{x_i^{'}, y_i\}$, onde $x_i^{'} = \{h_1(x_i),...,h_T(x_i)\}$
End for
Passo 3: aprenda um classificador meta
Aprenda H baseado em D_h
Retorne H

Para implementação tem-se a função StackingClassifier que foi utilizada na pes-

quisa.

O algoritmo Stacking possui sete parâmetros para a avaliação do desempenho do algoritmo na predição dos dados de exoplanetas. Na seção 4.3 serão testados várias combinações de valores para os seguintes parâmetros:

estimators: lista de string. Estimadores base que serão empilhados (stacked) juntos, cada elemento da lista é definido como uma tupla (tipon de estrutura de dados da linguagem Python) de strings.

final_estimator: valor default será o algoritmo LogisticRegression. Um classificador que será utilizado para combinar as predições dos estimadores-base.

Para mais informações, o manual da biblioteca pode ser consultado em (SCIKI-TLEARN, 2023e).

2.3.7 Random Subspace Method (RSM)

O método Random Subspace Method foi primeiramente introduzido por HO (1998) e desenvolvido para construir árvores de decisão. O RSM analisa todo o conjunto de dados com alta dimensão de variáveis, cria aleatoriamente conjuntos de amostras de baixa dimensionalidade (número de variáveis baixa e na literatura denominado subespaço), e constrói um classificador. Na próxima etapa é aplicada uma regra de combinação para a decisão final. O RSM é um método Ensemble muito simples e popular. Contudo, a utilização de amostras aleatórias do conjunto de dados de alta dimensão não é prática. A amostragem aleatória de sub espaço consome muito tempo, para assegurar um número suficiente e exato de amostras do conjunto de alta dimensão, levando a um maior esforço computacional (ZHU; LIU; CHEN, 2009).

O algoritmo RSM também pode ser explicado como um modelo que combina predições de várias árvores de decisão, treinadas em diferentes subconjuntos de colunas do conjunto de dados de treinamento. A variação na utilização de colunas para treinar cada membro do modelo ensemble contribui para a diversidade e também aumenta o desempenho em relação à apenas uma árvore de decisão. O RSM não está restrito somente a utilizar árvores de decisão, pode-se utilizar outros algoritmos no qual o desempenho varia significativamente na escolha dos parâmetros de entrada (BROWNLEE,).

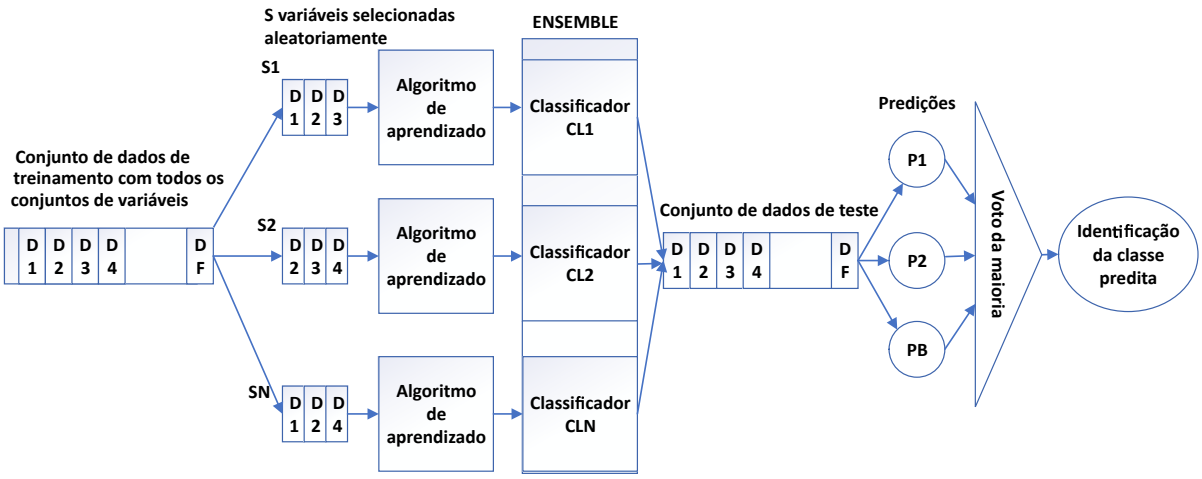
Cada coluna do conjunto de dados é denominada de variável. Pode-se considerar todas as variáveis de entrada juntas como sendo um vetor de espaço n-dimensional, no qual o valor n é o número de variáveis e cada linha deste conjunto de dados é um ponto no espaço destas variáveis. Este é um conceito comum em aprendizado de máquina e quando o espaço de variáveis de entrada se torna maior, consequentemente a distância entre os pontos do espaço aumentam, conhecido como curso de dimensionalidade (BROWNLEE, 2021).

Um subconjunto de variáveis de entrada pode ser definido como subespaço e selecionar as variáveis é uma forma de definir este subespaço no espaço das variáveis de entrada. Como exemplo, a seleção de variáveis refere-se a tentativa de reduzir o número de dimensões do espaço de variáveis de entrada, selecionando um subconjunto de variáveis para manter ou excluir, baseando-se no relacionamento com a classe alvo. Alternativamente, pode-se selecionar subconjuntos aleatórios de variáveis de entrada para definir subespaços aleatórios. Essa abordagem pode ser utilizada como base de um modelo ensemble, no qual pode ser treinado em cada subespaço aleatório de variáveis, assim definido como Random Subspace Method (BROWNLEE, 2021).

O algoritmo Random Subpasce Method constrói aleatoriamente um conjunto de variáveis, selecionando amostras de variáveis e treinando classificadores básicos nos subespaços, visando gerar múltiplos resultados antes do resultado final (DONG et al., 2020).

A Figura 9 exibe o processo de treinamento e teste do algoritmo Random Subpace Method e o Algoritmo 4 de (DU; LIU; XI, 2015) mostra seu pseudocódigo.

Figura 9 – Algoritmo Random Subspace Method.



Fonte - Derhab (2019).

Algoritmo 4: O algoritmo do Random Subspace method

Entrada: o conjunto original de treinamento
S_training = $\{(x_i, y_i)\}, i = 1, 2, ..., m$, onde $x_i \in X, y_i \in Y \subset \{l_1, l_2, ..., l_k\}$
D: número de variáveis do conjunto de treinamento;
For t = 1,. . . , T
Selecione aleatoriamente um subespaço com d (d<D) features das features D e obtenha um subconjunto de treinamento S_t;
Construa um classificador $C_t(x)$ utilizando o subconjunto de treinamento S_t;
End for
Saída: A predição final $P(x_i) = \arg\max_{y \in Y} \sum_{t; C_t(x_i)=y} 1$

Para implementação, tem-se o algoritmo BaggingClassifier que possui parâmetros que permitem a configuração para aleatoriamente trabalhar com amostras de subconjuntos de variáveis, assim esse algoritmo será configurado para funcionar como a lógica do Random Subspace Methods.

O algoritmo BaggingClassifier possui onze parâmetros, para a avaliação do desempenho do algoritmo na predição dos dados de exoplanetas, na seção 4.4 serão testados várias combinações de valores para os seguintes parâmetros:

n_estimators: inteiro. Número de estimadores no ensemble. Como default possui dez estimadores.

max_features: inteiro ou float. O número de features para serem retiradas de X para treinar cada estimador base. Valor default é igual a 1,0.

max_samples: inteiro ou float. O número de amostras para serem retiradas de X para treinar cada estimador base. Valor default é igual a 1,0.

Para mais informações, o manual da biblioteca pode ser consultado em (SCIKI-TLEARN, 2023b).

2.3.8 Extremely Randomized Trees

O algoritmo Extremely Randomized Trees ou Extra Trees foi proposto por (GEURTS; ERNST; WEHENKEL, 2006) como uma abordagem para gerar diversos algoritmos Ensemble, inserindo aleatoriedade no processo de treinamento. Além de adicionar o melhor atributo dos subconjuntos aleatórios de variáveis, pontos de divisão das variáveis também são randomizados no treinamento do algoritmo Extremely Randomized Trees (SAGI; ROKACH, 2018). Ademais, para cada variável (aleatoriamente selecionada em cada interior de um nó), um ponto máximo de discretização (*cut point*) é selecionado aleatoriamente para divisão, ao invés de escolher o melhor *cut point* baseado na amostra local (GEURTS; LOUPPE, 2011).

O mesmo conjunto de treinamento é utilizado para induzir todas as árvores de decisão. Embora, o Extremely Randomized Trees tende a ter um alto viés e componentes de variância. Para eliminar este problema, o algoritmo une um conjunto de árvores de decisão que seja o suficiente. Muitos estudos apresentam a eficácia de Randomized Trees no domínio da segmentação de tumores cerebrais (SAGI; ROKACH, 2018).

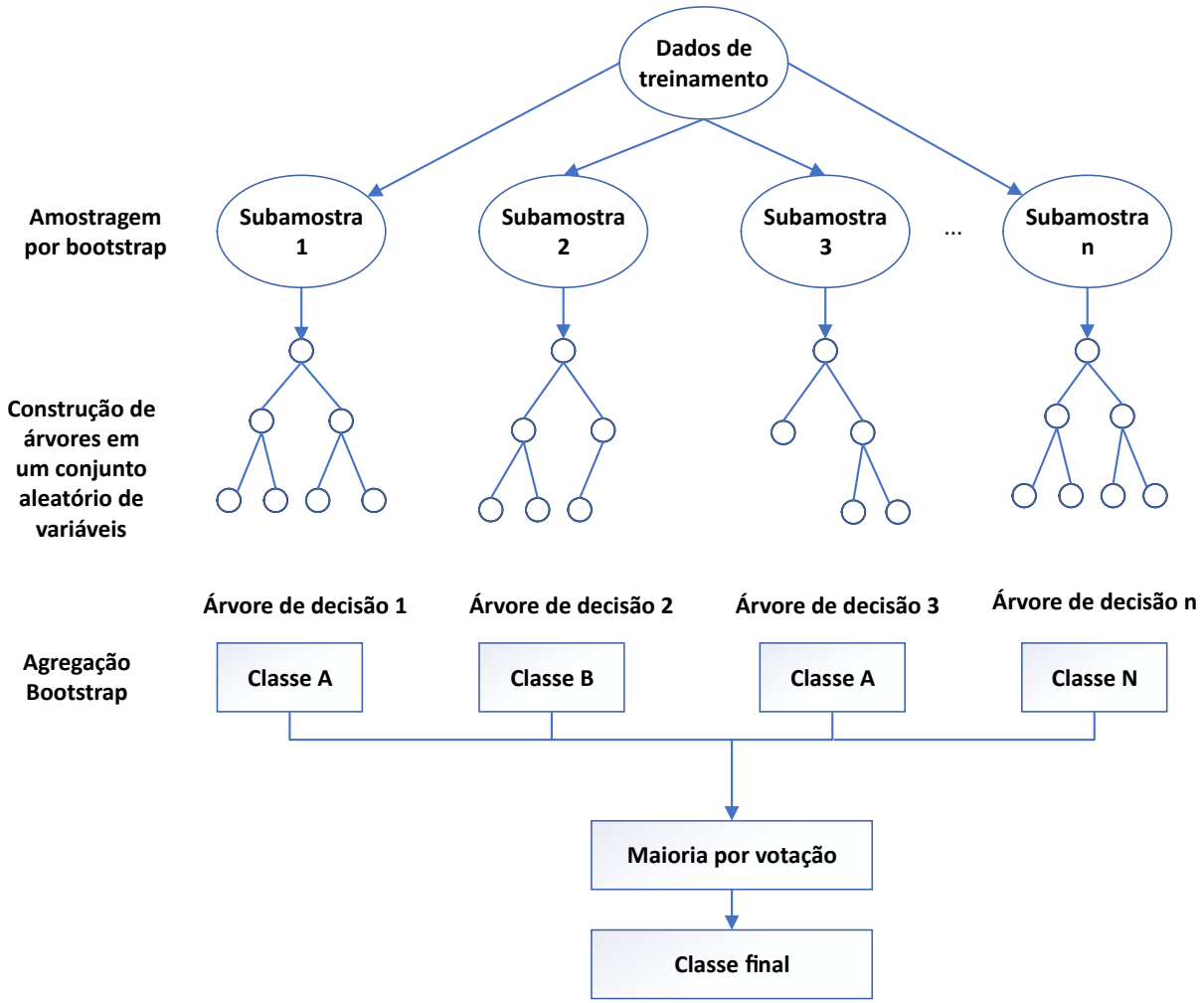
Este algoritmo combina predições de muitas árvores de decisão, aspecto que o faz ser relacionado ao algoritmo Random Forest. Mesmo utilizando um algoritmo mais simples para construir as árvores pertencentes ao modelo ensemble, em muitos casos é mais eficiente ou melhor do que o Random Forest. Seu funcionamento se inicia criando inúmeras árvores de decisão e as predições serão feitas utilizando a técnica de maioria

de votos para escolher a classe correta. O algoritmo irá aleatoriamente retirar amostras de variáveis em cada ponto de divisão da árvore de decisão e irá selecionar um ponto de divisão aleatoriamente.

Existem três principais parâmetros para serem alterados no algoritmo, sendo: número de árvores de decisão no modelo ensemble, número de variáveis de entrada para ser selecionada aleatoriamente e considerada para cada ponto de divisão e o número mínimo de amostras requeridas em um nó para criar um ponto de divisão. A seleção aleatória de pontos de divisão faz com que árvores de decisão fiquem menos correlacionadas, mas aumenta a variância no algoritmo. O aumento da variância pode ser tratado com aumento no número das árvores de decisão utilizadas no algoritmo (BROWNLEE, 2021).

A Figura 10 detalha a estrutura do algoritmo Extremely Randomized Trees e os Algoritmos 5 e 6 detalham seu pseudocódigo por (GEURTS; ERNST; WEHENKEL, 2006).

Figura 10 – Estrutura do algoritmo Extremely Randomized Trees.



Fonte - Geurts; Ernst; Wehenkel (2006).

Algoritmo 5: O algoritmo do modelo ensemble Extremely Randomized Trees

Contruir_uma_extra_tree_ensemble(S)
Entrada: um conjunto de treinamento S.
Saída: uma árvore ensemble $T = \{t_1, .., t_m\}$.
For i = 1 até M
Gere uma árvore: t_i = **Contruir_uma_extra_tree_ensemble(S)**;
_ Return T.

Algoritmo 6: O algoritmo do modelo Extremely Randomized Trees

Contruir_uma_extra_tree(S) **Entrada:** um conjunto de treinamento S.
Saída: uma árvore t.
_ Return T uma folha rotulada pela frequência de classes em S IF
(i) |S| < $nmin$, ou
(ii) todos os atributos candidatos são constantes em S, ou
(iii) a variável de saída é constante em S
_Senão:
1. Selecione randomicamente atributos K, $\{a_1, \ldots, a_k\}$, sem substituição, entre todos (não constante em S) atributos candidatos.
2. Gere K divisões $\{s_1, ..., s_k\}$, onde s_i = Escolha_uma_divisão_Aleatoria(S, a_i), $\forall$ =1,...,k;
3. Selecione uma divisão s_* que o Score(s_*,S) = $\max i = 1, ..., k^S core(s_*, S)$
4. Divida S em sub conjuntos S_1 e S_r de acordo com o teste S_*;
5. Construa t_1 = **Contruir_uma_extra_tree(S)** e t_r = **Contruir_uma_extra_tree(**S_r**)** dos subconjuntos;
6. Crie um nó com a divisão s_*, junte com t_1 e t_r, como árvores esquerda e direita deste nó e retorne a árvore resultante t.

Para implementação, tem-se o algoritmo ExtraTreesClassifier para ser utilizado na pesquisa.

O algoritmo Extremely Randomized Trees possui dezoito parâmetros para a avaliação do desempenho do algoritmo na predição dos dados de exoplanetas. Na seção 4.5 serão testadas várias combinações de valores para os seguintes parâmetros:

n_estimators: recebe valor inteiro. Como padrão possui 100 estimadores. Quantidade de estimadores para o modelo ensemble.

criterion: float, valor default igual à função "gini". Função para medir a qualidade de uma divisão do nó da árvore. Os outros valores possíveis são ["entropy", "log_loss"].

max_depth: float, valor default igual à None. A profundidade máxima da árvore, se o valor for considerado o default None, os nós serão expandidos até todas as folhas estiverem puras ou até todas as folhas conterem menos que a quantidade mínima de amostras de divisão.

max_features: inteiro ou float, valor default igual à "sqrt". O número de features consideradas quando procurando pela melhor divisão do nó.ç

Para mais informações, o manual da biblioteca pode ser consultado em (SCIKI-TLEARN, 2023c).

2.4 Avaliação dos algoritmos de aprendizado de máquina

De acordo com (RASCHKA, 2018), a avaliação dos algoritmos de aprendizado de máquina é realizada perante a resposta de algumas perguntas importantes que servem como base para a escolha do algoritmo, as quais estão descritas a seguir:

- Como saber se o algoritmo irá fazer previsões bem sucedidas com dados não observados a partir dos dados observados (dados de treino)? O primeiro passo do aprendizado supervisionado é treinar os dados sobre o modelo, mas como garantir que os novos conjuntos serão preditos de forma mais precisa?
- Como saber se o modelo não está apenas memorizando os dados sobre o qual foi treinado e falhando na predição correta de amostras futuras?
- Como selecionar o melhor algoritmo para o problema em questão?

A predição em dados futuros é geralmente o maior problema a ser resolvido em aplicação de aprendizado de máquina ou no desenvolvimento de novos algoritmos, sendo necessário estimar e ordenar o respectivo desempenho de cada modelo. Os principais pontos de avaliação do desempenho preditivo dos algoritmos são:

1. Estimar o desempenho de generalização do modelo sobre dados futuros (não vistos);
2. Aumentar o desempenho preditivo ajustando o algoritmo de aprendizado e selecionando o modelo com o melhor desempenho em um espaço de hipóteses;
3. Identificar o algoritmo mais adequado por meio de comparação e selecionar com o melhor desempenho.

2.4.1 Terminologias

Exatidão da predição: definida como as predições corretas dividas pelo número de exemplos no conjunto de dados.

Viés: refere-se ao viés estatístico. Em termos gerais, o viés de um estimador $\hat{\beta}$ é a diferença entre o valor esperado $E[\hat{\beta}]$ e o verdadeiro valor do parâmetro β estimado conforme a Equação (2.11):

$$Viés = E[\hat{\beta}] - \beta \tag{2.11}$$

Variância: simplesmente a variância estatística do estimador $\hat{\beta}$ e seu valor esperado $E[\hat{\beta}]$. A Equação (2.12) é definida pela diferença do valor valor real menos o valor esperado elevado ao quadrado.

$$Variância = E[(\beta - E[\hat{\beta}])^2] \tag{2.12}$$

Função alvo: a função alvo $f(x) = y$ é a função verdadeira $f(.)$ que será modelada.

Hipótese: uma certa função similar a verdadeira função.

Modelo: Em aprendizado de máquina, os termos hipótese e modelo são intercambiáveis. Em outras ciências, a hipótese pode ser considerada a melhor predição realizada pelo cientista, contudo o modelo será a manifestação desta predição para testar a hipótese.

Algoritmo de aprendizado: conjunto de instruções que tentam modelar a função alvo, usando um conjunto de dados de treinamento. Um algoritmo de aprendizado trabalha com um espaço de hipóteses e o conjunto de possíveis hipóteses pode ser explorado para modelar a função alvo ainda desconhecida, formulando a hipótese final.

Hiper-parâmetros: são parâmetros de configuração do algoritmo.

2.4.2 Métodos de validação

- **Validação *Holdout*:** o conjunto de dados é divido em duas partes, uma sendo para treinamento, geralmente corresponde a 70% dos dados, e a outra como teste, representando 30% dos dados. Estes dois novos conjuntos de dados serão utilizados para avaliação do algoritmo de aprendizado de máquina. É fundamental que o conjunto de testes seja utilizado somente uma vez, visando evitar viés na estimação do desempenho preditivo do algoritmo. A validação de *holdout* é realizada conforme o seguinte processo: um algoritmo é escolhido e treinado com os dados de treinamento. Na etapa seguinte, os dados de teste serão utilizados para verificar como o modelo irá predizer estes novos dados que ainda não são conhecidos e será finalmente realizada a estimativa de desempenho do modelo.

  Na validação *holdout*, quando um modelo ainda não atingiu sua capacidade máxima de predição, seu desempenho será enviesado de maneira chamada de pessimista, ou seja, o algoritmo ainda poderá aprender melhor se mais dados forem consumidos. Quando é dividido uma parte do conjunto de dados em dados de teste, são retidos dados valiosos que poderiam contribuir para o desempenho geral do modelo (RASCHKA, 2018).

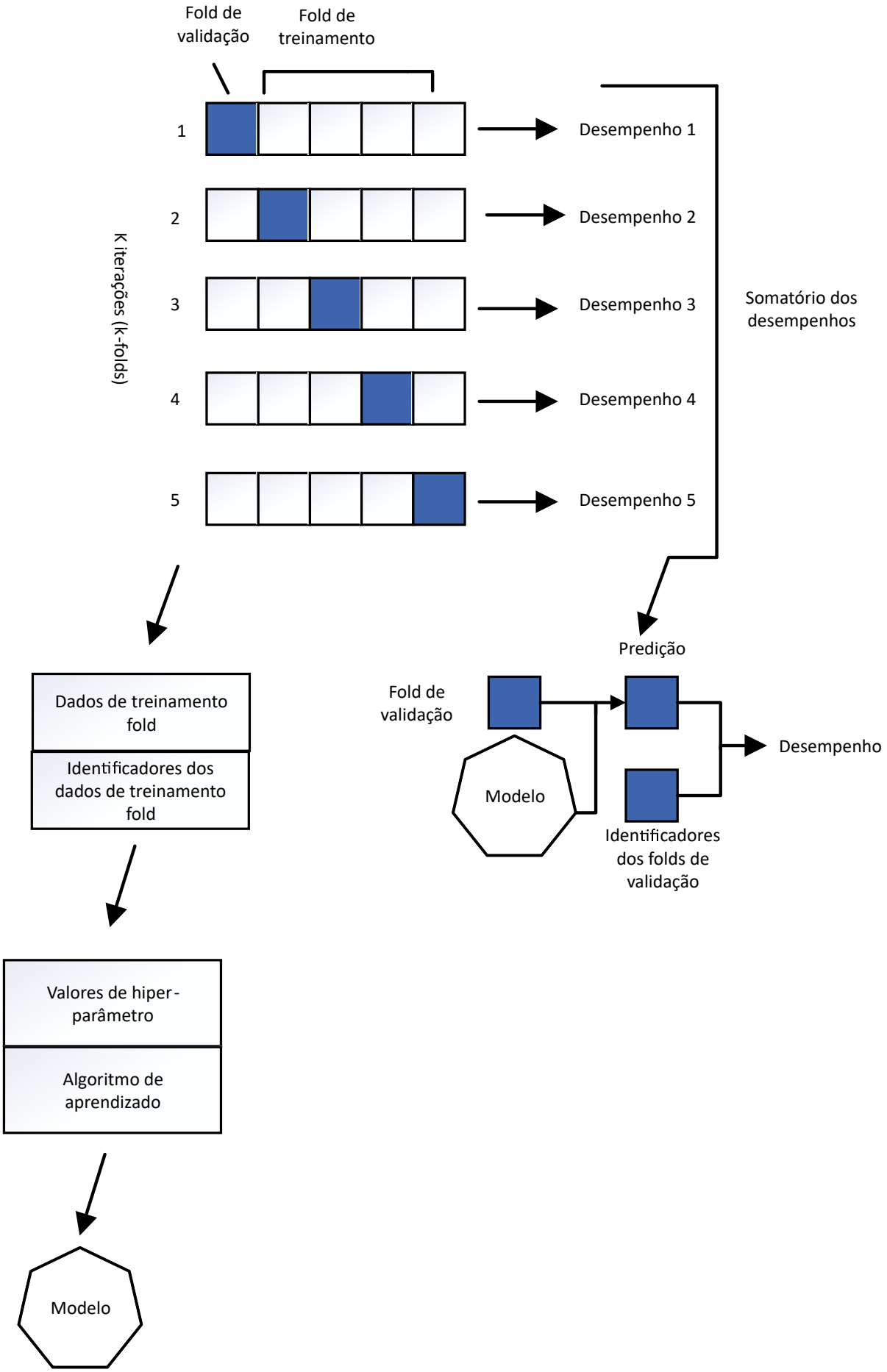
- **Validação cruzada *k fold***: é a técnica mais utilizada para validação de modelos de aprendizado de máquina, utilizado amplamente na literatura. Alguns pesquisadores mencionam o método de *holdout* como sendo uma técnica de validação cruzada, contudo é mais coerente uma técnica de validação cruzada percorrer as etapas de treinamento e teste em rodadas sucessivas. O principal conceito é testar toda a amostra do conjunto de dados. Um caso especial é denominado de *k fold* e este permite uma iteração sobre o conjunto de dados múltiplas (k) vezes. Em cada rodada, os dados são divididos em (várias) k partes: uma parte é utilizada para validação e as outras faltantes k-1 serão unidas em um subconjunto de treinamento, para avaliação do modelo conforme mostrado na Figura 11 onde é ilustrado o processo de validação cruzada de 5 *k folds*.

A principal diferença entre os métodos de validação cruzada e *holdout* seria a porcentagem dos dados utilizados para treinamento e teste. Enquanto o método de validação cruzada utiliza todos os dados como treinamento e teste, visando reduzir o viés pessimista, o método *holdout* divide os dados em dados de treinamento e dados de teste.

Para selecionar o melhor algoritmo específico para um problema de aprendizado de máquina através da validação cruzada *k fold*, se faz necessário seguir os seguintes passos:

1. Similar ao método de *holdout*, o conjunto de dados será dividido em duas partes: treinamento e teste, este sendo retirado para a etapa final;
2. Utilizar vários valores de hiperparâmetros. Por exemplo, através da função *grid search* na linguagem Python. Para cada hiperparâmetro será aplicada a validação cruzada *k fold* no conjunto de treinamento, resultando em múltiplos modelos e estimativas de desempenho;
3. Definir a melhor combinação de parâmetros, que será configurada no modelo para um novo treinamento;
4. Utilizar o conjunto de teste da etapa 1 para avaliar o modelo obtido no passo 3;
5. Treinar facultativamente o modelo com os dados de treinamento e testes combinados.

Figura 11 – Processo de Validação Cruzada k folds.



Fonte - Raschka (2018).

2.5 *Overfitting*

No aprendizado supervisionado, em diversos casos os algoritmos não conseguem generalizar suficientemente o conjunto de dados de treinamento e isso impacta na predição com os dados de testes. Este fenômeno é chamado de *overfitting*. O modelo que está com *overfitting* tem maior dificuldade em lidar com informações dos dados de teste do que dos dados de treinamento. Também, estes modelos tendem a memorizar todos os dados, incluindo ruídos inevitáveis dos dados de treinamento, ao invés de aprender com estes. Devido a este problema, o modelo terá um desempenho de predição melhor nos dados de treinamento do que nos dados de teste (YING, 2019).

Os seguintes motivos são responsáveis pelo *overfitting* nos dados de acordo com (YING, 2019):

1. Aprendizado de ruídos nos dados de treinamento: quando o conjunto de dados de treinamento é muito pequeno, possui poucos dados representativos ou existem dados com muitos ruídos. O modelo poderá aprender com estes ruídos e posteriormente utilizá-los na predição.

2. Complexidade das hipóteses: existe um conceito importante no aprendizado de máquina que é denominado *trade-off* em complexidade. Significa haver um comprometimento entre variância e viés no qual se refere ao equilíbrio entre consistência e exatidão. Quando os algoritmos possuem muitas hipóteses (entrada de dados), o modelo se torna mais exato em média, mas com pouca consistência. Portanto, os modelos podem ser drasticamente diferentes em diversos conjuntos de dados.

3. Muitos procedimentos de comparação: durante o processo de comparação em um modelo de aprendizado de máquina, múltiplos dados são comparados baseados em pontuação de uma função de avaliação e será selecionado o dado com a pontuação máxima. Contudo, este processo escolherá dados que não irão melhorar ou até mesmo reduzir a exatidão da predição.

3 Metodologia

A abordagem de metodologia utilizada nesta dissertação é quantitativa, por analisar e comparar os resultados de diferentes métricas de avaliação para os algoritmos. Quanto a análise dos dados, a pesquisa será exploratória, devido à utilização de métodos para a comparação dos modelos Ensemble sobre o conjunto de dados de exoplanetas.Esta comparação ainda não encontrada na literatura. Os algoritmos serão comparados em relação ao seus desempenhos e uma discussão será feita sobre o resultado obtido de cada algoritmo. O objetivo desta pesquisa é estudar, analisar e comparar o desempenho dos algoritmos Ensemble na predição de dados de exoplanetas.

O conjunto de dados será extraído do arquivo da NASA *exoplanet archive*, denominado KOI. Este está disponibilizado para a comunidade científica e dispõe de um conjunto de ferramentas e banco de dados mantidos pela NASA através da missão do telescópio Kepler. O conjunto de dados será primeiramente tratado para eliminar dados não necessários (ruídos). Os dados faltantes serão preenchidos e os que forem qualitativos serão convertidos para a forma binária, assim todos os dados serão numéricos. Os dados serão normalizados para garantir que fiquem em uma mesma escala de valor. Os algoritmos serão analisados quanto aos seus desempenhos sobre diferentes métricas relacionadas a capacidade de predizer corretamente um exoplaneta, utilizando o conjunto de dados. Será criada uma matriz de confusão para cada algoritmo, sendo possível obter os acertos e erros da predição.

Para a implementação e execução dos algoritmos foi utilizada a plataforma de desenvolvimento chamada de Kaggle, devido aos seguintes benefícios descritos por (PRZYBYLAR, 2020):

- **`Dados:`** possui uma lista com vários conjuntos de dados que podem ser pesquisados por nome e estão em maioria armazenados em formato .csv, mas podem ser encontrados em outros formatos como JSON, SQLite, arquivos e BigQuery.
- **`Código:`** muitos códigos fontes de outros usuários podem ser facilmente encontrados, onde é possível visualizá-los. Na maioria das vezes possuem comentários facilitando o aprendizado e benchmarking. Os códigos estão em sua maioria escritos na linguagem Python, mas também são encontrados em outras linguagens como R, SQLite e Julia. Os códigos são encontrados em um formato chamado notebook, também conhecido como Jupyter Notebook, que possui uma extensão .ipynb. A plataforma possui diversos exemplos que mostram modelos de Aprendizado de Máquina do começo ao fim. Alguns modelos incluem preparação dos dados, análise exploratória, *feature engineering*, implementação do modelo base, resultados e sua interpretação.

- **`Comunidade:`** como a aplicação GitHub, Stack Overflow, LinkedIn, os usuários do Kaggle participam em uma comunidade com analistas de dados, cientistas de dados e engenheiros de Aprendizado de Máquina, sendo possível o aprendizado, compartilhamento de informação e aumento do *networking*. Cada usuário, pode postar o próprio trabalho (códigos e notebooks), assim a comunidade sempre crescerá.
- **`Inspiração:`** possui diversos conjuntos de dados, códigos fontes, comunidades, cursos e competições, tudo isso, torna a possibilidade de ser uma inspiração ao usuário.
- **`Competição:`** oferece inúmeras competições que além de ajudar pessoas e empresas, oferecem remuneração.
- **`Cursos:`** possui cursos voltados a ciência de dados. Atualmente possui uma lista com 14 cursos como introdução a Aprendizado de Máquina, visualização de dados, SQL e Python.

A plataforma online Kaggle disponibiliza um hardware com as seguintes configurações (KAGGLE.COM, 2023): mémoria RAM de 30GB, CPU com 4 núcleos que pode ser AMD ou Intel, o fabricante da CPU varia em cada execução. É disponibilizado também caso necessário GPU, contudo os algoritmos implementados utilizam apenas a CPU.

A linguagem de desenvolvimento escolhida foi Python devido aos seguintes motivos citados por (BEKLEMYSHEVA, 2022):

- **`Simples e consistente:`** o código em Python é conciso e fácil de ser entendido. Os desenvolvedores podem colocar seus esforços em resolver os problemas de Inteligência Artificial, ao invés de focar nas técnicas da linguagem. Ademais, Python é considerado por muitos desenvolvedores como sendo fácil de se aprender e o código pode ser entendido facilmente por humanos, facilitando a criação de um modelo de Aprendizado de Máquina.
- **`Seleção extensiva de bibliotecas e *frameworks*:`** a implementação de algorítimos de Aprendizado de Máquina pode ser complicada e requer muito tempo. É essencial ter uma estrutura bem definida e ambiente bem testado para os desenvolvedores criarem as melhores soluções de código. Para reduzir tempo de desenvolvimento, os programadores utilizam inúmeros *frameworks* e bibliotecas da linguagem Python. Uma biblioteca é um código pré-escrito utilizado para resolver tarefas comuns de programação. Algumas das Bibliotecas mais utilizadas em Python são:
  - Keras, TensorFlow e Scikit-learn para Aprendizado de Máquina;
  - NumPy para computação científica de alto desempenho e análise de dados;
  - SciPy para computação avançada;

  - Pandas para análise de dados no geral;
  - Seaborn para visualização de dados.

- **`Plataforma independente:`** Python é suportado em diversas plataformas como Linux, Windows e MacOS, permitindo ser instalado entre diferentes sistemas operacionais.

- **`Grande comunidade e popularidade:`** em uma pesquisa realizada pela Stack Overflow em 2020, Python estava entre as 5 linguagens mais populares para projetos Aprendizado de Máquina, sendo assim é fácil encontrar desenvolvedores e conteúdo para construir estes modelos.

3.1 Pré-processamento dos dados

O seguinte processo se refere ao pré-processamento do conjunto de dados KOI. Foi realizado o download do conjunto de dados KOI, que possui o nome "cumulative.csv" através do site eletrônico (NASA, 2023). Após esta etapa, foi carregado na plataforma (KAGGLE, 2023). O conjunto de dados, ainda sem tratamento, continha 9.564 linhas e 50 colunas. O arquivo do conjunto de dados em formato CSV possui um tamanho de 3,03 MB. Para o tratamento dos dados, foram seguidas as seguintes etapas:

1. Remover colunas não utilizadas: seis colunas do KOI foram removidas, pois não contribuem para o processo de predição. Estas colunas possuem somente dados de identificação dos possíveis exoplanetas. As colunas removidas foram:
   - rowid: um identificador numérico de cada linha do KOI;
   - kepid: um identificador único e numérico de cada exoplaneta;
   - kepoi_name: segundo identificador único e numérico do KOI;
   - kepler_name: nome em formato textual dado ao exoplaneta já confirmado;
   - koi_pdisposition: dado textual que identifica a explicação provável do dado, sendo falso-positivo ou não disposto, ou candidato;
   - koi_score: valor número entre 0 e 1 que exibe a confiança no dado KOI.
2. Limitar o valor a ser encontrado nas variáveis "candidato" e "confirmado": O conjunto de dados contém linhas com valores falso-positivo. Essas linhas com estes valores foram removidas;
3. Transformar a coluna alvo "koi_disposition" em valores binários: como esta coluna possui as identificações "candidato" e "confirmado", foi atribuído o valor 1 para o exoplaneta candidato e 0 para o exoplaneta confirmado;

4. Remover as colunas sem valores: as colunas “koi_teq_err1” e “koi_teq_err2” estavam sem valores e foram removidas do conjunto de dados;

5. Substituir os valores faltantes na coluna “koi_tce_delivname” pela média de todos os valores encontrados: Esta coluna não possui variáveis do tipo float, portanto foram inseridas dummies, que são variáveis que transformam dados categóricos em valores binários, ou seja, os valores passam a ser 0 ou 1;

6. Dividir o conjunto de dados em duas partes: X (variáveis independentes) e y somente a coluna alvo “koi_disposition” (variável dependente);

7. Criar dois conjuntos de dados: 70% dos dados foram aleatoriamente separados para treinamento e 30% para teste;

8. Submeter os dados de treinamento e teste a uma função de escalonamento (*StandardScaler*), que padroniza os dados dentro da mesma escala de valor;

9. Utilizar uma semente para geração do mesmo número aleatório em todas as execuções, visando garantir que os modelos sempre classifiquem os dados sobre os mesmos números aleatórios.

Após o tratamento, o conjunto de dados de treinamento passou a ter 3.178 linhas e 42 colunas. Para a coluna alvo “koi_disposition”, 1.589 planetas foram confirmados e 1.589 classificados como candidatos.

3.2 Seleção dos algoritmos Ensemble

Os algoritmos Ensemble foram selecionados conforme uma variação entre método de fusão, dependência e abordagem de treinamento. Os cinco algoritmos escolhidos para a pesquisa foram: Adaboost, Random Forest, Random Subspace Methods, Extremely Randomized Trees e Stacking. A Tabela 2 exibe as diferenças entre as abordagens de formação de cada algoritmo Ensemble.

Tabela 2 – Algoritmos Ensemble escolhidos.

Algoritmo Ensemble	Método de fusão	Dependência	Abordagem de treinamento
Adaboost	Pesagem	Dependente	Manipulação de entrada
Random Forest	Pesagem	Independente	Hibridação Ensemble
Random Subspace Methods	Pesagem	Independente	Hibridação Ensemble
Extremely Randomized Trees	Pesagem	Independente	Particionamento
Stacking	Meta-aprendizado	Independente	Aprendizado manipulado

Fonte - Elaborado pelo autor (2023).

3.3 Treinamento e testes do conjunto de dados

A primeira implementação foi realizada para comparar o resultado de diferentes algoritmos, tanto Ensemble como não. Para essa primeira avaliação, o método de holdout foi utilizado para treinamento e testes do KOI. O motivo de utilização deste método foi a maior simplicidade e rapidez devido à quantidade de algoritmos testados. Para as próximas execuções individuais dos cinco algoritmos Ensemble escolhidos, o método de validação cruzada mencionado por (RASCHKA, 2018) foi utilizado. Este, dividiu o conjunto de dados em dez partes e aleatoriamente repetiu o processo cinco vezes. Foi estimada a média de desempenho entre as dez divisões do conjunto de dados e seu desvio padrão se baseando nas métricas: exatidão, sensibilidade, precisão, especificidade e nota F1.

3.4 Implementação dos algoritmos Ensemble

Experimentalmente, a avaliação dos cinco algoritmos Ensemble, foi conduzida mediante implementações sequenciais dos algoritmos, sendo dividas em duas partes. A primeira classificação dos exoplanetas foi realizada com valores padrão dos parâmetros do algoritmo. Na segunda implementação, foi utilizada uma função que testou aleatoriamente 100 combinações de valores de determinados parâmetros do algoritmo. No final deste teste, a função encontrou os valores dos parâmetros que, dentre estas 100 combinações, atingem o maior valor percentual de exatidão. Em seguida, estes valores foram passados para os seus respectivos parâmetros e uma nova classificação dos exoplanetas ocorreu. Para cada implementação, uma matriz de confusão foi gerada para levantar a quantidade de acertos e erros dos modelos.

3.5 Avaliação dos algoritmos

Os modelos de Aprendizado de Máquina foram avaliados de acordo com algumas métricas de medição de desempenho mencionadas no artigo (THARWAT, 2020). Foi criada uma matriz de confusão com os valores: TP (verdadeiro positivo) exoplanetas confirmados e classificados como confirmados, TN (verdadeiro negativo) exoplanetas candidatos e classificados como candidatos, FP (falso positivo) exoplanetas candidatos, mas classificados como confirmados e FN (falso negativo) exoplanetas confirmados, mas classificados como candidatos. Através do resultado da matriz de confusão, foi possível calcular as seguintes métricas a serem analisadas de cada algoritmo. Os resultados das Equações (3.1) a (3.5) são multiplicados por 100 e são apresentados em porcentagem (%).

Exatidão (Exa): é a razão das observações preditas corretamente. A Equação (3.1) é a relação entre amostras corretamente classificadas pelo total de amostras do conjunto de dados. Contudo, esta métrica não é adequada quando as classes estão desbalanceadas,

ou seja, quando o valor alvo possui diferente proporção de valor. Por exemplo, em um conjunto de dados de exoplanetas, 90% são de planetas confirmados e apenas 10% de planetas candidatos. Portanto, a métrica de exatidão será afetada, pois a exatidão será alta, mas o modelo terá falta de poder preditivo e as predições serão erradas (MAHARAJ, 2021).

$$Exa = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{3.1}$$

Sensibilidade (Sen): é o percentual dos dados preditos corretamente que pertencem às classes positivas. Ela é também chamada de recall. Indiretamente se refere a capacidade do modelo de aleatoriamente identificar uma observação que pertence a uma classe positiva.

$$Sen = \frac{(TP)}{(TP + FN)} \tag{3.2}$$

Especificidade (Spec): é a razão de instâncias negativas classificadas corretamente pelo total de instâncias negativas classificadas correta e incorretamente. Ela é similar a sensibilidade, mas com foco nas classes negativas.

$$Spec = \frac{(TN)}{(TN + FP)} \tag{3.3}$$

Precisão (Prec): é a razão de instâncias positivas classificadas corretamente pelo o total de instâncias positivas preditas correta e incorretamente.

$$Prec = \frac{(TP)}{(TP + FP)} \tag{3.4}$$

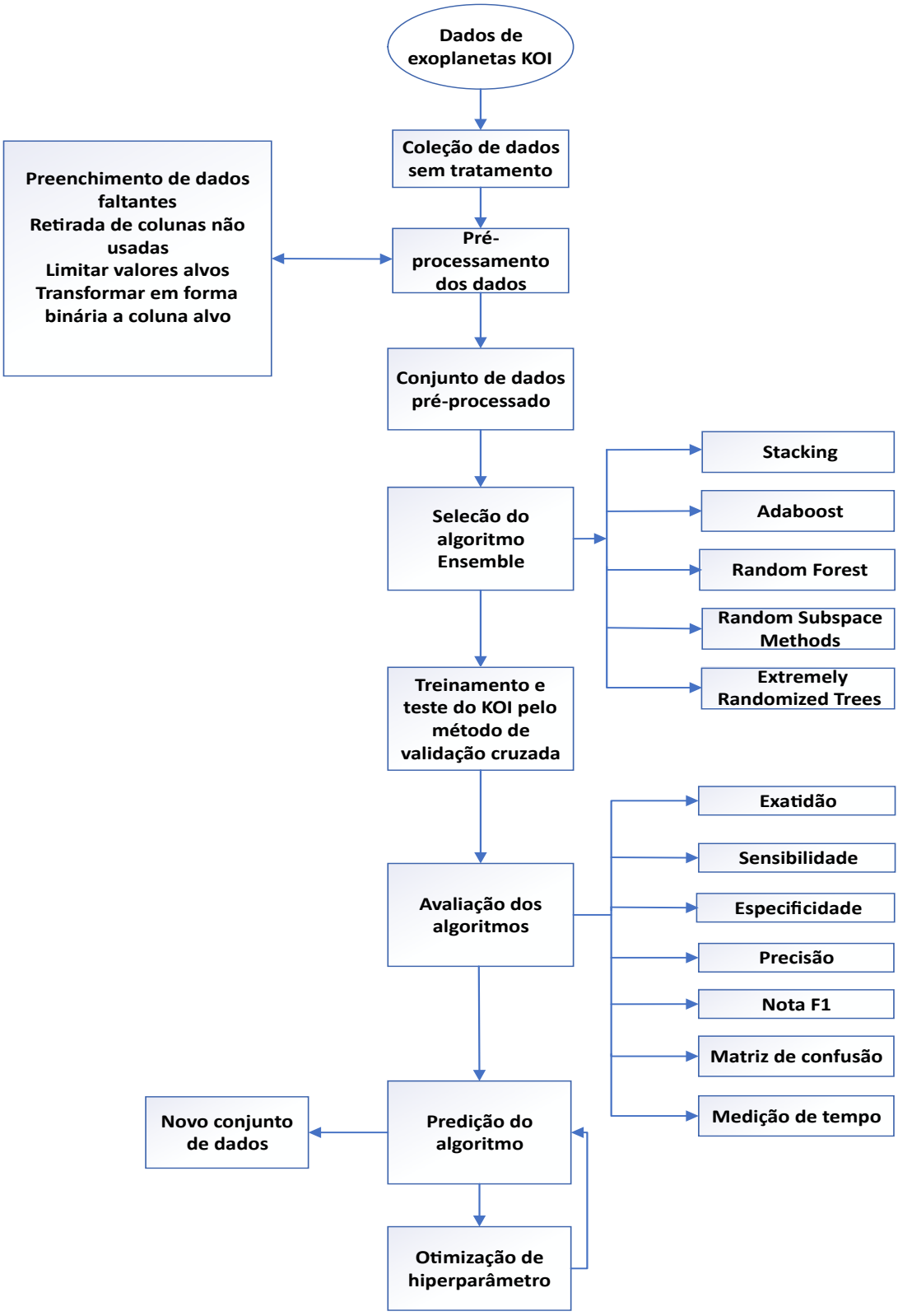
Nota F1 (F1): é a média harmônica entre precisão e especificidade. Quanto maior a nota F1 melhor. Um modelo terá uma nota F1 alta se as instâncias preditas como positivas forem realmente positivas (precisão) e não classificar instâncias positivas como negativas (especificidade) conforme a Equação (3.5).

$$F1 = \frac{2TP}{(2TP + FP + FN)} \tag{3.5}$$

Experimentalmente, o tempo consumido por cada implementação foi medido em segundos. No início do processo de treinamento de cada algoritmo foi inserida uma função para contar o tempo gasto até o final da predição.

A Figura 12 representa o fluxograma da metodologia.

Figura 12 – Fluxograma do processo de predição de exoplanetas com algoritmos Ensemble.



Fonte - Elaborado pelo autor (2023).

4 Resultados

Para estudar a atuação dos algoritmos de classificação, foi primeiramente realizada a predição dos exoplanetas com vários algoritmos de classificação ensemble e não ensemble, sem a alteração dos valores de seus parâmetros. Estes algoritmos são exibidos na Tabela 3. O resultado da classificação dos dados foi analisado e avaliado em cada algoritmo.

Tabela 3 – Algoritmos de classificação Ensemble e não Ensemble.

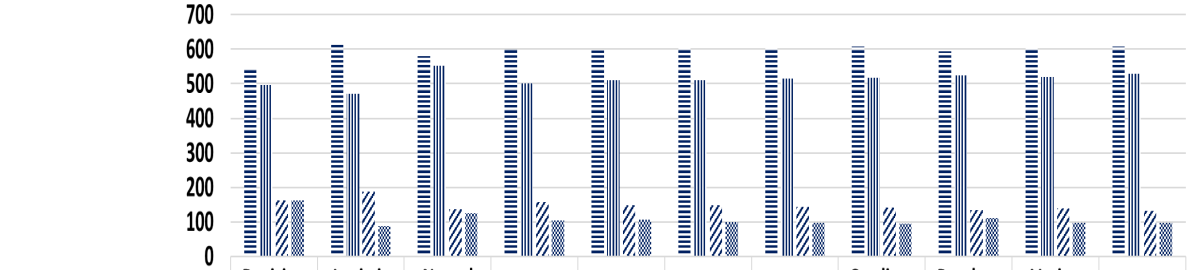
Algoritmo não Ensemble	Algoritmo Ensemble
Logistic Regression	Random Forest
Decision Tree	Adaboost
Neural Network	Gradient Boosting
	XGBoost
	LightGBM
	CatBoost
	Voting Classifier
	Stacking

Fonte - Elaborado pelo autor (2023).

Uma matriz de confusão foi criada com as 1.363 instâncias do conjunto de dados de teste para cada algoritmo. A combinação das matrizes de confusão para todos os algoritmos está ilustrada na Figura 13. A técnica Logistic Regression apesar de não ser Ensemble, foi a que mais acertou os dados positivos, com um valor de 615 instâncias classificadas corretamente como exoplaneta confirmados. Contudo, nota-se que este algoritmo teve o pior desempenho na classificação dos dados negativos em não exoplanetas, com 472 acertos. Portanto, a precisão foi uma das menores de todos os métodos. O método ensemble Stacking e Voting Classifier, tiveram o melhor desempenho entre todos os algoritmos na exatidão, classificando corretamente 82,61% dos dados.

O gráfico da Figura 14 mostra a porcentagem de desempenho atingida por cada algoritmo nas métricas.

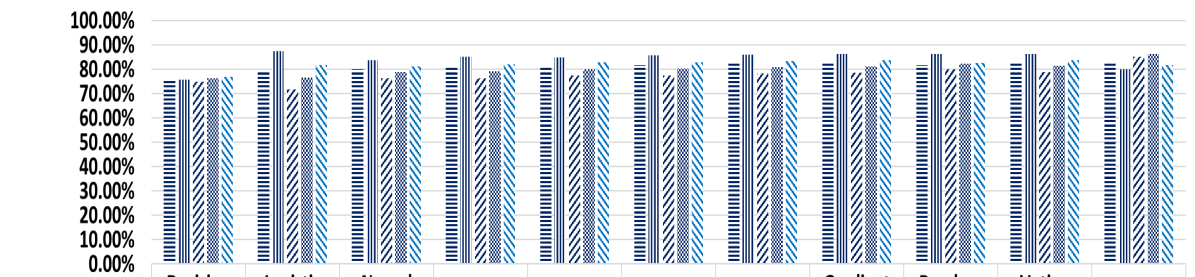
Figura 13 – Matriz de confusão para os algoritmos.



	Decision tree	Logistic Regression	Neural Network	AdaBoost	XGBoost	LightGBM	CatBoost	Gradient boosting	Random Forest	Voting classifier	Stacking
TP (Verdadeiro positivo)	542	615	579	599	597	604	606	608	593	606	607
TN (Verdadeiro negativo)	496	472	552	502	511	510	516	518	524	519	528
FP (Falso positivo)	163	187	137	157	148	149	143	141	135	140	131
FN (Falso negativo)	162	89	125	105	107	100	98	96	111	98	97

Fonte - Elaborado pelo autor (2023).

Figura 14 – Resultado em porcentagem das métricas de desempenho dos algoritmos.



	Decision tree	Logistic Regression	Neural Network	AdaBoost	XGBoost	LightGBM	CatBoost	Gradient boosting	Random Forest	Voting classifier	Stacking
Exatidão	76.01%	79.75%	80.12%	80.78%	81.29%	81.73%	82.32%	82.54%	81.73%	82.61%	82.61%
Sensibilidade	75.71%	87.36%	83.67%	85.09%	84.80%	85.80%	86.08%	86.22%	86.36%	86.36%	79.97%
Especificidade	74.96%	71.62%	76.18%	76.18%	77.54%	77.39%	78.30%	78.60%	79.97%	78.91%	85.08%
Precisão	76.36%	76.68%	78.95%	79.23%	80.13%	80.21%	80.91%	81.15%	82.16%	81.39%	86.38%
Nota F1	76.96%	81.67%	81.12%	82.06%	82.91%	82.91%	83.41%	83.61%	82.62%	83.67%	81.64%

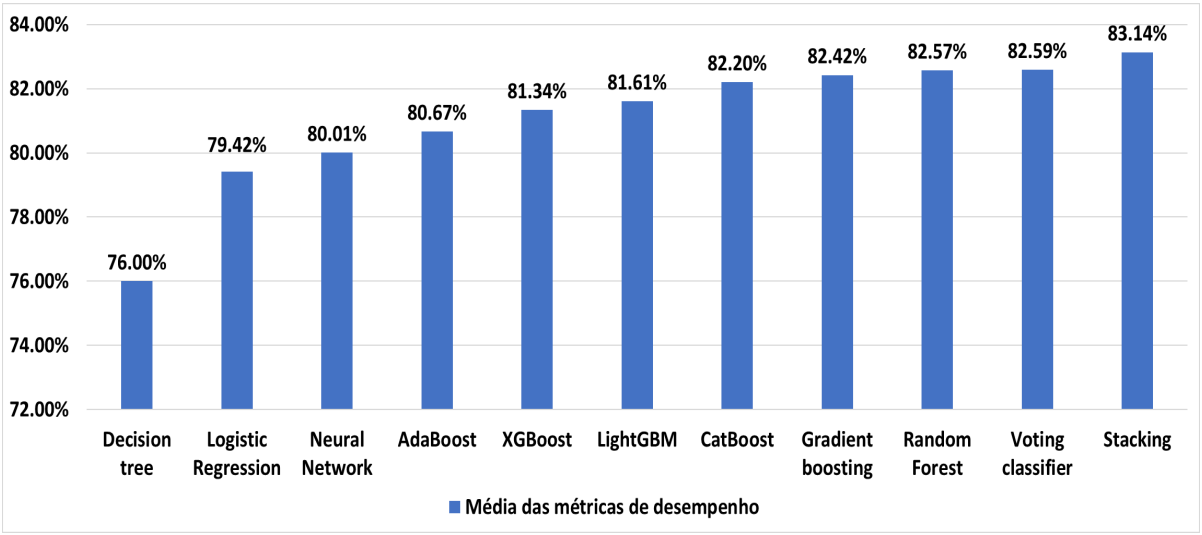
Fonte - Elaborado pelo autor (2023).

Os algoritmos de classificação Logistic Regression, Decision Tree, Neural Network que não são Ensemble, tiveram uma porcentagem relativamente menor na exatidão, especificidade e nota F1. O algoritmo Decision Tree teve o pior desempenho na exatidão com 76,01%. Os algoritmos Ensemble Stacking e Voting Classifier, tiveram o melhor desempenho, com 82,61% na exatidão. O algoritmo Random Forest, que utiliza de várias árvores de decisão para realizar a predição, também teve um excelente desempenho, com 81,73% na exatidão. Todos os algoritmos Ensemble tiveram a exatidão maior do que 80%, confirmando a eficiência do método em unir predições de vários métodos para encontrar o melhor resultado. Pode-se observar que o algoritmo não Ensemble Logistic Regression

teve a maior porcentagem de sensibilidade, com 87,36%, contudo a porcentagem da precisão foi a segunda menor, com 76,68%, reduzindo assim o valor final da nota F1 para 81,67%.

Na Figura 15 temos a média dos resultados de predição entre todas as métricas. O gráfico foi criado em ordem crescente, ficando evidenciado o melhor desempenho dos métodos Ensemble na predição dos dados de exoplanetas, quando comparado aos algoritmos não Ensemble. Por exemplo, o método Decision Tree teve uma média de 76% e o método Ensemble Stacking teve a maior média com 83,14%.

Figura 15 – Média em porcentagem do desempenho para cada algoritmo.



Fonte - Elaborado pelo autor (2023).

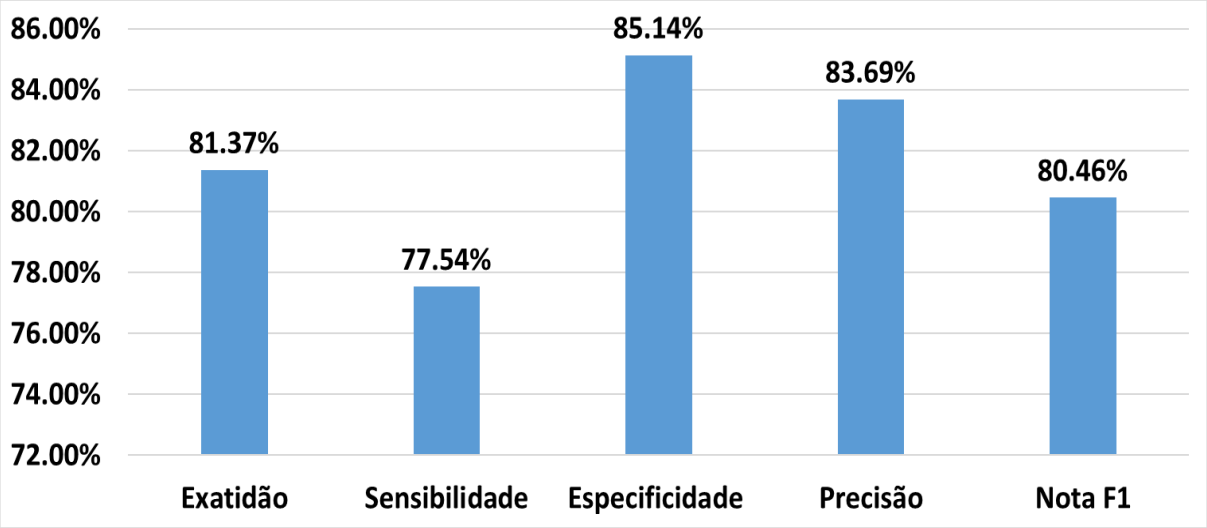
Após a execução do estudo comparativo inicial foi realizada uma implementação individual dos algoritmos Ensemble, que tiveram seus parâmetros alterados de modo a verificar se haveria uma melhora na eficiência da classificação dos exoplanetas. Os algoritmos Ensemble implementados foram: Adaboost, Random Forest, Random Subspace Methods, Extremely Randomized Trees e Stacking.

4.1 Implementação do Adaboost

Na primeira implementação, o algoritmo Adaboost predisse, corretamente, 502 exoplanetas confirmados (TP) e 599 exoplanetas candidatos (TN), atingindo um valor de exatidão de 81,37%. O desempenho do algoritmo na predição é exibido pela Figura 16. A matriz de confusão com as respectivas porcentagens de cada métrica é apresentada na Figura 17.

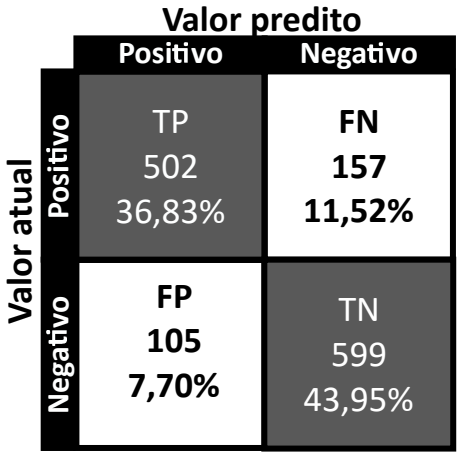
Os valores dos únicos dois parâmetros do Adaboost: número de estimadores e taxa de aprendizado foram alterados da seguinte forma:

Figura 16 – Resultado da predição com valores padrão dos parâmetros do Adaboost.



Fonte - Elaborado pelo autor (2023).

Figura 17 – Matriz de confusão com valores padrão dos parâmetros do Adaboost.



Fonte - Elaborado pelo autor (2023).

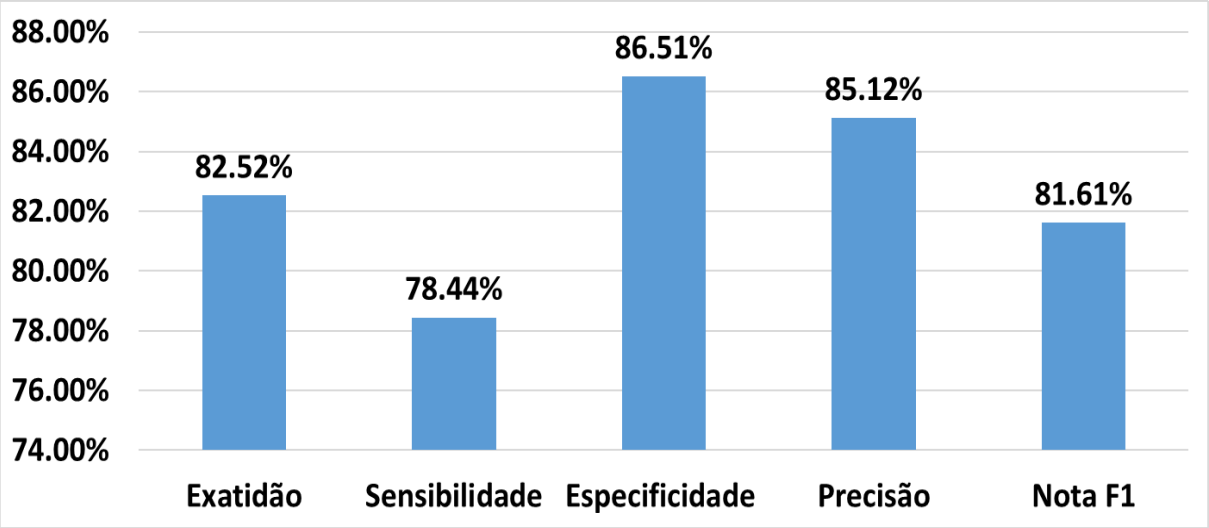
- **Número de estimadores:** o valor padrão deste parâmetro é de 50 algoritmos como estimadores. Foi criado um vetor para armazenar 40 valores de 10 a 1000 estimadores.
- **Taxa de aprendizado:** o valor padrão deste parâmetro é 1. Foi criado um vetor com 7 valores pré-definidos [0,1, 1, 2, 5, 10, 50, 100].

A combinação possível dos valores dos parâmetros é de 280 (40 valores estimadores × 7 valores de taxa de aprendizado). Contudo, foram realizadas 100 iterações, visando deixar o processo de comparação dos valores dos parâmetros mais rápido.

Na segunda implementação, os parâmetros foram, então, ajustados da seguinte forma: o número de estimadores foi alterado de 50 para 974 e a taxa de aprendizado foi

alterada de 1,0 para 0,1. Com a alteração dos valores padrão dos parâmetros, a porcentagem da exatidão aumentou em relação à primeira execução e atingiu 82,52%. Portanto, houve um aumento de 1,15%. Para os novos valores dos parâmetros, a Figura 18 exibe o resultado de todas as métricas.

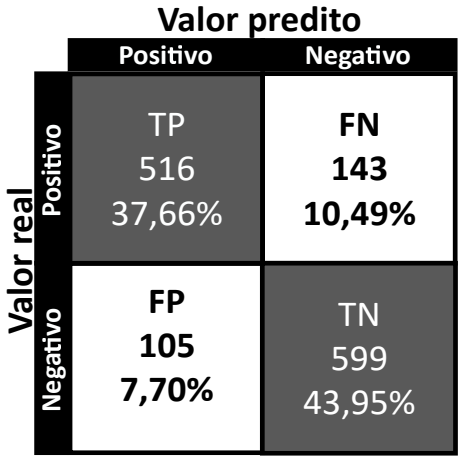
Figura 18 – Resultado da predição com a melhor combinação de valores parâmetros para melhorar a exatidão na predição dos dados do algoritmo Adaboost.



Fonte - Elaborado pelo autor (2023).

A matriz de confusão, apresentada na Figura 19, apresenta o resultado de classificação do algoritmo que classificou, corretamente, 516 exoplanetas confirmados (TP) e 599 exoplanetas candidatos (TN).

Figura 19 – Matriz de confusão com a melhor combinação de valores de parâmetros para melhorar a exatidão na predição dos dados do Adaboost.



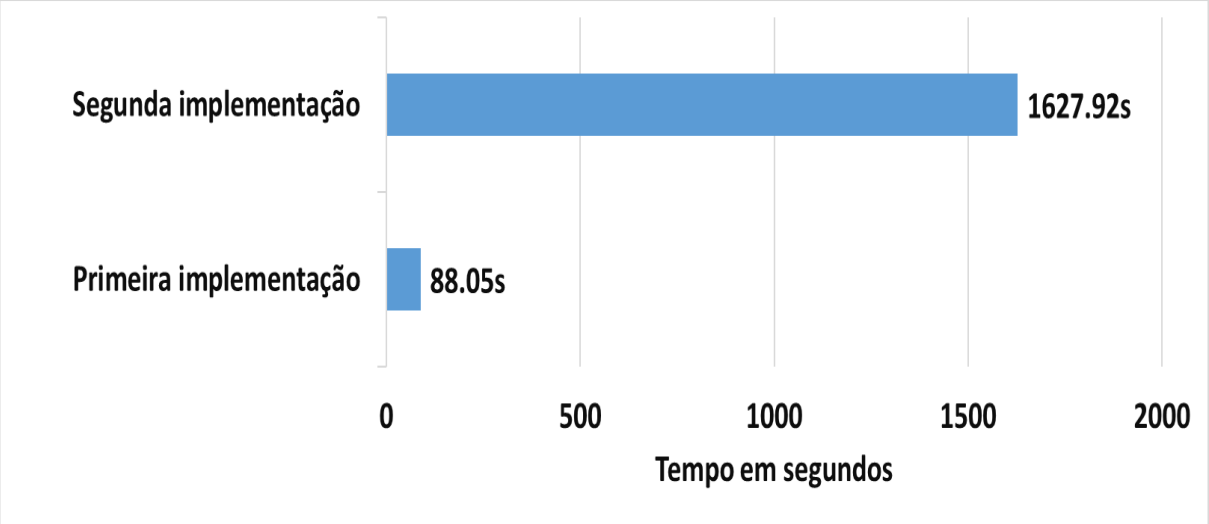
Fonte - Elaborado pelo autor (2023).

Esse resultado demonstrou que o algoritmo tem uma eficiência maior quando o valor do parâmetro número de estimadores é maior do que o valor padrão. Contudo,

em relação ao parâmetro taxa de aprendizado, o algoritmo obtém um melhor resultado quando este fica menor que seu valor padrão. Todas as outras métricas de desempenho, isto é, sensibilidade, especificidade, precisão e nota F1 tiveram um resultado melhor.

O tempo gasto pelo algoritmo nas duas implementações é exibido pela Figura 20. Na segunda implementação, o tempo consumido pelo algoritmo foi 18 vezes maior.

Figura 20 – Tempo consumido pelo algoritmo Adaboost nas duas implementações.



Fonte - Elaborado pelo autor (2023).

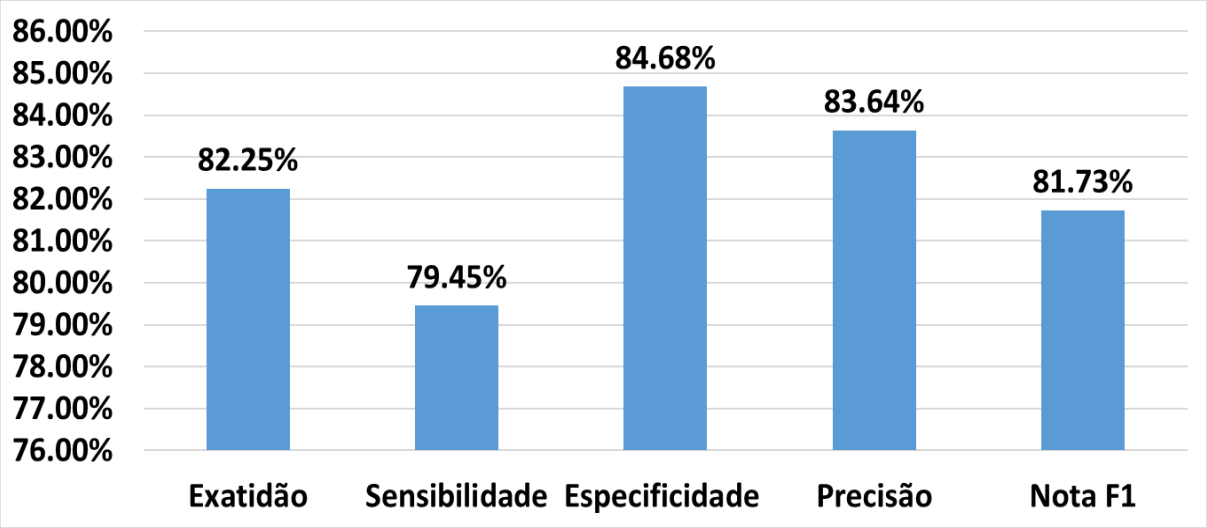
4.2 Implementação do Random Forest

Na primeira implementação, o algoritmo Random Forest predisse, corretamente, corretamente 514 exoplanetas confirmados (TP) e 593 exoplanetas candidatos (TN), atingindo um valor de exatidão de 82,25%. O desempenho do algoritmo na predição é exibido pela Figura 21. A matriz de confusão com as respectivas porcentagens de métrica é apresentada na Figura 22.

O Random Forest possui 18 parâmetros e, arbitrariamente, apenas quatro foram selecionados para a alteração de seus valores padrão. Os parâmetros selecionados foram: o número de estimadores, o número máximo de variáveis, a profundidade máxima das árvores e o *criterion*. Os valores nos parâmetros foram alterados da seguinte forma:

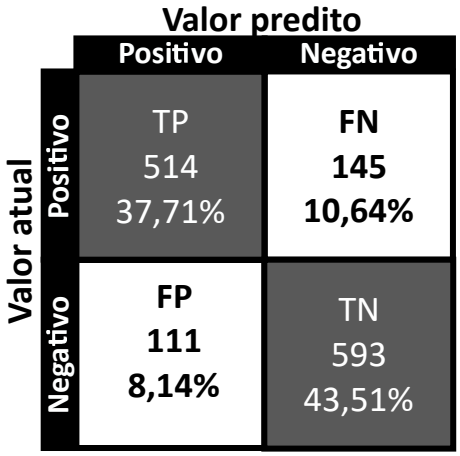
- **Número de estimadores:** o valor padrão deste parâmetro é de 100 árvores. Foi criado um vetor para armazenar 10 valores entre 200 a 2000.
- **Número máximo de variáveis:** o valor padrão é representado pela função *sqrt*. Foi incluído um vetor contendo todos os possíveis valores [*sqrt*, *log2*, *None*].

Figura 21 – Resultado da predição com valores padrão dos parâmetros do Random Forest.



Fonte - Elaborado pelo autor (2023).

Figura 22 – Matriz de confusão com valores padrão dos parâmetros do Random Forest.



Fonte - Elaborado pelo autor (2023).

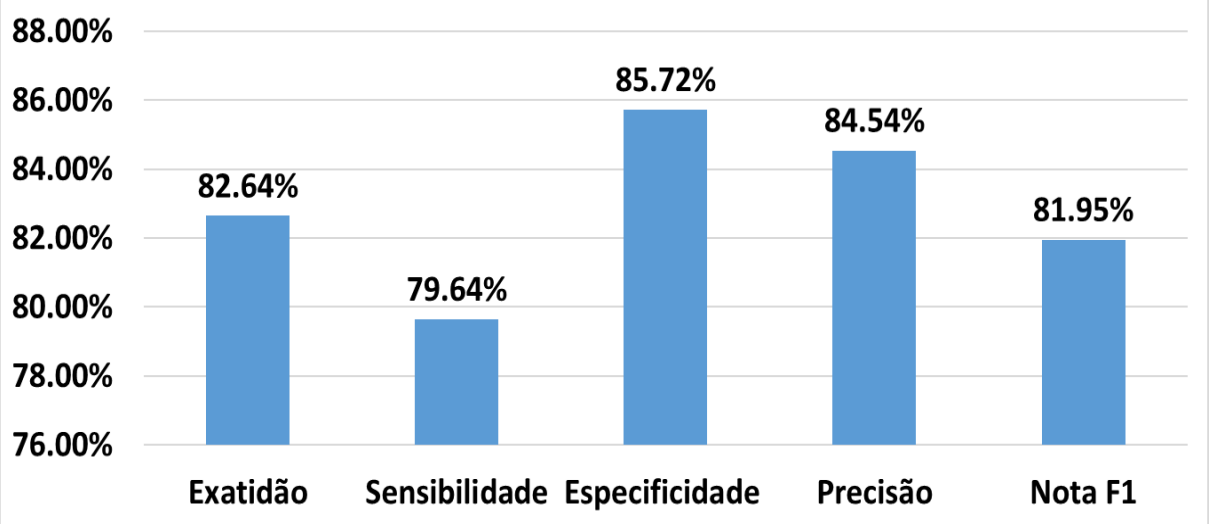
- **Profundidade máxima das árvores:** o valor padrão é *None*. Foi incluído um vetor com os valores [*None*, 4,5,6,7,8].
- **`criterion:`** o valor padrão é representado pela função *gini*. Foi incluído o vetor contendo todos os possíveis valores [*gini*, *entropy*, *log_loss*].

A combinação possível dos valores dos parâmetros é de 540 (10 valores de estimadores × 3 valores de funções para calcular o número de variáveis × 6 valores passados no vetor × 3 valores de funções). Contudo, foram realizadas 100 iterações, visando deixar o processo de comparação dos valores dos parâmetros mais rápido.

Na segunda implementação, os parâmetros foram alterados da seguinte forma: número de estimadores = 1600 e *criterion* = *entropy*. Os outros parâmetros: número

máximo de variáveis e profundidade máxima da árvore permaneceram, respectivamente, com seus valores iniciais: *sqrt* e *none*. A exatidão atingiu um desempenho de 82,64%, ou seja, um aumento de 0,39% em relação ao modelo anterior, conforme Figura 23. Evidencia-se que o aumento do número de árvores, que chegou a 1600, proporcionou uma melhora na exatidão do modelo.

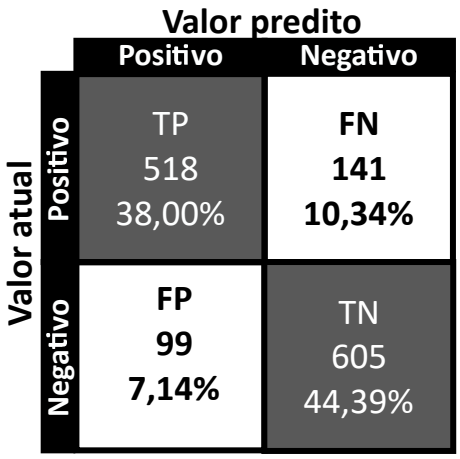
Figura 23 – Resultado da predição com a melhor combinação de valores parâmetros para melhorar a exatidão na predição dos dados do algoritmo Random Forest.



Fonte - Elaborado pelo autor (2023).

A sensibilidade aumentou de 78,44% para 79,45% e a especificidade de 84,68% para 85,72%. As métricas precisão e nota F1 aumentaram em relação à primeira implementação também. A matriz de confusão, mostrada na Figura 24, demonstra um aumento de 4 acertos nos exoplanetas confirmados (TP) e mais 12 acertos nos exoplanetas candidatos (TN).

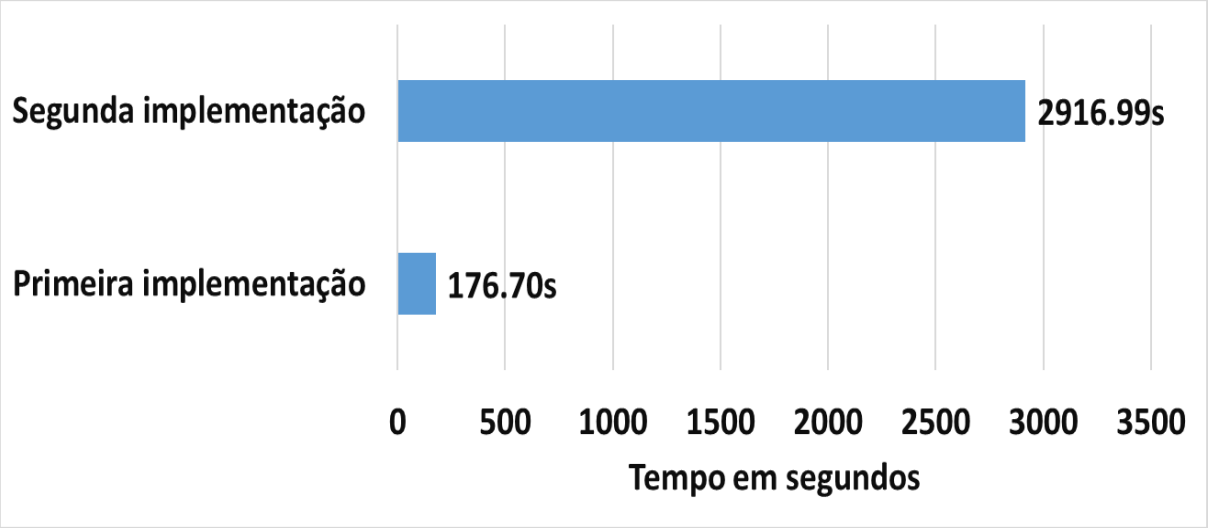
Figura 24 – Matriz de confusão com a melhor combinação de valores de parâmetros para melhorar a exatidão na predição dos dados do Random Forest.



Fonte - Elaborado pelo autor (2023).

O tempo gasto pelo algoritmo nas duas implementações é exibido pela Figura 25. Na segunda implementação, o tempo consumido pelo algoritmo foi 16 vezes maior.

Figura 25 – Tempo consumido pelo algoritmo Random Forest nas duas implementações.



Fonte - Elaborado pelo autor (2023).

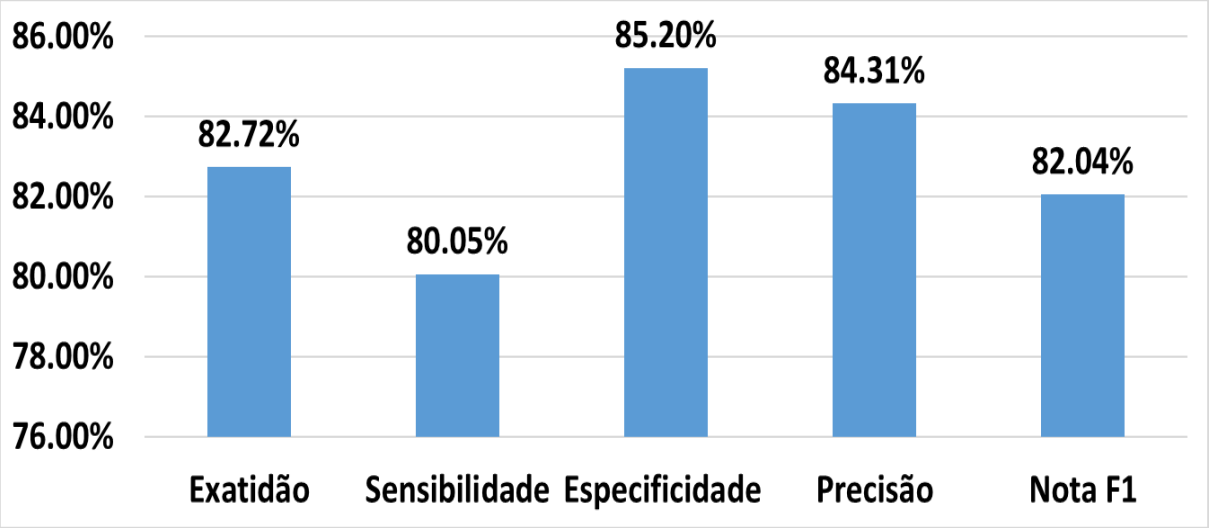
4.3 Implementação do Stacking

Na primeira implementação, o algoritmo Stacking conseguiu predizer corretamente 527 exoplanetas confirmados (TP) e 599 exoplanetas candidatos (TN), atingindo um valor de exatidão de 82,72%. O desempenho do algoritmo na predição é exibido pela Figura 26. A matriz de confusão com as respectivas porcentagens de métrica é apresentada na Figura 27. Na primeira implementação do algoritmo Stacking, foi necessário selecionar valores para o parâmetro estimadores. Pois, este parâmetro não possui valores iniciais. Arbitrariamente, dois estimadores foram escolhidos: Random Forest e Gradient Boosting, os quais são algoritmos Ensemble. Estes dois estimadores foram adicionados com os valores padrão de seus parâmetros.

Na segunda implementação, os valores de alguns dos parâmetros dos algoritmos estimadores escolhidos Random Forest e Gradient Boosting foram alterados. Os parâmetros alterados no Random Forest foram os mesmos implementados anteriormente, isto é: número de estimadores = 1600, número máximo de variáveis = *sqrt*, profundidade máxima das árvores = *none* e o parâmetro *criterion* = *entropy*. Os parâmetros com valores alterados no algoritmo Gradient Boosting foram: número de estimadores = 400 e taxa de aprendizado = 0,1.

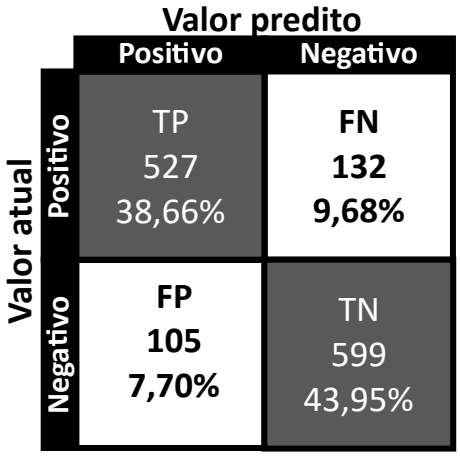
A exatidão atingiu um desempenho de 83,03%, um aumento de 0,31% em relação ao modelo anterior, conforme Figura 28. Todas as métricas de desempenho atingiram um maior resultado em relação à primeira implementação, exceto a precisão que teve uma

Figura 26 – Resultado da predição com valores padrão dos parâmetros do Stacking.



Fonte - Elaborado pelo autor (2023).

Figura 27 – Matriz de confusão com valores padrão dos parâmetros do Stacking.



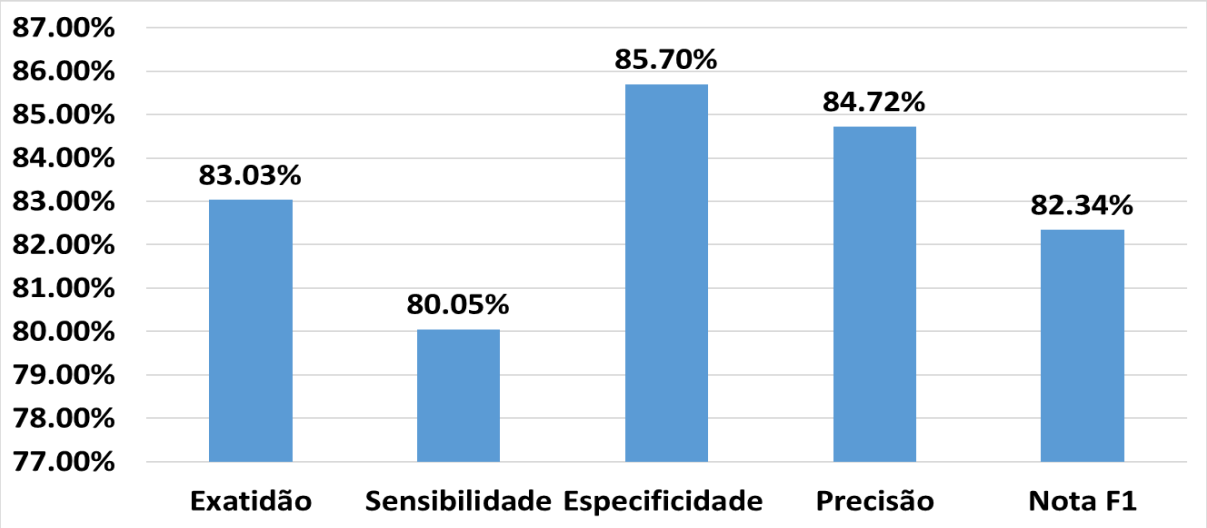
Fonte - Elaborado pelo autor (2023).

leve diminuição de 84,54% para 84,31%. A matriz de confusão, mostrada na Figura 29, demonstra um aumento de 1 acerto nos exoplanetas candidatos (TN).

Como o algoritmo Stacking pode utilizar dois ou mais estimadores para realizar as predições, foram treinados inicialmente nove algoritmos de classificação, alguns são Ensemble e outros são não Ensemble. Este treinamento ocorreu sem o método de validação cruzada, pois ele teve o objetivo de realizar uma análise mais rápida nos algoritmos propostos.

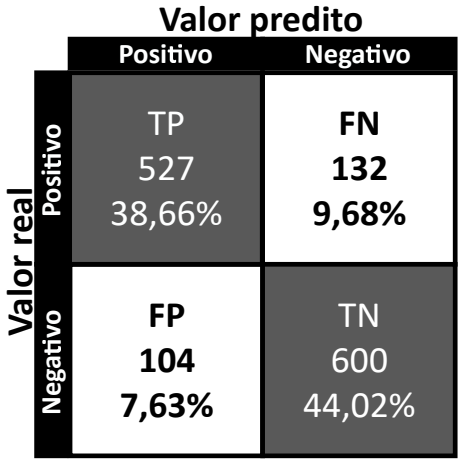
O resultado da predição dos algoritmos treinados é apresentado na Figura 30 em ordem crescente de desempenho na exatidão. O algoritmo Gradient Boosting atingiu o melhor desempenho entre todos, classificando corretamente com uma exatidão de 82.53%. Nota-se que os algoritmos não Ensemble, Naive Bayes, K Nearest Neighbor e Decision Tree

Figura 28 – Resultado da predição com combinação de valores do parâmetros dos algoritmos estimadores para melhorar a exatidão na predição dos dados do algoritmo Stacking.



Fonte - Elaborado pelo autor (2023).

Figura 29 – Matriz de confusão com estimadores com combinação de valores de parâmetros para o Stacking.

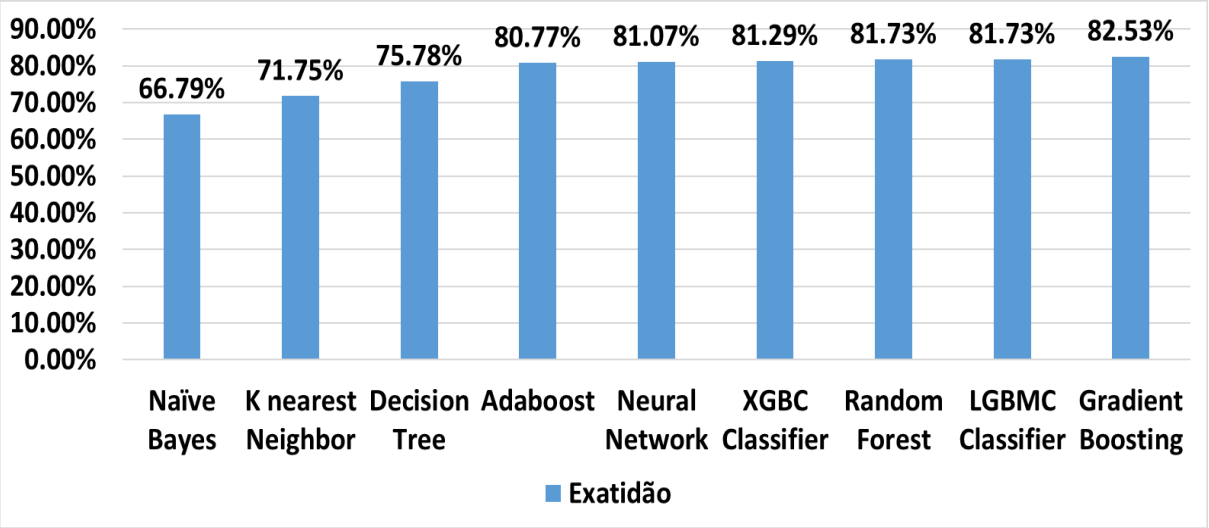


Fonte - Elaborado pelo autor (2023).

tiveram os três piores resultados. Isto evidencia que os algoritmos Ensemble possuem um desempenho superior para predizer os dados de exoplanetas mesmo com os valores padrão de seus parâmetros.

Cinco combinações de algoritmos como estimadores foram escolhidas para o parâmetro estimadores. O critério de escolha dos algoritmos foi embasado na avaliação e variação da união de algoritmos Ensemble e também a adição de alguns algoritmos não Ensemble. Cada combinação é designada como Estimador n, em que n assume os valores de 1 a 5. As combinações escolhidas são:

Figura 30 – Resultado da exatidão para os nove algoritmos de classificação para serem estimadores do Stacking.

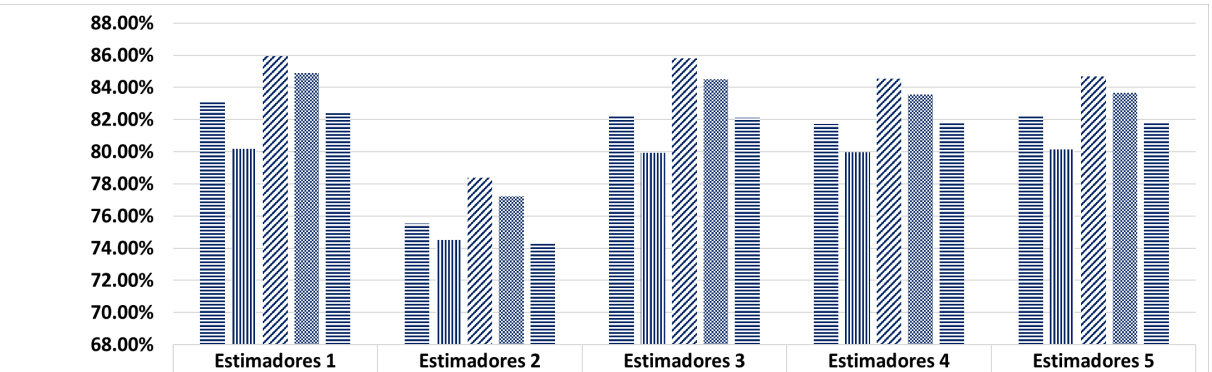


Fonte - Elaborado pelo autor (2023).

1. LGBMC Classifier + Gradient Boosting: dois algoritmos Ensemble com desempenho alto na exatidão;
2. Neural Network + K Nearest Neighbor + Decision Tree: três algoritmos não Ensemble;
3. Random Forest + Gradient Boosting + LGBMC Classifier + XGBC Classifier + Adaboost: cinco algoritmos Ensemble;
4. Random Forest + Naive Bayes: algoritmo Ensemble e não Ensemble;
5. Random Forest + Adaboost: algoritmos Ensemble.

A Figura 31 apresenta o desempenho de cada Estimador na classificação dos dados. O Estimador 1 obteve o melhor desempenho entre todos na exatidão, com o resultado de 83,08% de exatidão e atingiu uma sensibilidade, especificidade, precisão e nota F1 maior do a atingida na primeira implementação do Random Subspace method.

Figura 31 – Resultado da exatidão na classificação com 5 combinações de valores de estimadores para o algoritmo Stacking.

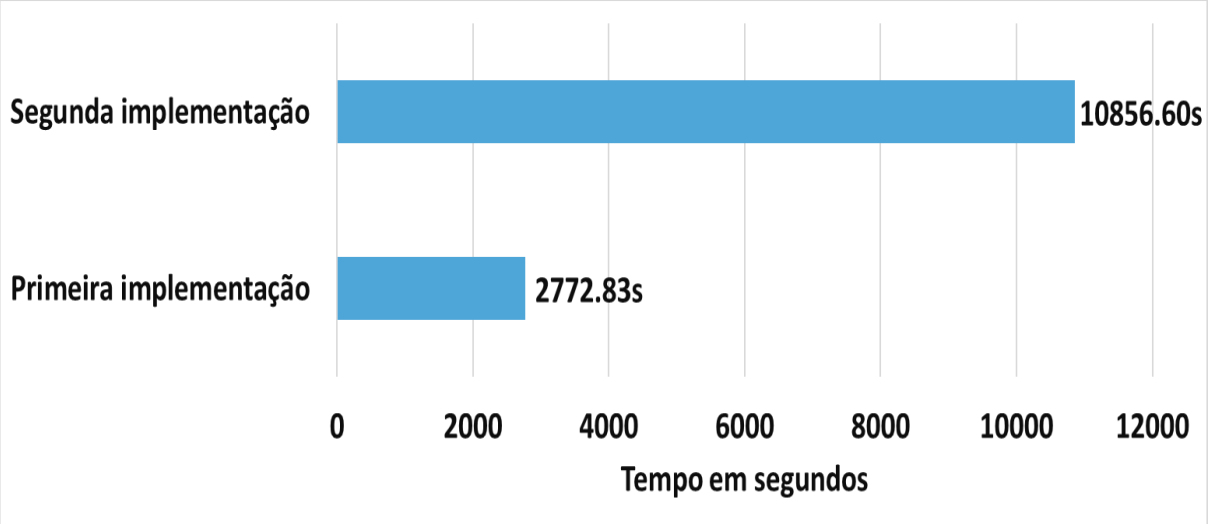


	Estimadores 1	Estimadores 2	Estimadores 3	Estimadores 4	Estimadores 5
Exatidão	83.08%	75.54%	82.29%	81.73%	82.24%
Sensibilidade	80.18%	74.53%	79.94%	79.97%	80.13%
Especificidade	85.96%	78.40%	85.79%	84.54%	84.67%
Precisão	84.88%	77.21%	84.51%	83.57%	83.67%
Nota F1	82.41%	74.25%	82.12%	81.76%	81.77%

Fonte - Elaborado pelo autor (2023).

O tempo gasto pelo algoritmo nas duas implementações é exibido pela Figura 32. Na segunda implementação, o tempo consumido pelo algoritmo foi 3 vezes maior.

Figura 32 – Tempo consumido pelo algoritmo Stacking nas duas implementações.



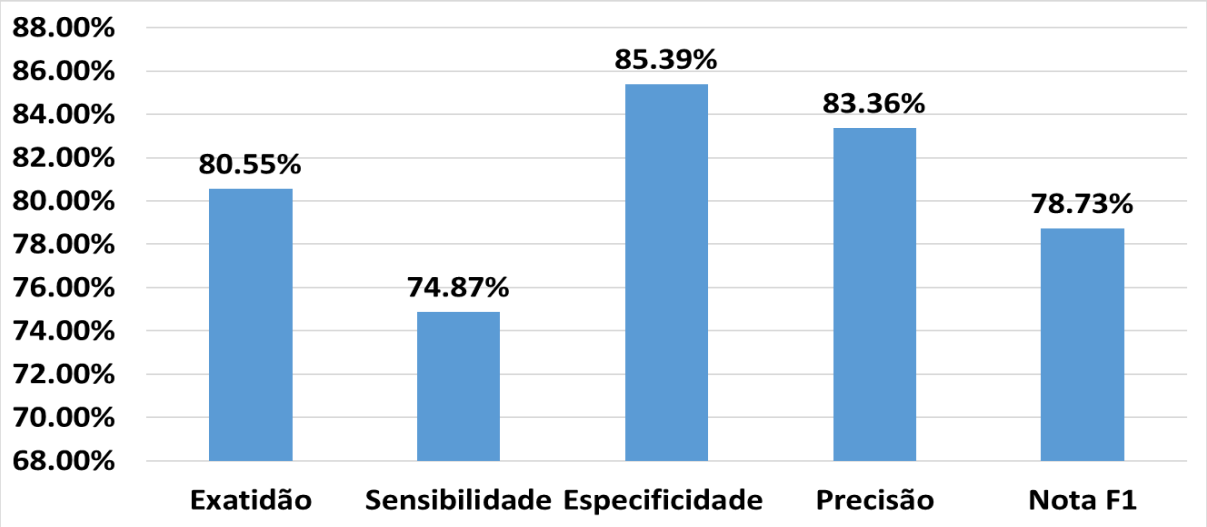
Fonte - Elaborado pelo autor (2023).

4.4 Implementação do Random Subspace Method

Na primeira implementação, o algoritmo Random Subspace Method conseguiu predizer corretamente 478 exoplanetas confirmados (TP) e 602 exoplanetas candidatos (TN), atingindo um valor de exatidão de 80,55%. O desempenho do algoritmo na predição

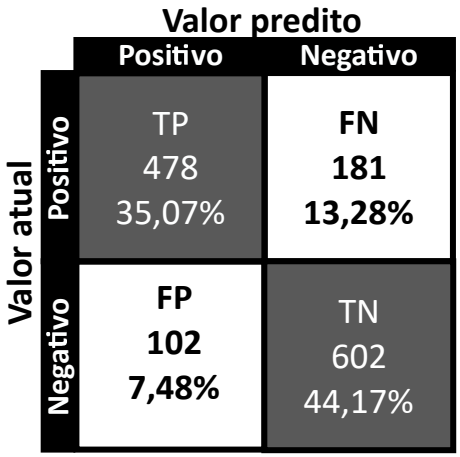
é exibido pela Figura 33. A matriz de confusão com as respectivas porcentagens de métrica é apresentada na Figura 34.

Figura 33 – Resultado da predição com valores padrão dos parâmetros do Random Subspace Method.



Fonte - Elaborado pelo autor (2023).

Figura 34 – Matriz de confusão com valores padrão dos parâmetros do Random Subspace Method.



Fonte - Elaborado pelo autor (2023).

O Random Subspace Method possui doze parâmetros. Três desses parâmetros foram, arbitrariamente, selecionados para terem seus valores alterados. Os parâmetros selecionados são: número de estimadores, número máximo de amostras e número máximo de variáveis. Os valores nos parâmetros foram alterados da seguinte forma:

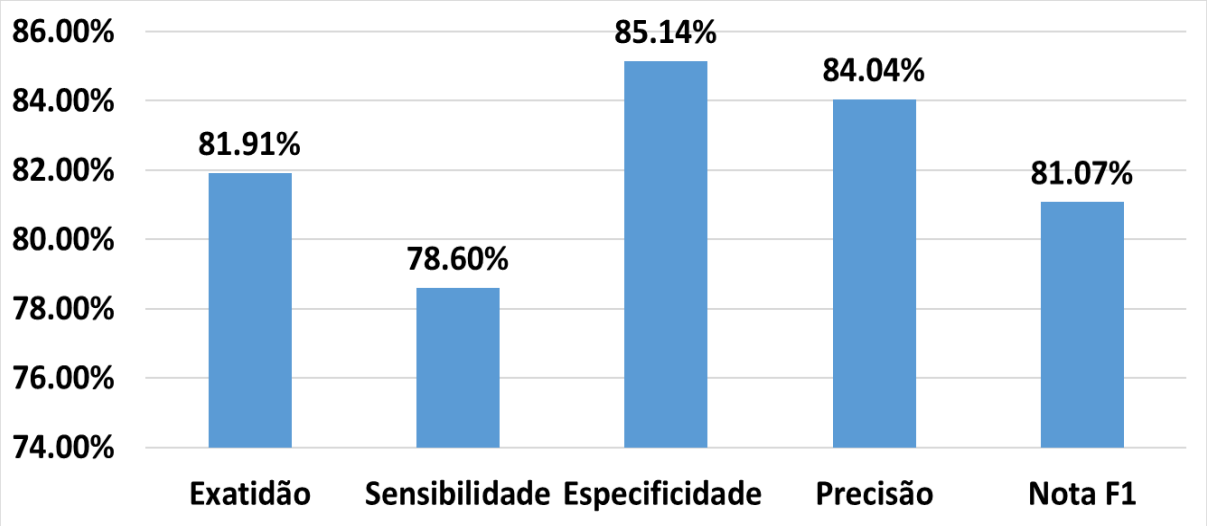
- **Número de estimadores:** o valor padrão deste parâmetro é de 10. Foi criado um vetor para armazenar 5 valores entre 10 e 1000.

- **Número máximo de amostras:** o valor padrão é 1. Foi incluído um vetor com os valores [0.1,1.0,10].
- **Número máximo de variáveis:** o valor padrão é 1. Foi incluído um vetor com os valores [0.1,1.0,10].

A combinação possível destes parâmetros é de 45 (5 valores de estimadores × 3 valores de amostras × 3 valores máximo de variáveis). Contudo, foram realizadas 100 iterações, visando deixar o processo de comparação dos valores dos parâmetros mais rápido.

Na segunda implementação, os valores dos parâmetros com o melhor desempenho de exatidão de 81,91% foram: número de estimadores alterado com o valor 1000, número máximo de amostras com o valor 0,1 e número máximo de variáveis permaneceu com o valor padrão 1. Nota-se que, para aumentar o desempenho do algoritmo na predição dos dados, são necessários mais estimadores e menos amostras e variáveis. O resultado para todas as métricas é ilustrado pela Figura 35 e a Figura 36 mostra a matriz de confusão gerada. O desempenho do algoritmo foi superior em relação à classificação de exoplanetas confirmados (TP), em relação ao resultado anterior, com a adição de 29 exoplanetas classificados corretamente. Contudo, o algoritmo classificou 5 planetas candidatos (TN) a menos do que na primeira implementação. Para as outras métricas de desempenho, com exceção da especificidade, o algoritmo atingiu um melhor resultado em todas quando comparado a primeira implementação.

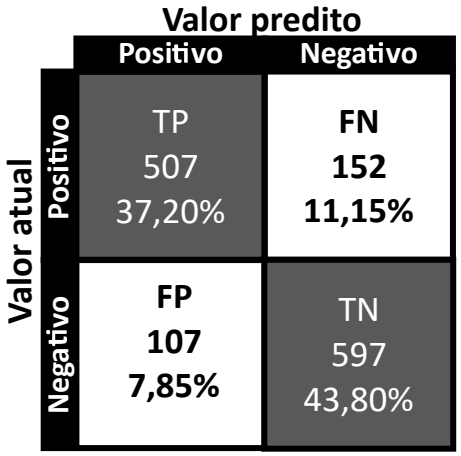
Figura 35 – Resultado da predição com a melhor combinação de valores parâmetros para melhorar a exatidão na predição dos dados do algoritmo Random Subspace Method.



Fonte - Elaborado pelo autor (2023).

O algoritmo Random Subspace method possui a opção de alterar o seu parâmetro estimador, cujo valor padrão é uma árvore de decisão, por outros algoritmos. Portanto,

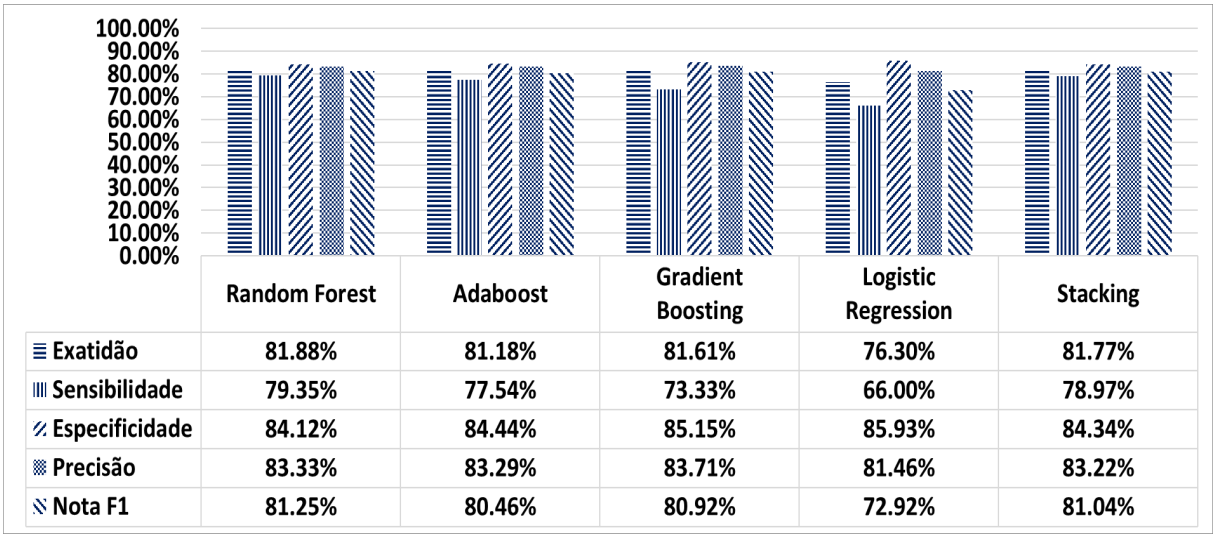
Figura 36 – Matriz de confusão com a melhor combinação de valores de parâmetros para melhorar a exatidão na predição dos dados do Random Subspace Method.



Fonte - Elaborado pelo autor (2023).

considerando este aspecto, o algoritmo foi implementado cinco vezes, cada uma delas com um algoritmo diferente atuando como estimador. Os algoritmos usados para esta finalidade foram: Random Forest, Adaboost, Gradient Boosting, Logistic Regression e Stacking. Os resultados de predição dos estimadores do Random Subspace são apresentados na Figura 37.

Figura 37 – Resultado da predição com diferentes estimadores do Random Subspace Method



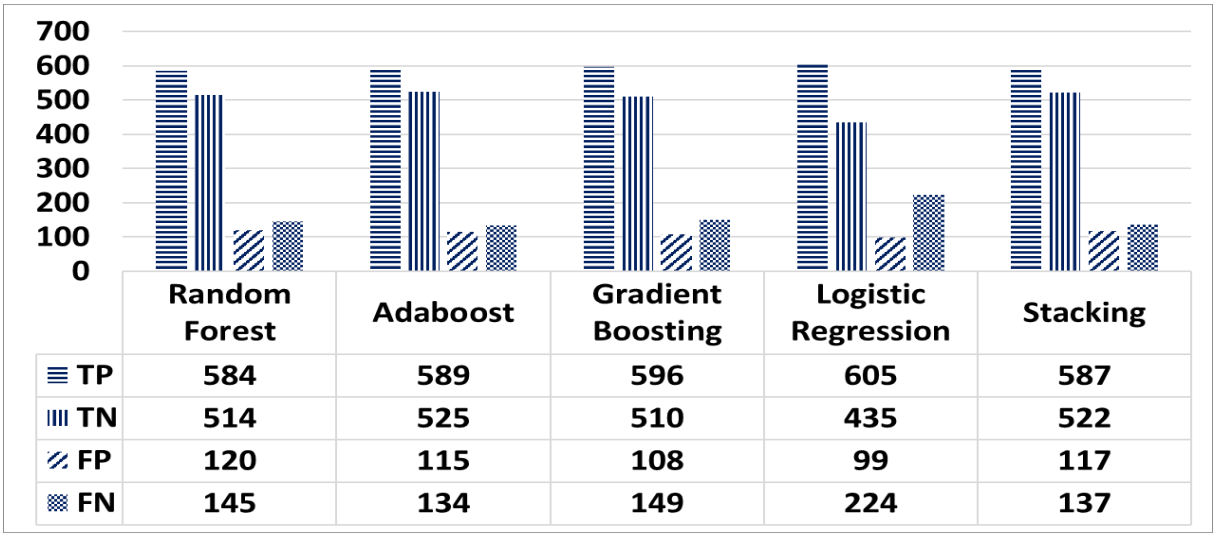
	Random Forest	Adaboost	Gradient Boosting	Logistic Regression	Stacking
Exatidão	81.88%	81.18%	81.61%	76.30%	81.77%
Sensibilidade	79.35%	77.54%	73.33%	66.00%	78.97%
Especificidade	84.12%	84.44%	85.15%	85.93%	84.34%
Precisão	83.33%	83.29%	83.71%	81.46%	83.22%
Nota F1	81.25%	80.46%	80.92%	72.92%	81.04%

Fonte - Elaborado pelo autor (2023).

O Random Subspace Method, com o Random Forest atuando como estimador padrão, obteve o melhor desempenho para: a exatidão com 81,88%, a sensibilidade com 79,35% e nota F1 com 81,25%. O Random Subspace, com Logistic Regression como estimador, que não é Ensemble, teve a melhor especificidade com 85,93%. O Gradient Boosting,

conseguiu a melhor precisão com 83,71%. A Figura 38 exibe os resultados de classificação da matriz de confusão de cada algoritmo quando atuando como estimador do Random Subspace Method.

Figura 38 – Matriz de confusão de diferentes estimadores do Random Subspace Method

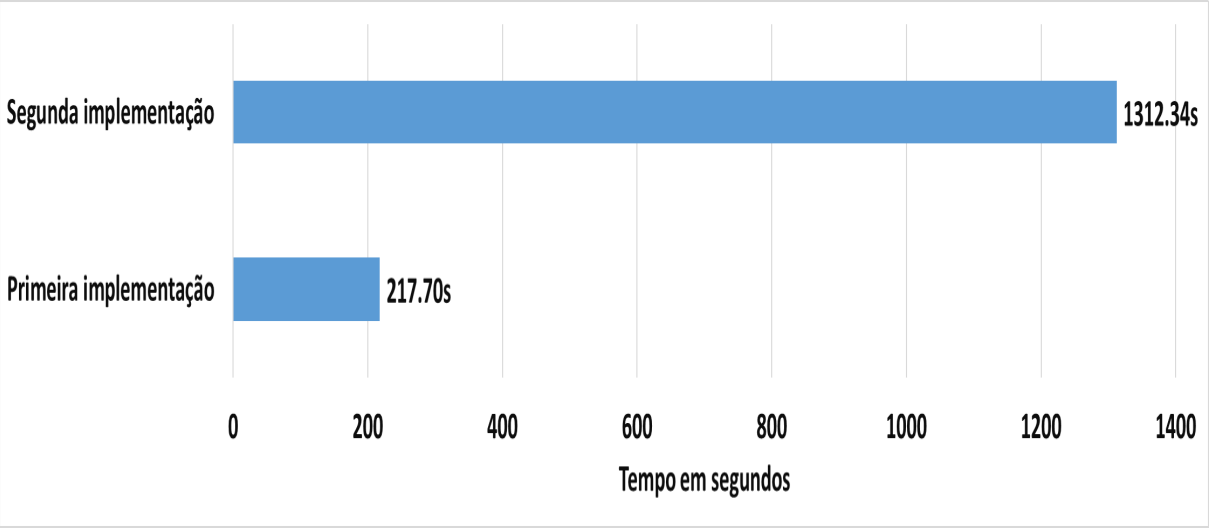


	Random Forest	Adaboost	Gradient Boosting	Logistic Regression	Stacking
TP	584	589	596	605	587
TN	514	525	510	435	522
FP	120	115	108	99	117
FN	145	134	149	224	137

Fonte - Elaborado pelo autor (2023).

O tempo gasto pelo algoritmo nas duas implementações é exibido pela Figura 39. Na segunda implementação, o tempo consumido pelo algoritmo foi 6 vezes maior.

Figura 39 – Tempo consumido pelo algoritmo Random Subspace Method nas duas implementações.

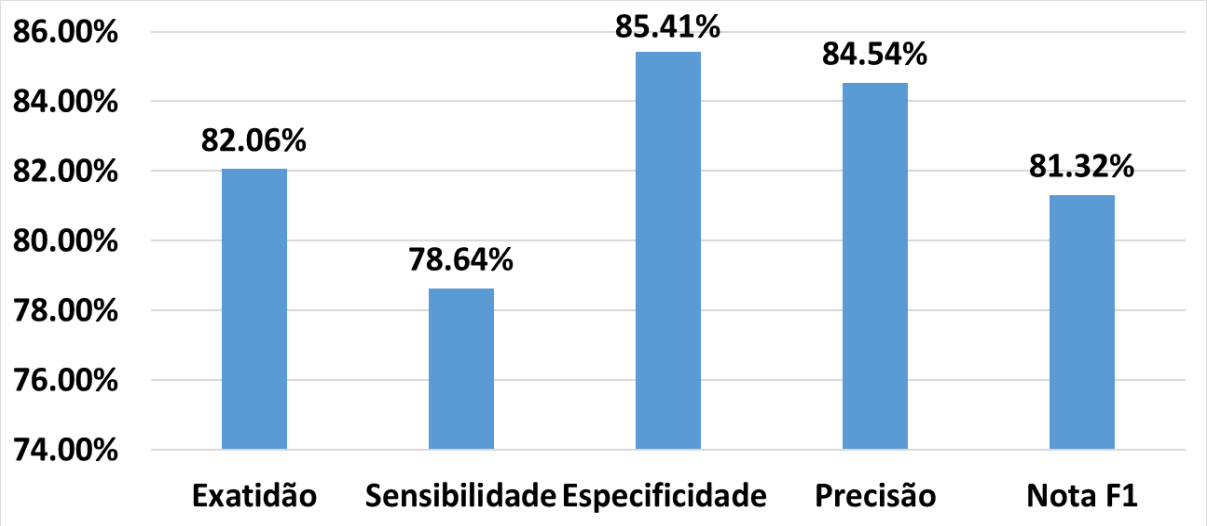


Fonte - Elaborado pelo autor (2023).

4.5 Implementação do Extremely Randomized Trees

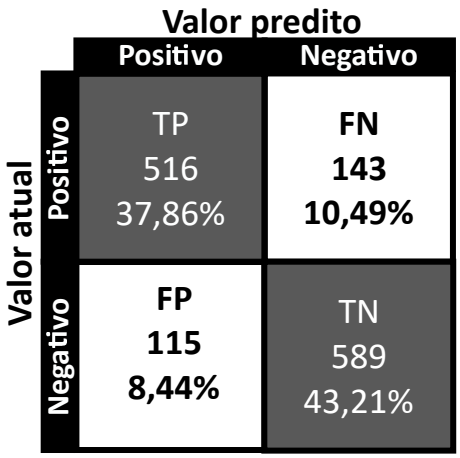
Na primeira implementação, o algoritmo Extremely Randomized Trees conseguiu predizer corretamente 516 exoplanetas confirmados (TP) e 589 exoplanetas candidatos (TN), atingindo um valor de exatidão de 82,06%. O desempenho do algoritmo na predição é exibido pela Figura 40. A matriz de confusão com as respectivas porcentagens de métrica é apresentada na Figura 41.

Figura 40 – Resultado da predição com valores padrão dos parâmetros do Extremely Randomized Trees.



Fonte - Elaborado pelo autor (2023).

Figura 41 – Matriz de confusão com valores padrão dos parâmetros do Random Subspace Method.



Fonte - Elaborado pelo autor (2023).

O algoritmo Extremely Randomized Trees possui 18 parâmetros. Quatro parâmetros foram selecionados arbitrariamente para modificação de seus valores padrão. Os

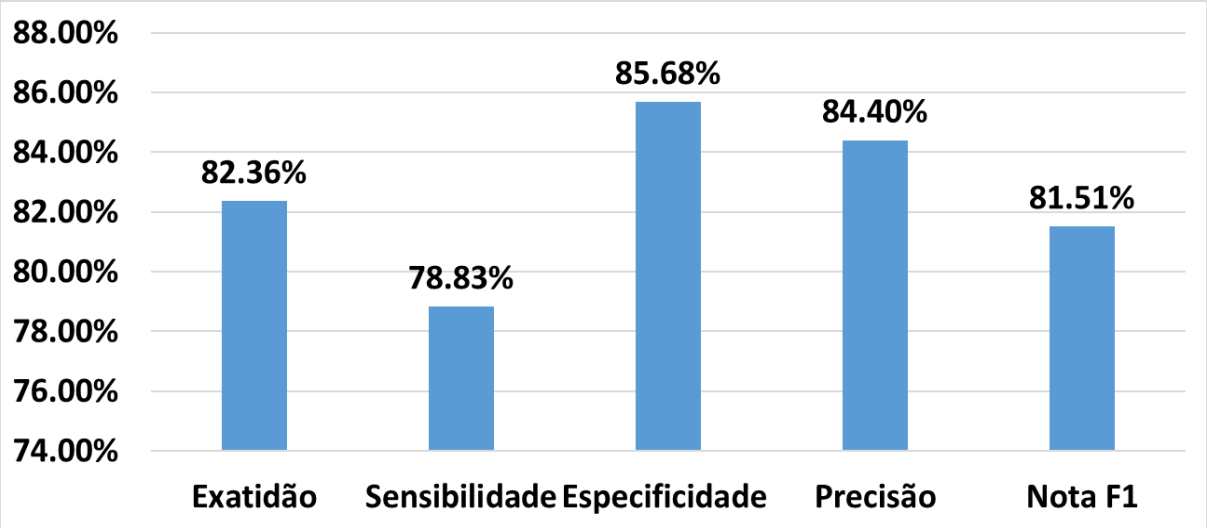
parâmetros selecionados são: número de estimadores, número máximo de variáveis, profundidade máxima da árvore e *criterion*. Os valores nos parâmetros foram alterados da seguinte forma:

- **Número de estimadores:** o valor padrão deste parâmetro é de 100. Foi criado um vetor para armazenar 5 valores aleatórios entre 10 e 1000.
- **Número máximo de variáveis:** o valor padrão é representado pela função *sqrt*. Foi incluído um vetor contendo os possíveis valores [*sqrt*, *log2*, *None*].
- **Profundidade máxima da árvore:** o valor padrão é *None*. Foi incluído um vetor com os valores [4, 5, 6, 7, 8, *None*].
- **Criterion:** o valor padrão é representado pela função *gini*. Foi incluído um vetor contendo os possíveis valores [*gini*, *entropy*].

A combinação possível destes parâmetros é de 180 (5 valores de estimadores $\times$ 3 valores de variáveis $\times$ 6 valores de profundidade máxima e 2 valores de funções). Contudo, foram realizadas 100 iterações, visando deixar o processo de comparação dos valores dos parâmetros mais rápido.

A Figura 42 apresenta o resultado para todas a métricas do Extremely Randomized Trees para a segunda implementação. Nesta implementação, os novos valores dos parâmetros, que resultaram em melhor desempenho para a exatidão com 82,36%, foram: número de estimadores com o valor 200 e *criterion* com o valor *entropy*. Os parâmetros: número máximo de variáveis e profundidade máxima da árvore permaneceram com seus valores padrões. As outras métricas, com exceção da precisão, apresentaram um pequeno aumento.

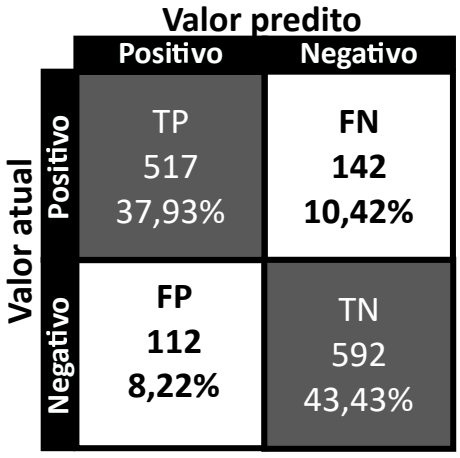
Figura 42 – Resultado da predição com a melhor combinação de valores parâmetros para melhorar a exatidão na predição dos dados do algoritmo Extremely Randomized Trees.



Fonte - Elaborado pelo autor (2023).

A Figura 43 apresenta a matriz de confusão gerada para a segunda implementação. O Extremely Randomized Trees, na condição mencionada, classificou mais sete exoplanetas candidatos como exoplanetas confirmados (TP). Contudo, classificou 4 exoplanetas confirmados(TP) a menos.

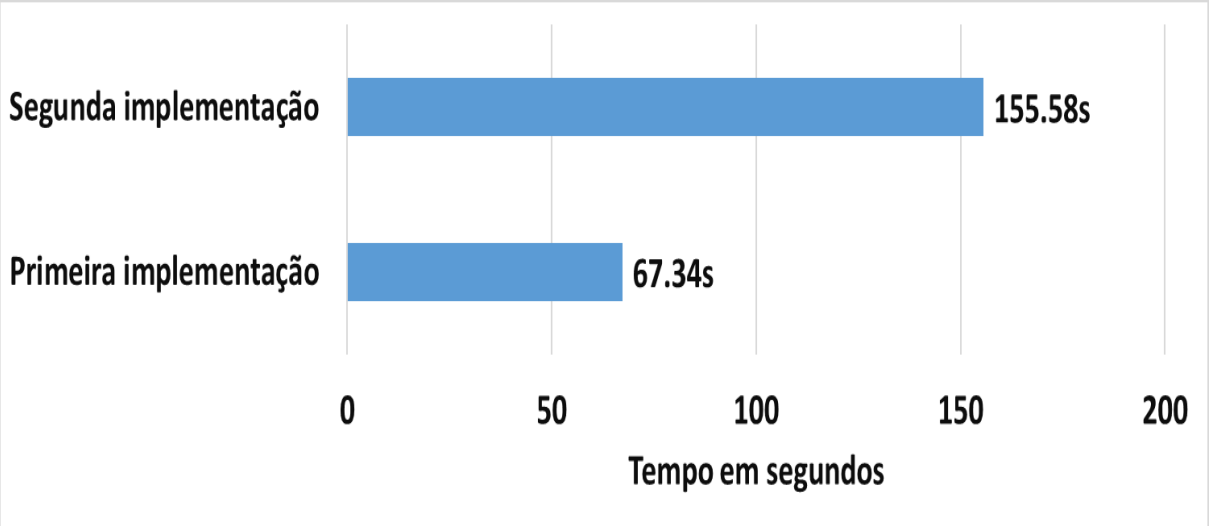
Figura 43 – Matriz de confusão com a melhor combinação de valores de parâmetros para melhorar a exatidão na predição dos dados do Extremely Randomized Trees.



Fonte - Elaborado pelo autor (2023).

O tempo gasto pelo algoritmo nas duas implementações é exibido pela Figura 44. Na segunda implementação, o tempo consumido pelo algoritmo foi 2,31 vezes maior.

Figura 44 – Tempo consumido pelo algoritmo Extremely Randomized Trees nas duas implementações.



Fonte - Elaborado pelo autor (2023).

4.6 Comparação de desempenho dos algoritmos Ensemble

A Tabela 4 exibe os resultados obtidos pelos algoritmos Ensemble com o valor de seus parâmetros alterados. Observa-se que o Stacking aingiu o melhor desempenho de exatidão e nota F1. Com base nos resultados previamente apresentados, este algoritmo se destaca como o mais versátil e com maior poder preditivo de todos. Sua versatilidade é devida a opção de utilizar como estimadores outros algoritmos Ensemble ou não Ensemble. Em sentido contrário, o mesmo não ocorre com outros algoritmos que possuem estimadores padrão. A título de exemplo, tem-se o algoritmo Random Forest que utiliza somente árvores de decisão.

Tabela 4 – Comparação das métricas dos algoritmos Ensemble.

Algoritmo Ensemble	Exatidão	Sensibilidade	Especificidade	Precisão	Nota F1
Adaboost	82,52%	78,44%	86,51%	85,12%	81,61%
Random Forest	82,64%	79,64%	85,72%	84,54%	81,91%
Stacking	83,08%	80,18%	85,96%	84,88%	82,41%
Random Subspace Method	81,91%	78,60%	85,14%	84,04%	81,07%
Extremely Randomized Trees	82,36%	78,83%	85,68%	84,40%	81,51%

Fonte - Elaborado pelo autor (2023).

5 Conclusão

Primeiramente foi implementado e executado onze algoritmos sobre a mesma base de dados de exoplanetas com o mesmo tratamento. Os oito algoritmos Ensemble tiveram um desempenho superior na média geral na predição. A métrica de exatidão ficou com um percentual maior que 80% para todos os algoritmos Ensemble, demonstrando que o modelo Ensemble possui um desempenho superior na predição, em comparação com os algoritmos não Ensemble. O algoritmo não Ensemble Logistic Regression, obteve uma sensibilidade classificando corretamente as classes positivas com uma porcentagem de 87,36%, maior entre todos os dez algoritmos. Porém, a especificidade classificando as classes negativas corretamente ficou com o menor valor, de 71,62%. Os algoritmos Ensemble ficaram com o percentual de acerto na métrica nota F1 maior que 81,50%. Uma matriz de confusão foi criada para todos os onze algoritmos mostrando a quantidade de acertos, para a situação verdadeiro positivo e verdadeiro negativo, erros em falso positivo e falso negativo.

Em resumo, a escolha dos cinco algoritmos Ensemble foi realizada com base na diversidade de seus modelos de funcionamento, os quais são: método de fusão, dependência e abordagem de treinamento. Esta escolha permitiu um resultado que integrasse diferentes estruturas destes algoritmos.

O Adaboost, primeiro algoritmo analisado, obteve uma exatidão de 81,37% com o valor padrão para seus parâmetros. Em sua segunda implementação foi utilizada a técnica denominada *hyperparameter tuning*. Com esta técnica testaram-se vezes diferentes valores dos parâmetros do algoritmo. Desse processo resultou uma combinação de valores de parâmetros que produziu a maior exatidão. Em consequência, o Adaboost teve seu número de estimadores e sua taxa de aprendizado alteradas e, com essa nova configuração, a exatidão aumentou mais de 1,1% e chegou a 82,52%. Este resultado demonstra que o algoritmo Adaboost tem capacidade para predizer com maior exatidão quando possui mais estimadores em sua estrutura. O Adaboost gastou apenas 88 segundos para predizer os dados na primeira implementação, porém na segunda demorou 1627 segundos.

O algoritmo Random Forest, que realiza a junção de árvores de decisão, obteve a terceira melhor exatidão dentre os cinco algoritmos quando executado com os valores padrão de seus parâmetros. Este algoritmo é simples de ser implementado e possui um bom desempenho preditivo. Portanto, este algoritmo é um dos mais utilizados dentre os algoritmos Ensemble. O Random Forest atingiu 82,25% de exatidão. Após a combinação de valores de parâmetros, a exatidão teve uma melhora de 0,39%. Dentre os valores dos parâmetros alterados, destaca-se que para o número de estimadores teve seu valor aumentado, significativamente, de 100 para 1600 árvores de decisão. Na primeira implementação,

o Random Forest gastou 176 segundos para predizer os dados, na segunda implementação esse tempo aumentou 16 vezes para 2916 segundos.

O algoritmo Stacking, em contraste ao Random Forest, une diferentes algoritmos e não somente árvores. O Stacking atingiu o melhor desempenho, entre todos os algoritmos, com o uso de valores de parâmetros padrão e obteve uma exatidão de 82,72%. Este resultado pode ser devido à capacidade do Stacking de utilizar uma camada de modelo-meta que trabalha com dados previamente gerados pelos algoritmos incluídos no Stacking. Consequentemente, o Stacking utiliza sua segunda camada de modelo-meta para combinar, da melhor forma, a capacidade de cada algoritmo escolhido em sua estrutura. Na segunda implementação, os valores de alguns dos parâmetros dos dois algoritmos escolhidos como estimadores Random Forest e Gradient Boosting foram alterados, assim o Stacking atingiu uma exatidão de 83,03%. Devido a esta capacidade do Stacking de permitir a união de diferentes estimadores, cinco combinações com diferentes algoritmos, agindo como estimadores, foram implementadas também. A combinação dos algoritmos LGBMC e o Gradient Boosting, ambos Ensemble, produziu uma exatidão maior que 83,08%. O resultado da classificação do Stacking foi o maior obtido por todos os modelos. Em contrapartida, o tempo consumido nas implementações foi o maior em relação a todos os algoritmos. Na primeira implementação ele gastou 2772 segundos e na segunda mais de 10856 segundos. Considerando o aspecto preditivo, o Stacking é o algoritmo mais recomendado para predição de exoplanetas. Contudo, caso o tempo seja uma variável importante, os outros algoritmos Ensemble do estudo possuem um melhor desempenho.

O algoritmo Random Subspace Method teve um desempenho parecido com o Adaboost e apresentou uma melhora superior a 1%. Este algoritmo, quando usou a combinação de valores de parâmetros, atingiu a exatidão de 81,91%. Este algoritmo possui uma estrutura simples, apesar de utilizar árvores de decisão e criar amostras aleatórias das variáveis. Ele não conseguiu atingir 82% de exatidão, mesmo com a alteração dos valores de seus parâmetros. O Random Subspace Method gastou 217 segundos na primeira implementação e 1312 segundos na segunda.

O algoritmo Extremely Randomized Trees conseguiu uma exatidão de 82,06% na primeira execução. Este algoritmo teve uma exatidão ótima, menor somente do que a do Stacking. Após ter os valores de seus parâmetros alterados, o Extremely Randomized Trees aumentou somente em 0,3% sua exatidão. Ele obteve o menor valor de melhoria em exatidão. Este algoritmo tem um processo aleatório, que ocorre em todo seu funcionamento e essa característica pode justificar esse resultado pouco positivo. Em relação ao tempo gasto para predizer os dados, o Extremely Randomized Trees obteve o melhor desempenho de tempo, gastando apenas 67 segundos e 155 segundos na primeira e segunda implementação respectivamente. Caso o tempo seja um aspecto importante, este algoritmo é recomendando para predição de exoplanetas.

Este estudo demonstra que os algoritmos Ensemble possuem melhor capacidade para predizer dados de exoplanetas. Todos os algoritmos conseguiram atingir uma exatidão maior que 80%. Porém, constata-se que é necessário realizar o ajuste de parâmetros para um melhor resultado. A Astronomia e Ciência da Computação poderiam utilizar mais o poder preditivo destes algoritmos para melhorar a descoberta de exoplanetas.

5.1 Limitações

O conjunto de dados foi tratado somente uma única vez. Nele foi realizado algumas técnicas de pré-processamento, como: normalização, eliminação de algumas colunas sem relevância, preenchimento de dados faltantes e transformação em valores binário para campos nominais. Nesta dissertação, todos os algoritmos receberam o mesmo tratamento de dados, não sendo possível avaliar outras formas de tratamento e consequentemente avaliar a possibilidade de se obter um desempenho superior em relação ao tratamento padrão.

5.2 Trabalhos futuros

Para próximas pesquisas, poderia ser utilizado como base de dados as curvas de luz das estrelas, ao invés de, dados quantitativos com as informações extraídas conforme executado nesta dissertação. Na literatura, muitos trabalhos realizam a predição dos dados com base em análises sobre os valores encontrados nestes dados.

Atualmente, existem telescópios mais sofisticados do que o Kepler lançados na atmosfera terrestre, por exemplo, o James Webb. Um próximo estudo poderia predizer os exoplanetas do conjunto de dados que está sendo criado por este telescópio.

Um próximo trabalho poderia estudar mais a fundo o algoritmo Stacking, propondo modificações em sua estrutura, com o propósito de melhorar o resultado preditivo e diminuir o tempo gasto na predição.

Anexos

ANEXO A – Definição das colunas do conjunto de dados de exoplanets KOI

A seguir algumas colunas do KOI são descritas. As definições de todas as colunas do KOI podem ser acessadas em (NASA, 2023).

- **kepid** Target identification number, as listed in the Kepler Input Catalog (KIC). The KIC was derived from a ground-based imaging survey of the Kepler field conducted prior to launch. The survey's purpose was to identify stars for the Kepler exoplanet survey by magnitude and color. The full catalog of 13 million sources can be searched at the MAST archive. The subset of 4 million targets found upon the Kepler CCDs can be searched via the Kepler Target Search form. The Kepler ID is unique to a target and there is only one Kepler ID per target.
- `KOI Name` A number used to identify and track a Kepler Object of Interest (KOI). A KOI is a target identified by the Kepler Project that displays at least one transit-like sequence within Kepler time-series photometry that appears to be of astrophysical origin and initially consistent with a planetary transit hypothesis. A KOI name has an integer and a decimal part of the format KNNNNN.DD. The integer part designates the target star; the two-digit decimal part identifies a unique transiting object associated with that star. It is not necessarily the planetary candidate listed in that order within a DV report, nor does it indicate the distance of the planet from the the host star relative to other planets in the system.
- **Vetting Status** The vetting status for this KOI delivery. Current possible states are ACTIVE and DONE. As vetting tests for the null hypothesis that a TCE is a planet are performed, the disposition of each KOI as either a planet candidate or false positive will be updated and, most importantly, may change over time. It is therefore critical that the scientific community not conduct sample completeness studies on KOI tables that remain ACTIVE. Active tables do, however, provide the latest information for community scientists interested in follow-up observations and disposition activities. After a period of activity, the classification of the KOI table will change from ACTIVE to DONE when all dispositions are judged as final and all model parameters have been updated appropriately. This will typically occur after a new delivery of TCEs to the archive based on a longer data baseline.
- **Date of Last Parameter Update** ate of the last parameter update for this KOI.

- **Disposition Using Kepler Data** The pipeline flag that designates the most probable physical explanation of the KOI. Typical values are FALSE POSITIVE, NOT DISPOSITIONED, and CANDIDATE. The value of this flag may change over time as the evaluation of KOIs proceeds to deeper levels of analysis using Kepler time-series pixel and light curve data, or follow-up observations. A not dispositioned value corresponds to objects for which the disposition tests have not yet been completed. A false positive has failed at least one of the tests described in Batalha et al. (2012). A planetary candidate has passed all prior tests conducted to identify false positives, although this does not a priori mean that all possible tests have been conducted. A future test may confirm this KOI as a false positive. False positives can occur when: 1) the KOI is in reality an eclipsing binary star, 2) the Kepler light curve is contaminated by a background eclipsing binary, 3) stellar variability is confused for coherent planetary transits, or 4) instrumental artifacts are confused for coherent planetary transits.

- **Disposition Score** value between 0 and 1 that indicates the confidence in the KOI disposition. For CANDIDATEs, a higher value indicates more confidence in its disposition, while for FALSE POSITIVEs, a higher value indicates less confidence in that disposition. The value is calculated from a Monte Carlo technique such that the score's value is equivalent to the frction of iterations where the Robovetter yields a disposition of CANDIDATE.

- **Not Transit-Like Flag** A KOI whose light curve is not consistent with that of a transiting planet. This includes, but is not limited to, instrumental artifacts, non-eclipsing variable stars, and spurious (very low SNR) detections.

- **Stellar Eclipse Flag** A KOI that is observed to have a significant secondary event, transit shape, or out-of-eclipse variability, which indicates that the transit-like event is most likely caused by an eclipsing binary. However, self-luminous, hot Jupiters with a visible secondary eclipse will also have this flag set, but with a disposition of PC.

- **Centroid Offset Flag** The source of the signal is from a nearby star, as inferred by measuring the centroid location of the image both in and out of transit, or by the strength of the transit signal in the target's outer (halo) pixels as compared to the transit signal from the pixels in the optimal (or core) aperture.

- **Ephemeris Match Indicates Contamination Flag** The KOI shares the same period and epoch as another object and is judged to be the result of flux contamination in the aperture or electronic crosstalk.

- **Disposition Provenance** Disposition Provenance

- **KOI Comment** A description of the reason why an object's disposition has been given as false positive. The following keywords are shorthand for certain criterion used to determine if a KOI is a false positive: APO: "Active Pixel Offset"The pixels showing the transit do not coincide with the target star, indicating that the transit is actually on a background object. Binary: Indicates the transit event is due to an eclipsing binary, not a planet. EB: Target is an eclipsing binary, or there is an unresolved background binary. odd-even: The depth of the even-numbered transits are statistically different than the depths of the odd-numbered transits; this is a sign of a background eclipsing binary. V-shaped: Likely a grazing eclipsing binary. SB1: Target star is a single-lined spectroscopic binary. SB2: Target star is a double-lined spectroscopic binary. A comment field may also contain a list of the minor flags as set by the Robovetter. See the documents for the DR 25 and DR 24 Robovetter KOI Flags for detailed descriptions.

- **Orbital Period (days)** The interval between consecutive planetary transits.

- **Transit Epoch (BJD - 2,454,833.0)** (BJD - 2,454,833.0) The time corresponding to the center of the first detected transit in Barycentric Julian Day (BJD) minus a constant offset of 2,454,833.0 days. The offset corresponds to 12:00 on Jan 1, 2009 UTC.

- **Transit Epoch in BJD** The time corresponding to the center of the first detected transit in Barycentric Julian Day (BJD).

- **Eccentricity** Eccentricity Value,454,833.0 days. The offset corresponds to 12:00 on Jan 1, 2009 UTC.

- **Long. of Periastron (deg)** Longitude of Periastron

- **Impact Parameter** The sky-projected distance between the center of the stellar disc and the center of the planet disc at conjunction, normalized by the stellar radius.

- **Transit Duration (hours)** The duration of the observed transits. Duration is measured from first contact between the planet and star until last contact. Contact times are typically computed from a best-fit model produced by a Mandel-Agol (2002) model fit to a multi-quarter Kepler light curve, assuming a linear orbital ephemeris.

- **Ingress Duration (hours)** The time between first and second contact of the planetary transit. Contact times are typically computed from a best-fit model produced by a Mandel-Agol (2002) model fit to a multi-quarter Kepler light curve, assuming a linear orbital ephemeris.

- **Transit Depth (parts per million)** The fraction of stellar flux lost at the minimum of the planetary transit. Transit depths are typically computed from a best-fit model produced by a Mandel-Agol (2002) model fit to a multi-quarter Kepler light curve, assuming a linear orbital ephemeris.

- **Planet-Star Radius Ratio** The planet radius divided by the stellar radius.

- **Fitted Stellar Density [g/cm*3]** Fitted stellar density is a direct observable from the light curve that, in the small-planet approximation, depends only on the transit's period, depth, and duration (see Seager and Mallen-Ornelas 2003). This quantity is directly fitted in the LS and MCMC methods, and is completely independent from the listed stellar mass and radius, which are derived using ground-based photometry, spectroscopy, and other observations

- **Planetary Fit Type** Type of Fit for planetary parameters. Options are: LS (Least Squares fit) MCMC (Markov Chain Monte Carlo fit) DV (Data Validation pipeline fit) none (fit is not provided, only orbital period, transit epoch and transit duration are reported) LS+MCMC (Least Squares Fit with Markov Monte Carlo error bars)

- **Planetary Radius (Earth radii)** The radius of the planet. Planetary radius is the product of the planet star radius ratio and the stellar radius.

- **Inclination (deg)** The angle between the plane of the sky (perpendicular to the line of sight) and the orbital plane of the planet candidate.

- **Equilibrium Temperature (Kelvin)** Approximation for the temperature of the planet. The calculation of equilibrium temperature assumes a) thermodynamic equilibrium between the incident stellar flux and the radiated heat from the planet, b) a Bond albedo (the fraction of total power incident upon the planet scattered back into space) of 0.3, c) the planet and star are blackbodies, and d) the heat is evenly distributed between the day and night sides of the planet.

Referências

AKESON, R. L. et al. The nasa exoplanet archive: Data and tools for exoplanet research. *Publications of the Astronomical Society of the Pacific*, [The University of Chicago Press, Astronomical Society of the Pacific], v. 125, n. 930, p. 989–999, 2013. ISSN 00046280, 15383873. Disponível em: <http://www.jstor.org/stable/10.1086/672273>. 17, 18

ALZUBI, J.; NAYYAR, A.; KUMAR, A. Machine learning from theory to algorithms: an overview. In: IOP PUBLISHING. *Journal of physics: conference series*. [S.l.], 2018. v. 1142, n. 1, p. 012012. 15

AMIT, Y.; GEMAN, D. Shape quantization and recognition with randomized trees. *Neural computation*, MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , v. 9, n. 7, p. 1545–1588, 1997.

BEKLEMYSHEVA, A. *Why Use Python for AI and Machine Learning?* 2022. Disponível em: <https://steelkiwi.com/blog/python-for-ai-and-machine-learning/#:~:text=Benefits%20that%20make%20Python%20the,overall%20popularity%20of%20the%20language./>. 46

BHAMARE, A. R.; BARAL, A.; AGARWAL, S. Analysis of kepler objects of interest using machine learning for exoplanet identification. In: IEEE. *2021 International Conference on Intelligent Technologies (CONIT)*. [S.l.], 2021. p. 1–8.

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

BROWNLEE, J. *How to Develop a Random Subspace Ensemble With Python*. <https://machinelearningmastery.com/random-subspace-ensemble-with-python/>. (Accessed on 11/09/2022). 35

BROWNLEE, J. *Ensemble learning algorithms with Python: Make better predictions with bagging, boosting, and stacking*. [S.l.]: Machine Learning Mastery, 2021. 32, 33, 34, 35, 36, 38

CALTECH. *kepler objects of interest (koi) activity tables*. 2023. Disponível em: <https://exoplanetarchive.ipac.caltech.edu/docs/PurposeOfKOITable.html>. Acesso em: 13 out. 2023. 18

CALTECH. *NASA Exoplanet Archive*. 2023. Disponível em: <https://exoplanetarchive.ipac.caltech.edu/>. Acesso em: 13 out. 2023. 17

CHANAMARN, N.; TAMEE, K.; SITTIDECH, P. Stacking technique for academic achievement prediction. *Int. Work. Smart Info-Media Syst. Asia (SISA 2016), no. Sisa*, v. 2016, p. 14–17, 2016.

DERHAB, A. et al. Blockchain and random subspace learning-based ids for sdn-enabled industrial iot security. *Sensors*, MDPI, v. 19, n. 14, p. 3119, 2019.

DONG, X. et al. A survey on ensemble learning. *Front. Comput. Sci.*, Springer Science and Business Media LLC, v. 14, n. 2, p. 241–258, abr. 2020. 20, 23, 36

DU, S.; LIU, C.; XI, L. A selective multiclass support vector machine ensemble classifier for engineering surface classification using high definition metrology. *Journal of Manufacturing Science and Engineering*, American Society of Mechanical Engineers Digital Collection, v. 137, n. 1, 2015. 36

FLUKE, C. J.; JACOBS, C. Surveying the reach and maturity of machine learning and artificial intelligence in astronomy. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Wiley Online Library, v. 10, n. 2, p. e1349, 2020.

GEURTS, P.; ERNST, D.; WEHENKEL, L. Extremely randomized trees. *Machine learning*, Springer, v. 63, n. 1, p. 3–42, 2006. 37, 38

GEURTS, P.; LOUPPE, G. Learning to rank with extremely randomized trees. In: PMLR. *Proceedings of the Learning to Rank Challenge*. [S.l.], 2011. p. 49–61. 37

HO, T. K. Random decision forests. In: IEEE. *Proceedings of 3rd international conference on document analysis and recognition*. [S.l.], 1995. v. 1, p. 278–282.

KAGGLE. *Kepler Exoplanet Search Results*. 2023. Disponível em: <https://www.kaggle.com>. Acesso em: 13 out. 2023. 47

KAGGLE.COM. *Notebooks documentation*. 2023. Disponível em: <https://www.kaggle.com/docs/notebooks>. Acesso em: 13 out. 2023. 46

KUMAR, A.; MAYANK, J. Ensemble learning for ai developers. *Berkeley, CA: BApress*, Springer, 2020. 32

KURAMA, V. *A Guide To Understanding AdaBoost*. 2020. Disponível em: <https://blog.paperspace.com/adaboost-optimizer/>. 24, 25, 27

MAHARAJ, S. *Model Evaluation Techniques/Machine Learning Model Evaluation*. 2021. <https://www.analyticsvidhya.com/blog/2021/05/machine-learning-model-evaluation/>. (Accessed on 11/21/2022). 50

MISHINA, Y. et al. Boosted random forest. *IEICE TRANSACTIONS on Information and Systems*, The Institute of Electronics, Information and Communication Engineers, v. 98, n. 9, p. 1630–1636, 2015. 29, 31

MITCHELL, T. M. *Machine learning*. [S.l.]: McGraw-hill New York, 1997. v. 1. 12, 15

NASA. *Data columns in Kepler Objects of Interest Table*. 2023. Disponível em: <https://exoplanetarchive.ipac.caltech.edu/docs/API_kepcandidate_columns.html>. Acesso em: 13 out. 2023. 77

NASA. *Kepler Exoplanet Search Result*. 2023. Disponível em: <https://www.kaggle.com/datasets/nasa/kepler-exoplanet-search-results>. Acesso em: 13 out. 2023. 47

NIGRI, E.; ARANDJELOVIC, O. Light curve analysis from kepler spacecraft collected data. In: *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*. [S.l.: s.n.], 2017. p. 93–98.

OFMAN, L. et al. Automated identification of transiting exoplanet candidates in nasa transiting exoplanets survey satellite (tess) data with machine learning methods. *New Astronomy*, Elsevier, v. 91, p. 101693, 2022. 12

PRIYADARSHINI, I.; PURI, V. A convolutional neural network (cnn) based ensemble model for exoplanet detection. *Earth Science Informatics*, Springer, v. 14, n. 2, p. 735–747, 2021. 12

PRZYBYLAR, M. *Why Does Everyone Use Kaggle?* 2020. Disponível em: <https://https://towardsdatascience.com/why-does-everyone-use-kaggle-db1bdf1f1b1a/>. 45

RASCHKA, S. Model evaluation, model selection, and algorithm selection in machine learning. *arXiv preprint arXiv:1811.12808*, 2018. 40, 41, 49

REID, I.; METCHEV, S. *Exoplanets: Detection, Formation, Properties, Habitability ed JW Mason*. [S.l.]: Berlin: Springer, 2008. 12, 17

SAGI, O.; ROKACH, L. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Wiley Online Library, v. 8, n. 4, p. e1249, 2018. 13, 19, 20, 21, 23, 28, 29, 37

SCHANCHE, N. et al. Machine-learning approaches to exoplanet transit detection and candidate validation in wide-field ground-based surveys. *Monthly Notices of the Royal Astronomical Society*, Oxford University Press, v. 483, n. 4, p. 5534–5547, 2019.

SCIKIT-LEARN. *Scikit-learn Machine Learning in Python*. 2023. Disponível em: <https://scikit-learn.org/stable/index.html>. Acesso em: 13 out. 2023. 27

SCIKITLEARN. *Sklearn.Ensemble.AdaBoostClassifier*. 2023. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html#sklearn.ensemble.AdaBoostClassifier/>. Acesso em: 13 out. 2023. 27

SCIKITLEARN. *Sklearn.Ensemble.BaggingClassifier*. 2023. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html>. Acesso em: 13 out. 2023. 37

SCIKITLEARN. *Sklearn.Ensemble.ExtraTreesClassifier*. 2023. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html>. Acesso em: 13 out. 2023. 40

SCIKITLEARN. *Sklearn.Ensemble.RandomForestClassifier*. 2023. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html/>. Acesso em: 13 out. 2023. 32

SCIKITLEARN. *Sklearn.Ensemble.StackingClassifier*. 2023. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingClassifier.html>. Acesso em: 13 out. 2023. 35

SMOLYAKOV, V. *Ensemble Learning to Improve Machine Learning Results - KDnuggets*. 2017. <https://www.kdnuggets.com/2017/09/ensemble-learning-improve-machine-learning-results.html/2>. (Accessed on 11/01/2022).

SOOFI, A. A.; AWAN, A. Classification techniques in machine learning: applications and issues. *Journal of Basic & Applied Sciences*, v. 13, p. 459–465, 2017. 12, 16

THARWAT, A. Classification assessment methods. *Applied computing and informatics*, Emerald Publishing Limited, v. 17, n. 1, p. 168–192, 2020. 49

YING, X. An overview of overfitting and its solutions. In: IOP PUBLISHING. *Journal of physics: Conference series*. [S.l.], 2019. v. 1168, n. 2, p. 022022. 44

YIU, T. *Understanding random forest*. Towards Data Science, 2021. Disponível em: <https://towardsdatascience.com/understanding-random-forest-58381e0602d2>. 28, 29, 31

ZHU, Y.; LIU, J.; CHEN, S. Semi-random subspace method for face recognition. *Image and vision computing*, Elsevier, v. 27, n. 9, p. 1358–1370, 2009. 35