UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO DE CIÊNCIAS, TECNOLOGIAS E SAÚDE DO CAMPUS ARARANGUÁ
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

Milena Seibert Fernandes

Inteligência Artificial Explicável aplicada a Aprendizado de Máquina: Um estudo para Identificar Estresse Ocupacional em Profissionais da Saúde

Araranguá
2022

Milena Seibert Fernandes

Inteligência Artificial Explicável aplicada a Aprendizado de Máquina: Um estudo para Identificar Estresse Ocupacional em Profissionais da Saúde

Trabalho de Conclusão de Curso do Curso de Graduação em Engenharia de Computação submetido ao Centro de Ciências, Tecnologias e Saúde do Campus Araranguá da Universidade Federal de Santa Catarina para a obtenção do título de Bacharel em Engenharia de Computação.
Orientadora: Profa. Analúcia Schiaffino Morales, Dra.

Araranguá
2022

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Seibert Fernandes, Milena
Inteligência Artificial Explicável aplicada a Aprendizado de Máquina: Um estudo para Identificar Estresse Ocupacional em Profissionais da Saúde / Milena Seibert Fernandes ; orientador, Analúcia Schiaffino Morales, 2022.
71 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Campus Araranguá, Graduação em Engenharia de Computação, Araranguá, 2022.

Inclui referências.

1. Engenharia de Computação. 2. inteligência artificial explicável. 3. estresse. 4. biomarcadores. I. Schiaffino Morales, Analúcia. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Computação. III. Título.

Milena Seibert Fernandes

Inteligência Artificial Explicável aplicada a Aprendizado de Máquina: Um estudo para Identificar Estresse Ocupacional em Profissionais da Saúde

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de Bacharel em Engenharia de Computação e aprovado em sua forma final pelo Curso de Graduação em Engenharia de Computação.

Araranguá, 16 de Dezembro de 2022.

Documento assinado digitalmente
Analucia Schiaffino Morales
Data: 20/12/2022 15:05:25-0300
CPF: ***.256.420-**
Verifique as assinaturas em https://v.ufsc.br

Profa. Analúcia Schiaffino Morales, Dra.
Coordenadora do Curso

Banca Examinadora:

Documento assinado digitalmente
Analucia Schiaffino Morales
Data: 20/12/2022 15:05:47-0300
CPF: ***.256.420-**
Verifique as assinaturas em https://v.ufsc.br

Profa. Analúcia Schiaffino Morales, Dra.
Orientadora

Documento assinado digitalmente
SILVIO CESAR CAZELLA
Data: 20/12/2022 16:24:07-0300
CPF: ***.172.680-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Silvio César Cazella, Dr.
Avaliador
Universidade Federal de Ciências da Saúde de Porto Alegre

Documento assinado digitalmente
Alison Roberto Panisson
Data: 20/12/2022 15:00:58-0300
CPF: ***.887.420-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Alison Roberto Panison, Dr.
Avaliador
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Jim Lau
Data: 20/12/2022 19:02:57-0300
CPF: ***.464.702-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Jim Lau, Dr.
Avaliador Suplente
Universidade Federal de Santa Catarina

Inteligência Artificial Explicável aplicada a Aprendizado de Máquina: Um estudo para Identificar Estresse Ocupacional em Profissionais da Saúde

Milena Seibert Fernandes* Analúcia Schiaffino Morales†

2022, DEZEMBRO

Resumo

Estudos científicos ao longos dos anos têm demonstrado que o estresse ocupacional afeta os trabalhadores, principalmente na área da saúde, significando um desequilíbrio entre as condições de trabalho, a capacidade de resposta do trabalhador para realizar suas tarefas e o apoio social recebido de colaboradores e chefias. O objetivo do presente trabalho foi avaliar diferentes respostas de algoritmos de aprendizado de máquina, treinados com dados de biomarcadores provenientes de uma base de dados, chamada *AffectiveRoad*, para identificar diferentes níveis de estresse. Os dados de biomarcadores referem-se a frequência cardíaca, atividade eletrodérmica e temperatura da pele. Foram considerados no estudo os seguintes algoritmos: *Support Vector Machine*, *k-Nearest Neighbors*, *Neural Network*, *Random Forest* e *Logistic Regression*. Para o desenvolvimento da pesquisa foi realizado um levantamento bibliográfico para identificar os principais trabalhos relacionados, entender o contexto da temática, selecionar a base de dados, investigar os algoritmos de aprendizado de máquina e os principais tipos de explicadores. Foram empregados recursos de programação em Python para avaliar o uso dos algoritmos utilizando métricas como AUC e curva de ROC, acurácia, *Recall* e *F1-Score*. Para o processo de seleção e explicação da melhor alternativa, foram utilizadas uma série de métricas relacionadas. E finalmente, para a otimização do treinamento foi utilizado o recurso de *Feature Importance*. Os resultados indicaram que o *Random Forest* apresentou melhor desempenho para o conjunto de teste analisados. Também indicaram que a redução de características dentro do contexto investigado não alterou significativamente os resultados obtidos. Apontando uma possibilidade de ser feita uma redução da quantidade de biomarcadores para o estresse ocupacional de acordo com os dados analisados, sugerindo uma otimização do uso de biomarcadores para a identificação de estresse. A pesquisa apresenta como principal contribuição resultados de testes de algoritmos com explicadores globais e locais que auxiliam na compreensão da identificação do estresse.

Palavras-chaves: inteligência artificial explicável, estresse, biomarcadores.

*misefe1964@gmail.com
†analucia.morales@ufsc.br

Explainable Aritifical Intelligence applied to Machine Learning: A Study on Identifying Occupational Stress in Healthcare Professionals

Milena Seibert Fernandes* Analúcia Schiaffino Morales†

2022, DECEMBER

Abstract

Scientific studies over the years have shown that occupational stress affects workers, especially in the medical field, indicating an imbalance between working conditions, the worker's responsiveness to perform their tasks and the social support received from employees and superiors. The present work aims to evaluate machine learning models that were trained with biomarkers from a database, called *AffectiveRoad*, in order to identify different levels of stress. The available biomarker data refers to heart rate, electrodermal activity and skin temperature. The following algorithms were considered in this study: *Support Vector Machine*, *k-Nearest Neighbors*, *Neural Network*, *Random Forest* and *Logistic Regression*. For the development of the research, a review of the scientific literature was carried out to identify the related works, understand the context of the subject, select the database, investigate the machine learning algorithms and the most important explanator types. Python programming resources were used to evaluate the use of the algorithms through metrics such as AUC, ROC curve, accuracy, Recall and F1-Score. For the process of selecting and explaining the best model, a series of related metrics was used. And finally, to optimize the training of the models, the *Feature Importance* resource was used. The results indicated *Random Forest* as the model that presented the best performance among those analyzed for this test dataset. They also indicated that the reduction of features observed in this context by each model did not significantly change the results obtained, suggesting an optimization in the use of biomarkers analyzed for occupational stress detection. The research presents as its main contribution results of algorithm tests with global and local explanators that help in understanding the identification of stress.

Key-words: explicable artificial intelligence, distress, biomarkers.

*misefe1964@gmail.com
†analucia.morales@ufsc.br

1 Introdução

A pandemia de COVID-19 trouxe inúmeros desafios para profissionais da saúde. Em seu auge, o dia a dia nos hospitais foi inteiramente afetado, apresentando situações com decisões difíceis e práticas exaustivas. Essa realidade alterou consideravelmente as rotinas desses profissionais e ocasionou o aumento de problemas de saúde relacionados as altas cargas de atividades nos ambientes hospitalares de profissionais de diversos setores (PABLO et al., 2020).

Nesse contexto, Osório et al. (2021) apontam índices que registram o sofrimento mental de profissionais de saúde que atuaram na linha de frente da pandemia de COVID-19. Metade dos trabalhadores participantes da pesquisa relataram sofrer sobrecarga de trabalho devido à pandemia, independentemente do nível de complexidade da unidade de saúde, ou em centros de referência para COVID-19. Cerca de 30% apontaram algum problema relacionado a transtornos mentais devido a atividade profissional, a maioria deles relacionados à insônia que é considerada um alerta relevante, pois a privação de sono que atinge os profissionais de saúde é um indicador que pode comprometer a segurança dos pacientes. Os resultados do estudo sugerem um alto nível de sofrimento psíquico em todos os grupos de profissionais avaliados, e indicam também, que os profissionais de saúde estão lidando com altas cargas de trabalho. Isso pode ter tido impacto negativo nesses profissionais, e pode ter refletido no atendimento dos pacientes, comprometendo os resultados clínicos devido ao estresse da atividade ocupacional.

Em um estudo feito sobre parâmetros fisiológicos que podem auxiliar na identificação de estresse, Morales et al. (2022b) destacam os dois tipos de estresse que podem ser considerados. O primeiro tipo, chamado de eustresse, é associado com euforia, e considerado positivo, presente em profissionais de saúde nos casos de promoções ou sucesso no tratamento de pacientes. O segundo, chamado de distresse, é o estresse negativo e vem acompanhado de uma série de sinais apontando exaustão devido os agentes estressores do ambiente de trabalho e pode ocasionar problemas maiores, como é o caso da Síndrome de *Burnout*.

Outras evidências foram apresentadas por Benfante et al. (2020) sobre a dificuldade enfrentada por profissionais de saúde, lidando com uma patologia mortal, sem cura ou vacina, carregando a obrigação de tomar decisões difíceis que podiam resultar em vida ou morte de pacientes. O estudo feito buscou entender o impacto da pandemia de COVID-19 por meio de uma revisão da literatura, apresentando impactos de estresse agudo e pós-traumático. Dentre os sintomas de estresse em geral, foram destacados dificuldade para dormir, mau humor, irritabilidade e falta de atenção, falta de simpatia, perda de apetite, fatiga, medo e conflitos interpessoais.

Estes estudos indicam que é de extrema importância monitorar situações de estresse em profissionais da saúde. Uma vez que os sintomas e o agravamento dessa condição, podem prejudicar o desempenho desses profissionais em situações de vida ou morte, colocando em risco a segurança dos pacientes. A remediação do estresse ocupacional ajuda na diminuição do impacto dos seus sintomas, melhora a saúde mental dos profissionais de saúde e, por consequência, sua performance em situações críticas e/ou de alta pressão. Ou seja, a diminuição do índice de estresse ocupacional pode auxiliar na redução do impacto negativo de pandemias e outras situação agravadas por infecções contagiosas, melhorando a qualidade dos serviços prestados por esses profissionais (BENFANTE et al., 2020).

As técnicas de inteligência artificial têm avançado para auxiliar em processos de diagnóstico e apoio nas tomadas de decisão. No contexto de aprendizado de máquinas, exis-

tem técnicas denominadas de modelos caixa preta (*Black Box*, em inglês). São algoritmos que não possuem transparência e explicabilidade, podendo gerar informações equivocadas ou tendenciosas para determinados resultados. Nos últimos anos surgiram soluções que tentam lidar com áreas que necessitam maior cuidado e atenção com os processos de tomada de decisão, como é o caso da saúde, justiça e educação, denominada inteligência artificial explicável (*Explicable Artificial Intelligence* - XAI)(GUNNING, 2017; ADADI; BERRADA, 2018). O uso de modelos ou algoritmos explicáveis de inteligência artificial apresentam transparência, justificando as decisões tomadas, gerando maior confiabilidade nas saídas encontradas por sistemas inteligentes (DAVE et al., 2020; TJOA; GUAN, 2019; HOLZINGER et al., 2017; GHASSEMI; OAKDEN-RAYNER; BEAM, 2021; ADADI; BERRADA, 2018).

Considerando o contexto de estresse ocupacional, principalmente em situações pandêmicas e a necessidade de prevenir e/ou remediar o estresse, o objetivo deste estudo foi avaliar um conjunto de algoritmos de aprendizado de máquina treinados com dados estatísticos de biomarcadores disponíveis pela base de dados *AffectiveRoad* (LOPEZ-MARTINEZ; EL-HAOUIJ; PICARD, 2019; VOS et al., 2022) para detecção de diferentes níveis de estresse. Os biomarcadores utilizados foram frequência cardíaca (*Heart Rate* - HR), atividade eletrodérmica (*Electrodermal Activity* - EDA) e temperatura da pele (*skin temperature* - TEMP), dados extraídos de Hosseini et al. (2022). Os modelos avaliados foram *Support Vector Machine* (SVM), *k-Nearest Neighbors* (kNN), *Neural Network* (NN), *Random Forest* (RF) e *Logistic Regression* (LR). Após a determinação do algoritmo com melhor desempenho, foram apresentadas explicações, evidenciando quais características deste conjunto de dados influenciam nos resultados da identificação do estresse. Foram empregados para a identificação de explicadores: *Partial Dependency Plot*, *Feature Importance* e *Summary Plot*. Por último, foi feita uma otimização dos modelos considerando apenas as características mais importantes identificadas nas etapas anteriores. E finalmente, foram testados e avaliados novamente, os algoritmos e explicadores para o conjunto de dados otimizado.

A relevância da pesquisa consiste em combinar explicabilidade e aprendizado de máquina para auxiliar na identificação de diferentes níveis de estresse. Trabalhos similares foram identificados no levantamento bibliográfico (CHALABIANLOO et al., 2022; TSENG et al., 2020), no entanto, não foram encontradas evidências de trabalhos que avaliem e expliquem os métodos de aprendizado de máquinas especificamente para o domínio do estresse ocupacional. Desta forma, o preente estudo contribui para o avanço do conhecimento, tanto da área da engenharia da computação como na área da saúde. Isto representa um avanço para o desenvolvimento de ferramentas que possam ser aplicadas nas organizações de saúde, bem como para a realização de futuras pesquisas envolvendo profissionais de saúde, ou de qualquer outra área que exerçam atividades expostas a agentes estressores.

O presente artigo está organizado em sete seções. A fundamentação teórica na seção 2 abrange conceitos de inteligência artificial explicável e biomarcadores para a identificação de estresse. A seção 3 apresenta o levantamento bibliográfico evidenciando trabalhos relacionados, incluindo os mais recentes e significativos avanços da inteligência artificial explicável direcionados à área da saúde. A metodologia explicando as ferramentas e métodos utilizados no trabalho são descritos na seção 4. Seguem os detalhes de desenvolvimento na seção 5, explicando a implementação do trabalho. A seção 6 apresenta os resultados e discussões. Finalmente, seguem as considerações finais e proposições futuras, bem como, as referências bibliográficas. No final do trabalho foram incluídos três apêndices: Apêndice

A com os principais acrônimos utilizados, Apêndice B que contém uma explicação de cada coluna da base de dados utilizada, e o Apêndice C que apresenta o programa na linguagem Python utilizado para o treinamento dos modelos e obtenção dos explicadores.

2 Fundamentação Teórica

Com o passar dos anos, a inteligência artificial vem sendo utilizada em aplicações complexas e que requerem escalabilidade e automação (INAM et al., 2021). No âmbito hospitalar, além de computação em nuvem e Internet das Coisas, a inteligência artificial permite que sistemas de saúde sejam mais eficientes, convenientes e personalizados (PAWAR et al., 2020). No entanto, ainda existem diversos desafios no uso de técnicas de inteligência artificial, para atender estas necessidades e sobretudo, para contribuir nos processos de tomadas de decisão médica (MORALES; OURIQUE; CAZELLA, 2021).

2.1 Por que Inteligência Artificial Explicável?

Segundo Inam et al. (2021), a complexidade e sofisticação de sistemas que envolvem inteligência artificial cresceram até o ponto em que humanos nem sempre são capazes de entender a razão por trás das decisões dos mecanismos que utilizam aprendizado de máquina, um subdomínio da inteligência artificial. Pode-se atribuir esse fato aos grandes conjuntos de dados, compostos por um enorme volume de informações utilizadas para treinar e testar sistemas cada vez mais complexos (LINARDATOS; PAPASTEFANOPOULOS; KOTSIANTIS, 2021).

Atualmente, a possibilidade de interpretar e entender os mecanismos por trás da inteligência artificial é essencial para a validação de modelos de aprendizado de máquina (MONTAVON; SAMEK; MÜLLER, 2018). Sistemas que aplicam inteligência artificial para tomada de decisões podem ser classificados em uma das três categorias apresentadas a seguir.

- **Sistemas opacos** não oferecem nenhuma visão sobre seus mecanismos algorítmicos, escondendo do usuário o seu conhecimento interno (DORAN; SCHULZ; BESOLD, 2017; GUIDOTTI et al., 2018).
- **Sistemas interpretáveis** cujos mecanismos algorítmicos podem ser analisados por seus usuários (DORAN; SCHULZ; BESOLD, 2017).
- **Sistemas compreensíveis** que emitem símbolos que permitem explicações orientadas pelo usuário sobre como uma conclusão é alcançada (DORAN; SCHULZ; BESOLD, 2017).

O aumento de complexidade de sistemas com aprendizado de máquina impacta não somente em aspectos éticos, mas também na sua responsabilidade e segurança (GUIDOTTI et al., 2018). Devido à aplicação de componentes de aprendizado de máquina em indústrias com pouca segurança, como assistentes robóticos e carros autônomos, é importante ter certeza de que os modelos de aprendizado de máquina fazem uso adequado das informações coletadas, e não fazem uso de dados indevidos (MONTAVON; SAMEK; MÜLLER, 2018).

A inteligência artificial explicável se refere a métodos e técnicas que produzem modelos explicáveis e precisos, evidenciando o porquê de um modelo chegar a uma decisão específica. Desse modo, soluções obtidas por sistemas artificialmente inteligentes podem

ser entendidas por humanos. Neste caso com mais transparência e interpretabilidade para promover a confiança nos resultados produzidos por soluções baseadas em inteligência artificial (INAM et al., 2021), visto que pode ser desconfortável confiar em uma decisão tomada sem explicação alguma (DORAN; SCHULZ; BESOLD, 2017; MILLER, 2019).

O fato de que nem sempre humanos são capazes de entender os resultados de algoritmos caixa preta, aumenta a necessidade de interpretabilidade, transparência e explicabilidade de saídas produzidas por sistemas de inteligência artificial. Esses fatores são importantes para que os seres humanos possam entender e confiar em sistemas baseados em inteligência artificial (INAM et al., 2021). Em modelos sofisticados de aprendizado de máquinas, treinados com grandes conjuntos de dados, existe um risco da criação de sistemas de decisão que não são entendidos por seres humanos (GUIDOTTI et al., 2018). Além disso, quando as decisões tomadas por sistemas de inteligência artificial vêm sem explicações, existe um risco grande de que o resultado não será considerado confiável, pois podem haver vieses relacionados à saída (INAM et al., 2021). Além disso, Guidotti et al. (2018) destacam a possibilidade de componentes de aprendizado de máquina tomarem decisões equivocadas sem proporcionar a possibilidade de detecção do problema de aprendizado.

Com explicações sendo providenciadas pelo próprio modelo, é possível determinar sua confiabilidade, sua justiça e sua ética, por exemplo (DORAN; SCHULZ; BESOLD, 2017). Dentre as vantagens da aplicação de métodos de inteligência artificial explicável, destacam-se:

- **Acionabilidade**: Indicações sobre como um resultado pode ser alterado permite que usuários adaptem os parâmetros corretos para se obter o resultado desejado (INAM et al., 2021);
- **Confiabilidade**: Obter confiança de humanos explicando as características e base lógica da saída(INAM et al., 2021);
- **Confiança**: Resultados estáveis e explicáveis sustentam a confiança de seus usuários (INAM et al., 2021);
- **Consciência de privacidade**: Garantia de que os métodos de inteligência artificial não expõem dados confidenciais (INAM et al., 2021);
- **Explicações personalizadas**: Permitem que humanos tenham um melhor entendimento sobre o comportamento do algoritmo fornecendo explicações adaptadas de acordo com a área do conhecimento, especialidade e preferência do usuário (INAM et al., 2021);
- **Informatividade**: O usuário é mais bem informado sobre o funcionamento do modelo gerado pelo algoritmo, de forma a evitar equívocos (INAM et al., 2021);
- **Mais transparência**: Fornecem justificativa para a decisão tomada pelo método de inteligência artificial, aumenta a transparência da operação em questão, aumentando, também, a confiança no resultado (PAWAR et al., 2020);
- **Melhora de modelos**: Com explicações acompanhando os resultados de treinamento, é mais simples detectar bases de dados errôneas e equívocas, fazendo com que erros possam ser identificados ainda durante o treinamento do algoritmo e geração do modelo (PAWAR et al., 2020);

- **Rastreamento de resultados**: As explicações geradas podem facilitar a determinação de qual fator de entrada mais influenciou na decisão tomada pelo resultado do modelo (PAWAR et al., 2020); e
- **Transferibilidade**: A explicação permite um bom entendimento sobre o funcionamento do método, de modo que esse possa ser transferido para outro problema ou aplicação adequadamente (INAM et al., 2021).

2.2 Explicabilidade e Interpretabilidade

Na literatura científica, existem diferentes definições para os conceitos de explicabilidade/explicação e interpretabilidade/interpretação (DORAN; SCHULZ; BESOLD, 2017). Por exemplo, de acordo com Montavon, Samek e Müller (2018), a interpretação é um mapeamento de um conceito abstrato em um domínio que o ser humano é capaz de entender, como imagens e textos. Já Guidotti et al. (2018) expressam que interpretabilidade se trata da habilidade de explicar e providenciar significado, em termos inteligíveis para um ser humano.

Ainda, segundo Montavon, Samek e Müller (2018), uma explicação se trata de uma coleção de fatores do domínio interpretável que contribuíram para que certo exemplo tenha gerado uma determinada solução, como um mapa de calor de pixeis ou textos sublinhados destacando as partes mais influentes para a decisão. Sendo assim, infere-se que explicabilidade caracteriza modelos que podem ser explicados. Por último, Linardatos, Papastefanopoulos e Kotsiantis (2021) associam explicabilidade com lógica interna e mecanismos dentro de um sistema de aprendizado de máquina.

Dessa forma, interpretável é o que pode ser entendido por um ser humano, e explicabilidade é considerada um conjunto de termos interpretáveis que contribuíram para que um exemplo tenha gerado uma determinada solução. Para fins deste artigo, esses conceitos são relevantes pois evidenciam o que o adjetivo "explicável" expressa ao ser associado a um modelo de aprendizado de máquina.

2.3 Dimensões de Interpretabilidade

É importante ter em mente certas dimensões ao se projetar um método de inteligência artificial explicável, de modo que a interpretabilidade do modelo seja adequada à área em que será aplicado e aos usuários por quem será utilizado. As dimensões são citadas a seguir, adaptadas de Guidotti et al. (2018).

- **Interpretabilidade global ou local**: modelos cujas explicações são completamente inteligíveis são globalmente interpretáveis, enquanto modelos localmente interpretáveis que geram explicações para casos semelhantes, não são válidas para todos os casos presentes nos dados de entrada;
- **Limite de tempo**: indica o tempo que o usuário tem para entender as explicações dadas, essa dimensão está relacionada à área de atuação do usuário;
- **Natureza da especialidade do usuário**: usuários podem ter especialidades diferentes, então é ideal determinar os usuários que gostariam de explicações mais sofisticadas e os que gostariam de explicações mais simples.

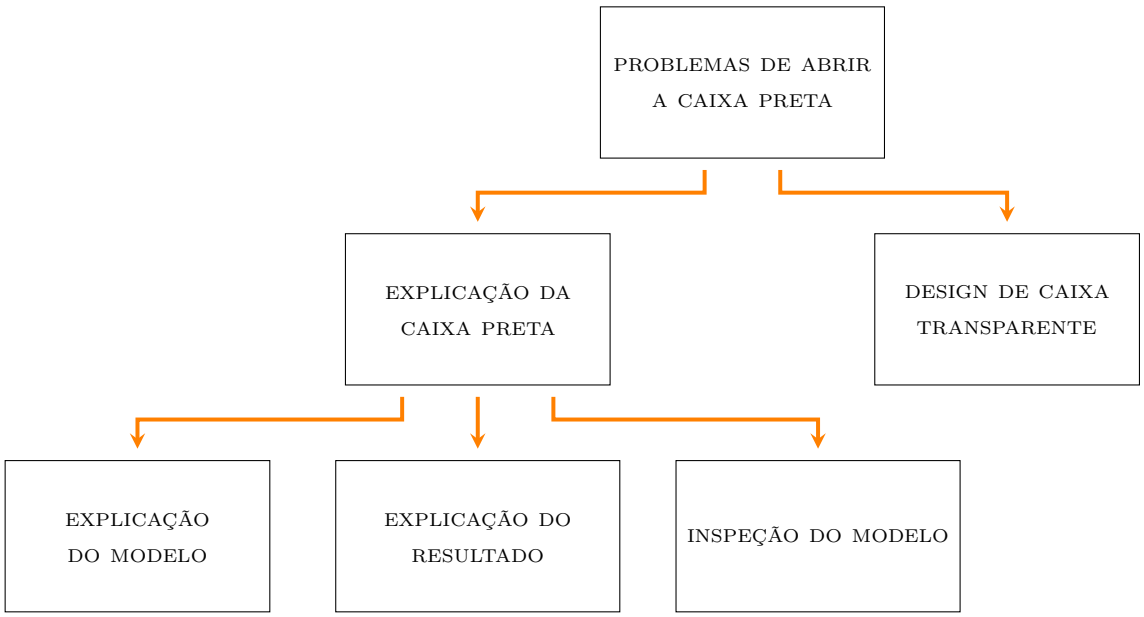
Com isso, o contexto da aplicação deve ser discutido de modo que a dimensão de sua interpretabilidade seja adequada à situação de uso. Nesse sentido se estabelece como a interpretabilidade de um modelo deve ser medida, dada a sua aplicação e seu público-alvo.

2.4 Problemas de Abrir a Caixa Preta

Existem métodos de inteligência artificial opacos, também conhecidos como caixa preta. Como mencionado previamente, esses métodos não oferecem visão sobre o seu funcionamento e lógica interna.

Nesse contexto, Guidotti et al. (2018) também evidenciam **problemas de abrir a caixa preta** cuja taxonomia está descrita na Figura 1. Tal taxonomia representa a separação do problema em duas principais categorias: a **explicação da caixa preta**, responsável por evidenciar como o sistema de decisão retornou certas saídas; e **design de caixa transparente**, que se trata de pensar na solução do problema como transparente, desenvolvendo um classificador que justifica suas decisões por padrão. Ademais, a explicação da caixa preta pode ser subdividida em **explicação do modelo**, quando a explicação envolve toda a lógica do classificador; **explicação do resultado**, quando o objetivo é entender as decisões para uma determinada instância de entrada; e **inspeção do modelo**, quando se deseja entender como o comportamento do classificador muda à medida que os dados de entrada são alterados.

Figura 1 – Problemas de Abrir a Caixa Preta.



Fonte: Adaptado de Guidotti et al. (2018)

Mais especificamente, o conceito apresentado por Guidotti et al. (2018) de **explicação de modelo** consiste na elaboração de um modelo explicador, capaz de obter uma explicação global ou local de um método caixa preta. Tal explicador deve ser capaz de imitar o raciocínio da caixa preta, bem como ser inteligível para seres humanos. No presente trabalho, os explicadores tem papel fundamental para a identificação de características associadas à identificação de níveis de estresse.

2.4.1 Tipos de Algoritmos de Aprendizado de Máquina Caixa Preta

Os algoritmos caixa preta são amplamente utilizados. A desvantagem do seu uso é justamente a falta de interpretabilidade de suas decisões (MORALES et al., 2022b). Destaca-se que estes modelos são adequados para diversas áreas do conhecimento que não necessitam de explicações durante o seu processo de aprendizado. No entanto, como já mencionado, outras áreas exigem o uso de explicações e interpretabilidade, como é o caso da área da saúde. A seguir são descritos algoritmos de caixa preta que são bastante utilizados na literatura científica e tecnológica:

- **Floresta Aleatória (RF)**: Combinação de uma série de árvores de decisão randomizadas, agregando suas pré-direções por meio de médias. Mostra um bom desempenho quando aplicado a problemas em larga escala, com muitas características (BIAU; SCORNET, 2016; SINGH; THAKUR; SHARMA, 2016). Esse algoritmo faz parte de um conjunto de algoritmos intitulado **Conjunto de árvores, ou *Tree ensemble*** (GUIDOTTI et al., 2018).

- **K-Vizinho Mais Próximo (kNN)**: Classifica observações sem rótulos, atribuindo a elas os rótulos de observações similares. O parâmetro k indica quantos vizinhos são selecionados para o algoritmo, podendo reduzir o impacto de valores atípicos, se for um valor alto, ou facilitando o conhecimento de padrões nos dados, se for baixo (ZHANG, 2016; SINGH; THAKUR; SHARMA, 2016).

- **Máquina de vetor de suporte (SVM)**: Utilizando subconjuntos dos dados de treinamento, SVMs buscam por hiperplanos com a maior margem para o limite de decisão. Para isso, *kernels* diferentes são utilizados (GUIDOTTI et al., 2018; SINGH; THAKUR; SHARMA, 2016).

- **Métodos não lineares (NLM)**: Função usada para modelar observações, resultando em uma combinação não linear dos parâmetros do modelo, dependendo de uma ou mais variáveis independentes (GUIDOTTI et al., 2018).

- **Rede neural (NN)**: Formada por um conjunto de neurônios conectados, inspirada em redes neurais biológicas. As conexões entre os neurônios podem transmitir um sinal. Cada conexão geralmente possui um peso atribuído a ela, que se altera durante o processo de treinamento do modelo ajustando a intensidade do sinal transmitido. O neurônio que recebe o sinal pode processá-lo e passá-lo adiante. Tipicamente, os neurônios são organizados em camadas que, pouco a pouco, transformam a entrada em uma saída a partir dos sinais que são transmitidos de uma camada para a outra (GUIDOTTI et al., 2018; SINGH; THAKUR; SHARMA, 2016).

- **Rede neural profunda (DNN)**: Trata-se de uma rede neural capaz de modelar relações complexas e não lineares com o uso de camadas ocultas de neurônios. Existem diversos tipos de DNNs, como Redes Neurais Recorrentes (*Recurrent Neural Networks* (RNNs)), efetivas em aplicações de modelagem de linguagem e Redes Neurais Convolucionais (*Convolutional Neural Networks* (CNNs)), muito utilizadas no processamento de imagens (GUIDOTTI et al., 2018; MONTAVON; SAMEK; MÜLLER, 2018).

- **Regressão Logística (LR)**: Esse modelo se baseia nas chances de um resultado de dois níveis, definindo um dos níveis como resultado, ou evento, de interesse. Com

isso, o modelo considera o algoritmo natural das probabilidades como uma função de regressão dos preditores (LAVALLEY, 2008).

Ainda, Doran, Schulz e Besold (2017) destacam que enquanto interpretação requer transparência dos mecanismos do sistema, um sistema opaco pode ser compreensível desde que emita símbolos que mostram o raciocínio adotado, ou seja, que apresente uma explicação acompanhada de sua saída.

2.4.2 Tipos de Explicadores para Modelos Caixa Preta

Considerando o objetivo do presente trabalho, é importante mencionar os principais tipos de explicadores de modelos opacos (ou "caixa preta") que foram encontrados na literatura científica. São eles:

- **Análise de sensibilidade (SA)**: consiste em avaliar a incerteza da saída de um modelo dadas as imprecisões das entradas (GUIDOTTI et al., 2018; MONTAVON; SAMEK; MÜLLER, 2018);
- **Árvore de decisão (DT)**: um dos modelos mais interpretáveis e fáceis de entender, tanto para explicações globais quanto explicações locais, com representações gráficas das decisões em formato de árvore (GUIDOTTI et al., 2018);
- **Decomposição de Taylor simples (STD)**: explica a decisão do modelo por meio da decomposição do valor do resultado da função em uma soma ou pesos relevantes. Esses pesos são obtidos a partir da identificação dos termos de uma expansão de Taylor de primeira ordem da função em um certo ponto x', para o qual $f(x') = 0$ (MONTAVON; SAMEK; MÜLLER, 2018);
- **Gráfico de dependência parcial (PDP)**: gráficos que ajudam a visualizar e entender a relação entre a saída e parte da entrada de um modelo de caixa preta (GUIDOTTI et al., 2018; NOHARA et al., 2022);
- **Gráfico de sumário (SP)**: Esse tipo de gráfico combina o levantamento de importância de características e gráficos de dependência, fazendo uso de valores SHAP, calculados a partir da contribuição de uma característica em relação às outra características (NOHARA et al., 2022);
- **Importância de características (FI)**: uma solução simples que consiste em retornar a influência e magnitude (peso) das características da entrada utilizadas pelo modelo de caixa preta (GUIDOTTI et al., 2018; MURDOCH et al., 2019; NOHARA et al., 2022);
- **Máscara de saliência (SM)**: maneira de apontar o que causa uma certa saída quando o tipo de dado de entrada é imagem, ou texto, destacando visualmente os aspectos da entrada que mais influenciaram na saída (GUIDOTTI et al., 2018);
- **Maximização de ativação (AM)**: inspeciona redes neurais profundas observando quais neurônios são ativados com quais padrões de entradas. Em resumo, é notado quais neurônios foram ativados em quais camadas, resultando em uma certa saída (GUIDOTTI et al., 2018; MONTAVON; SAMEK; MÜLLER, 2018; HOLZINGER et al., 2017);

- **Regras de decisão (DR)**: muito inteligível para humanos, utilizando vários tipos de regras que servem para explicar a decisão do modelo (GUIDOTTI et al., 2018);
- **Seleção de protótipo (PS)**: um protótipo é um objeto que representa instâncias similares dos pontos observados na entrada. Esse explicador retorna, junto com a saída, um exemplo muito similar de pontos observados, deixando claro quais fatores mais influenciaram na saída obtida (GUIDOTTI et al., 2018).

2.5 Biomarcadores de Estresse

Fica evidente que a presença de estresse resulta em mudanças fisiológicas nos seres humanos, sendo considerada a resposta do corpo humano diante de situações à sensação de estresse ou presença de agentes estressores no ambiente de trabalho Morales et al. (2022b). Os biomarcadores que têm sido identificados para auxiliar na detecção de estresse, são:

- Atividade eletrodérmica (EDA)
- Frequência cardíaca (HR)
- Temperatura da pele (TEMP)
- Pulso de volume sanguíneo (BVP)
- Eletrocardiograma (ECG)
- Eletromiograma (EMG)
- Temperatura corporal (BT)
- Respiração (RESP)
- Aceleração de três eixos (ACC)
- Resposta galvânica da pele (GSR)
- Variação da frequência cardíaca (HRV)
- Eletroencefalograma (EEG)

Outrossim, é notável que biomarcadores não invasivos são vistos como promissores para a identificação de estresse (MORALES et al., 2022b). Como exemplo, pode-se citar a variabilidade da frequência cardíaca, medições do suor e de alterações na pele, e valores relacionados à respiração e temperatura corporal. Ainda, os biomarcadores mostram mudanças físicas e fisiológicas devido ao funcionamento do sistema nervoso simpático, indicando alterações em níveis hormonais como o aumento de cortisol e glicose, bem como outros fatores físicos como contração muscular, respiração, diâmetro da pupila e pressão sanguínea (MORALES et al., 2022).

Destacam-se três biomarcadores recomendados pelos estudos realizados por Morales et al. (2022a): EDA - medida de atividade elétrica da pele relacionado a condutividade; HR - frequência de batimentos cardíacos; e TEMP - temperatura da superfície da pele. Esses parâmetros serão utilizados no presente trabalho para identificação de estresse, e estão disponíveis em uma base de dados acessível publicada em Hosseini et al. (2022).

2.5.1 Coleta de Biomarcadores

Para complementar o estudo sobre aquisição de dados, destacam-se as tecnologias vestíveis. Nos últimos anos tem sido intensificado o interesse em monitoração de estresse através de sensores físicos práticos de utilizar. Chalabianloo et al. (2022) apresentam um levantamento de desempenho de sete modelos de dispositivos vestíveis na medição de estresse. Tais dispositivos são equipados com sensores capazes de medir e transmitir valores de biomarcadores. O levantamento inclui biomarcadores que cada dispositivo é capaz de coletar, juntamente com a taxa de amostragem e o protocolo de comunicação para transmissão de dados coletados. Em outra publicação recente foram destacados os

vestíveis de pulso e o uso para a identificação de estresse ocupacional (MORALES et al., 2022).

A tabela 1 apresenta um resumo das informações apresentadas no trabalho de Chalabianloo et al. (2022).

Tabela 1 – Levantamento de dispositivos vestíveis

Dispositivo	Sensores	Taxa de Amostragem	Transmissão
Empatica E4	HR, EDA, ACC, BVP, TEMP	64 Hz	Bluetooth
Samsung Gear S2	HR, ACC, BVP, Ângulo	100 Hz	Bluetooth
Firstbeat Bodyguard 2	ECG, ACC	1000 Hz	USB
BITalino (r)evolution	ECG, EEG, EDA, EMG, ACC	1000 Hz	Bluetooth
Polar H10	ECG	130 Hz	Bluetooth
Zephyr HxM	ECG	250 Hz	Bluetooth
CorSense	HR	500 Hz	Bluetooth

Fonte: Adaptado de Chalabianloo et al. (2022)

3 Levantamento Bibliográfico

Para investigar melhor o tema, foi realizada uma revisão da literatura científica em forma de biometria. Os termos de busca foram: *"Explainable Artificial Intelligence" AND ((("health professionals" OR "heath workers") AND "stress" AND "monitoring") OR "healthcare")*. Inicialmente, foi estabelecido o período de busca para os últimos cinco anos (de 2018 a 2022), porém, como foram encontrados apenas 13 estudos relevantes para a pesquisa nesse período, foram analisados também artigos do período anterior correspondendo aos anos entre 2013 e 2017. Apenas mais um estudo foi adicionado aos resultados com ano de publicação de 2017.

A tabela 2 mostra o número de artigos obtidos na investigação e a base de referência utilizada. A segunda coluna mostra o número total de resultados em cada plataforma de busca. Enquanto a terceira, mostra o número de trabalhos escolhidos como relevantes após leitura preliminar, observando o título e o resumo, seguindo com exploração e interpretação do material para análise de temas dessemelhantes. Por fim, a quarta coluna mostra o número total de artigos selecionados, descartando-se aqueles que já haviam sido encontrados e selecionados em uma outra plataforma listada.

Tabela 2 – Resultados de Buscas por Artigos Relevantes

Plataforma	Total	Filtrados por leitura	Removendo duplicados
Scielo	0	0	0
Science Direct	212	2	2
Periódico CAPES	125	8	6
BDTD	0	0	0
Science.gov	305	1	1
Google Scholar	112	10	1
World Wide Science	583	14	4
Total	1337	31	**14**

Fonte: Própria Autora.

3.1 Trabalhos relacionados

A partir dos resultados obtidos foram identificadas pesquisas de revisões de literatura, ou *surveys*, que objetivaram avaliar o estado da arte das tecnologias de inteligência artificial explicável na área da saúde. Em seguida, foram analisadas as pesquisas que permitem comparar os métodos explicáveis na área da saúde com aplicações bem específicas. Foram destacadas pesquisas que propuseram modelos com aplicações na área médica empregando recursos e técnicas de explicabilidade; e trabalhos relacionados a identificação de estresse. Os quatorze trabalhos foram organizados em duas categorias, artigos de revisão e artigos de pesquisa na área da saúde.

3.1.1 Artigos de Revisão

Com relação a revisão de literatura ou *survey* foram selecionados cinco artigos. O primeiro identificado foi o trabalho de Holzinger et al. (2017) que se trata de uma visão geral do estado da arte de modelos explicáveis presentes na literatura até o ano de 2017. O trabalho evidencia que os métodos de inteligência artificial com melhor performance são os menos transparentes, enquanto os mais transparentes possuem menos acurácia. Além de trazer conceitos importantes, como explicabilidade e modelos explicáveis, o trabalho traz um instantâneo do estado da arte de inteligência artificial explicável em vários campos da medicina, incluindo reconhecimento e classificação de imagens e interpretação de texto. Como conclusão, a pesquisa indica o grande número de desafios que a inteligência explicável ainda tem a enfrentar, enquanto ainda oferece grandes oportunidades na área da medicina. Além disso, também são evidenciadas as vantagens da explicabilidade em uma área em que decisões tomadas podem resultar em vida ou morte.

Outra pesquisa no estilo *survey* foi feita por Tjoa e Guan (2019). Focando em tipos de interpretabilidade apresentados, o trabalho buscou categorizá-los de acordo com seu grau de interpretabilidade. Como resultado, além de categorias de interpretabilidade, foram levantados desafios e possíveis melhorias para a área de inteligência artificial explicável. Os desafios levantados incluem: manipulação de explicações, como ruídos nos dados que podem facilmente alterar explicações para torná-las equivocadas; restrições incompletas, indicando que as restrições presentes em modelos não explicáveis muitas vezes não são considerados em modelos explicáveis, prejudicando seu desempenho; e, bases de dados com ruídos, visto que na área médica, muitas informações advindas de profissionais da saúde podem nem sempre ser a verdade absoluta. Além disso, a pesquisa descreve direções futuras para profissionais da saúde, desenvolvedores de software e pesquisadores. Essas direções incluem

seguir supervisionando modelos de inteligência artificial explicável já implantados no âmbito hospitalar, levando suas explicações apenas como suporte complementar enquanto não há maneira mais robusta de juntar interpretabilidade e acurácia, e também continuar o avanço da interpretabilidade na área médica, coletando mais dados, desenvolvendo novos métodos e evoluindo os algoritmos existentes.

De forma semelhante, Ghassemi, Oakden-Rayner e Beam (2021) providenciam uma visão geral do estado da arte de modelos de inteligência artificial explicável em um período mais recente, por meio de uma revisão da literatura e análise sobre acurácia e eficiência dos modelos explicáveis existentes. O título do trabalho indica qual é a sua conclusão: apesar da certeza do impacto positivo que a inteligência artificial terá no futuro da medicina, os métodos explicáveis disponíveis ainda não são capazes de fornecer a certeza de que as decisões tomadas por eles são corretas e confiáveis. Desse modo, o estudo indica que é preciso cuidado por parte dos médicos, e certo ceticismo antes de se confiar em decisões tomadas por modelos de aprendizado de máquina explicáveis.

Outro estudo, feito por Morales et al. (2022), apresenta também uma revisão sistemática da literatura, mas foca em publicações científicas que tratam de relógios ou pulseiras que ajudam a identificar níveis de estresse. O trabalho faz uma seleção de estudos relevantes, também apresentando um levantamento de biomarcadores utilizados em cada um dos trabalhos selecionados. Apesar de não apresentar análises sobre métodos de inteligência artificial explicável, e 20% dos trabalhos analisados não fazerem uso de nenhuma técnica de inteligência artificial, o trabalho evidencia informações relevantes sobre os efeitos do estresse ocupacional e como a detecção dele é feito em publicações científicas.

Por fim, Morales et al. (2022b) publicaram um capítulo de livro que aborda os problemas relacionados ao monitoramento de estresse em profissionais da saúde utilizando dispositivos vestíveis. A pesquisa trata de uma revisão da literatura focada nos biomarcadores e técnicas de aprendizado de máquina utilizados nos trabalhos revisados. Como resultado, foi apresentada uma proposta de sistema de recomendação que se baseia no grau de estresse percebido, sendo transparente e explicável. O sistema proposto é capaz de aplicar algoritmos de aprendizado profundo para identificar o estresse, e mostra aos usuários como mitigá-lo para que o quadro não se agrave. O sistema também agrega conhecimentos de médicos e psicólogos, gerando explicações textuais e visuais. Além disso, o sistema também aceita *feedbacks* dos usuários, a fim de melhorar seu desempenho para recomendações futuras. Salienta-se que no capítulo foi apresentada uma proposta de sistema, e não o seu desenvolvimento em si, não constam muitos detalhes referentes aos algoritmos e métodos de interpretabilidade que poderiam ser aplicados. Além disso, a pesquisa destaca preocupações com segurança de dados e privacidade, evidenciando uma lacuna na literatura referente a segurança durante transmissão de dados em redes sem fio e para a nuvem, e propõe a incorporação de *hashing* de dados para melhorar a segurança do sistema.

3.1.2 Artigos de pesquisa na área da saúde

Dentro desta categoria foram agrupados em artigos de comparação de métodos explicáveis, modelos com aplicações específicas e modelos que tratam especificamente de estresse ocupacional.

Madanu et al. (2022) se encaixam parcialmente na categoria de artigos de revisão, pois apresentam uma pesquisa no estilo *survey*, avaliando a importância de inteligência artificial na área médica, destacando que recentemente a inteligência artificial vem sendo

aplicada para diagnósticos de dores associadas a diversas patologias baseados em fatores comportamentais e/ou fisiológicos. O artigo chama atenção para o papel que a inteligência artificial tem potencial para desempenhar na área médica futuramente, avaliando métodos de inteligência artificial explicável e como esses são aplicados para diferentes tipos de dor. Como resultado, o trabalho evidencia que biomarcadores são utilizados para detecção de diferentes tipos de dores, e ainda evidencia que a aplicação de técnicas de inteligência artificial explicável pode apontar os indicadores mais relevantes e aclarar as verdadeiras causas de patologias.

A pesquisa de Pawar et al. (2020) visa discutir qual é a melhor maneira de aplicar inteligência artificial explicável para fazer diagnósticos de pacientes baseado em seus dados médicos. A proposta do trabalho se resume a utilizar dispositivos que coletam os dados médicos relevantes do paciente, como relógios inteligentes, passando esses dados para modelos de inteligência artificial, que predizem a probabilidade de patologias. Após isso, métodos de inteligência artificial explicável são aplicados a essas predições, adicionando transparência aos diagnósticos. Para casos em que os diagnósticos estão corretos, esse sistema pode ser usado para fornecer informações e recomendações de grande valor para os pacientes. Por outro lado, no caso de diagnósticos errôneos, as inconsistências entre as predições e o conhecimento dos médicos podem ser usadas para calibrar os modelos utilizados e melhorar futuras predições. Por fim, o estudo evidencia como a inteligência artificial explicável pode ser utilizada para tornar sistemas inteligentes e autônomos inteligíveis e rastreáveis.

Em adição, Dave et al. (2020) apresentam um estudo relacionado a técnicas de interpretabilidade de sistemas explicáveis, de modo a esclarecer para profissionais da saúde a compreensibilidade de sistemas de inteligência artificiais explicáveis. Ainda, o estudo faz uso de exemplos baseados em bases de dados de doenças cardíacas, aclarando como técnicas de explicabilidade devem ser aplicadas para fornecer mais confiabilidade no uso de sistemas de inteligência artificial na área da saúde. O objetivo foi aplicar técnicas de inteligência artificial baseados em características e exemplos, treinando modelos de aprendizado de máquina utilizando a base de dados de doenças cardíacas do repositório UCI ML e o algoritmo XGBoost (*eX-treme Gradient Boosting*). Os modelos treinados são implantados em ambientes de produção, para que a predição do modelo e sua explicação sejam combinadas em sua saída. Os resultados do artigo mostram como diversas técnicas de inteligência artificial explicável apresentam suas explicações, indicando que fatores são considerados e o seu impacto na saída final. Para isso, são apresentados gráficos de força, de resumo e de dispersão, contrastando explicações globais e locais. Em adição, são comparadas as performances dos explicadores utilizados, avaliando também a facilidade de entendimento das explicações locais e globais dadas pelos explicadores.

A pesquisa de Hossain, Muhammad e Guizani (2020) propõe um sistema de vigilância em massa baseada em inteligência artificial explicável para combater pandemias como a COVID-19. A estrutura proposta utiliza a rede 5G para detectar COVID-19 utilizando raios-X ou ressonâncias. Para tanto, são utilizados os modelos de aprendizado profundo *ResNet50*, *Deep tree* e *Inception v2*. Em cada hospital que é monitorado pelo sistema existe um servidor, que utiliza inteligência artificial explicável para permitir que médicos entendam as saídas de cada um dos modelos. Essa explicação se dá por visualização, com destaque para regiões das imagens coletadas que indicam o diagnóstico positivo ou negativo. Ainda, é gerada uma explicação interativa, em que a interação do médico combina a imagem destacada e uma explicação em texto.

Tseng et al. (2020) desenvolveram modelos utilizando métodos de aprendizado de máquina, como SVM, RF e LR para prever o desenvolvimento de lesões no rim posteriores a cirurgias cardíacas. A análise do desempenho dos modelos foi feita por meio de curvas de ROC dos modelos, indicando *Random Forest* e *XGboost* como os modelos com o melhor desempenho. Para complementar, são apresentadas explicações desses modelos, incluindo importância de características, gráfico de dependência parcial e árvore de decisão. Como resultado, o artigo obtém as características mais influentes e seus impactos, sendo positivos ou negativos, na saída do modelo, utilizando gráficos de dependência para evidenciar tais impactos.

Ainda, o trabalho de Ammar e Shaban-Nejad (2020) aborda experiências adversas da infância, mais especificamente menciona dados coletados durante estudos conduzidos nos últimos 20 anos. Tais dados, de acordo com os autores, podem ser utilizados por modelos de inteligência artificial explicável para fornecer um entendimento mais completo das análises realizadas. É evidenciado também que modelos de aprendizado de máquina tradicional não permitiriam tal interpretação. Com isso, a pesquisa teve o objetivo de aplicar técnicas de inteligência artificial explicável para desenvolver um protótipo de sistema de recomendação que visa melhorar o monitoramento de saúde mental de pacientes. Para o desenvolvimento do sistema, a metodologia lista etapas de design de plataforma, arquitetura de sistemas, e agente de conversação que utiliza intenção, entidades de diversos tipos, parâmetros de contexto e eventos para conversar com pacientes e coletar dados sobre sua saúde mental. Com isso, foram obtidos os principais aspectos de uma plataforma chamada SPACES, que usa os dados coletados pelo agente de conversação para fazer recomendações utilizando um modelo de aprendizado de máquina explicável. Assim, segundo os autores, as recomendações feitas possuem mais credibilidade e transparência.

Bahani, Moujabbir e Ramdani (2021) propõem um sistema que providencia uma base de conhecimento explicável que evidencia o processo de tomada de decisão no diagnóstico de doenças cardíacas. Para o desenvolvimento desse sistema, foram utilizadas as bases de dados de doenças cardíacas Cleveland, Hungarian e Va long beach e técnicas de classificação *fuzzy* por meio de agrupamento. O sistema desenvolvido ainda é comparado com outros modelos de aprendizado de máquina como NN, SVM, *K-Nearest neighbor*, *Random Forest*. O artigo mostra a acurácia, especificidade e sensitividade de cada um dos métodos de inteligência artificial utilizados, evidenciando superioridade do modelo *fuzzy* com agrupamento desenvolvido. A interface de explicação do modelo utiliza um explicador de importância de características.

Considerando o grupo de trabalhos relacionados à identificação de estresse ocupacional tem-se a pesquisa de Hosseini et al. (2022), que conta com a detecção de estresse a partir de dados coletados de enfermeiras durante o desempenho do seu trabalho no período da pandemia. Seu escopo inclui o recrutamento de enfermeiras, cujos dados são coletados com o dispositivo vestível *Empatica E4*. O modelo utilizado para a detecção de estresse foi treinado com a base de dados *AffectiveRoad* (LOPEZ-MARTINEZ; EL-HAOUIJ; PICARD, 2019). Ainda, quando o estresse é detectado pelo modelo, um aplicativo de dispositivo móvel aplica um questionário, que deve ser respondido pelas participantes, sobre o evento, indicando que fatores contribuíram para a manifestação do estresse ocupacional naquele período. A principal contribuição do trabalho se trata dos dados coletados, contendo biomarcadores de frequência cardíaca, atividade eletrodérmica e temperatura da pele, acompanhada das respostas dos questionários aplicados. Ressalta-se que esta pesquisa não utilizou nenhuma técnica de Inteligência Artificial Explicável em sua solução.

O trabalho de Chalabianloo et al. (2022) avalia dispositivos vestíveis visando a coleta de dados para classificar estresse aplicando técnicas de inteligência artificial explicável. Além de evidenciar a contribuição do avanço da tecnologia vestível para o monitoramento do estresse, o estudo compara o desempenho de sete dispositivos vestíveis para a detecção de estresse, usando-os em 32 participantes e aplicando diferentes algoritmos de aprendizado de máquina. A pesquisa apresenta um levantamento dos sensores, taxa de amostragem e protocolo de comunicação para transmissão de dados coletados de cada um dos dispositivos considerados. Como resultado, o estudo aponta, por meio de explicações para os modelos desenvolvidos, que existem diferenças nas influências que cada característica pode ter na saída, dependendo dos modelos utilizados. Mais especificamente, o estudo aponta que modelos que levam em consideração somente o efeito de variação de frequência cardíaca (HRV) não são sempre confiáveis, e é recomendado o uso de mais biomarcadores como atividade eletrodérmica. Complementarmente, o estudo aponta que modelos que levam em consideração dados de eletrocardiografia (ECG), coletados por vestíveis peitorais, bem como dispositivos multisensores como *Empatica E4* coletando dados de frequência cardíaca (HR) e atividade eletrodérmica (EDA) tiveram as melhores performances na detecção de estresse.

4 Metodologia

Visando atingir o objetivo proposto, sobre avaliar algoritmos de aprendizado de máquina treinados para detectar diferentes níveis de estresse, foram adotados os seguintes procedimentos metodológicos. Através da análise de conteúdo (Laurence Bardin, 2011) foi possível desenvolver as seções 2 e 3 deste artigo. Essas etapas permitiram identificar os tipos de modelos de caixa preta, os tipos de explicadores para modelos de caixa preta, bem como os biomarcadores de estresse utilizados em pesquisas anteriores e dar sequência à pesquisa.

Análise de conteúdo é um conjunto de técnicas metodológicas e procedimentos sistemáticos para análises qualitativas, com objetivos de descrição de conteúdo que permitem inferir conhecimento a partir dos textos e ou mensagens analisadas (Laurence Bardin, 2011). Ela é composta por três fases fundamentais:

1. A pré-análise envolve leitura preliminar/flutuante, o primeiro contato com os materiais em análise;
2. A exploração do material envolve a codificação, classificação e categorização dos materiais; e,
3. O tratamento dos resultados envolve a inferência e interpretação (de conteúdos e proposições) dos resultados dos materiais analisados (Laurence Bardin, 2011).

4.1 Critérios adotados e procedimentos de seleção de dados com biomarcadores

Para a escolha da base de dados utilizada para o treinamento dos modelos avaliados, procurou-se artigos que utilizaram bases de dados disponíveis publicamente para a detecção de estresse em ambientes hospitalares, ou aplicações relacionadas à área da saúde. Logo, considerou-se o artigo de Hosseini et al. (2022), que utiliza a base de dados *AffectiveRoad* (LOPEZ-MARTINEZ; EL-HAOUIJ; PICARD, 2019; VOS et al., 2022), e ainda disponibiliza a base de dados já pré-processada com 48 colunas de dados estatísticos

sobre os biomarcadores coletados. Os biomarcadores em questão são frequência cardíaca (HR), atividade eletrodérmica (EDA) e temperatura da pele (TEMP), já destacados anteriormente.

A base de dados *AffectiveRoad* é mencionada por Vos et al. (2022) como sendo feita originalmente para monitorar a atenção de motoristas, mas aplicada também para detecção de estresse a partir de dados coletados por dispositivos vestíveis. Sendo assim, utilizou-se essa base de dados que se encontrava pré-processada e disponibilizada pública e gratuitamente para o treinamento dos modelos avaliados.

4.2 Critérios adotados e procedimentos de seleção de algoritmos de aprendizado de máquina

Para a escolha de algortimos de aprendizado de máquina avaliados neste estudo, utilizou-se os levantamentos feitos por Morales et al. (2022b), que apontam quais tipos de algoritmos são utilizados quando se objetiva detectar estresse ocupacional. Ainda, Guidotti et al. (2018) levam em consideração que tipos de dados são utilizados no treinamento de modelos de aprendizado de máquina. Logo, levando em consideração que os dados presentes na base escolhida são numéricos e organizados em tabela, foram selecionados os modelos *Support Vector Machine* (SVM), *Random Forest* (RF), *Logistic Regression* (LR), *k-Nearest Neighbors* (kNN) e *Neural Network* (NN) para dar sequência à pesquisa.

Vale ressaltar que os algoritmos selecionados enquadram-se em algoritmos de aprendizado supervisionado. O emprego de aprendizado supervisionado se justifica, visto que os dados utilizados, advindos da pesquisa de Lopez-Martinez, El-Haouij e Picard (2019), são rotulados com nível de estresse esperado como saída.

4.3 Critérios adotados e procedimentos de seleção de explicadores

A seleção de explicadores foi baseada no levantamento bibliográfico. Pesquisas como a de Guidotti et al. (2018) e Montavon, Samek e Müller (2018) evidenciam que explicadores podem ser utilizados dependendo do formato de entrada e saída do modelo. Como por exemplo, quando se utilizam imagens, uma boa escolha seria máscara de saliência para destacar que regiões da imagem mais influenciaram na saída. No caso dessa aplicação para detecção de estresse, boas escolhas para evidenciar claramente o motivo das decisões do modelo são explicadores como *Summary Plot* (SP), *Partial Dependency Plot* (PDP) e *Feature Importance* (FI). Em adição, foram utilizados *Shapley Additive Explanators* (ANTWARG et al., 2021) para a obtenção de tais explicadores.

4.4 Avaliação de algoritmos de aprendizado de máquina

Para fins de avaliar o desempenho dos cinco algoritmos de aprendizado de máquina treinados foi feito um programa na linguagem de programação Python empregando as bibliotecas *NumPy*[1], *Pandas*[2] e *sklearn*[3] para treinar modelos classificadores. Com isso, a base de dados foi dividida para treino e teste, de modo que os modelos pudessem ser treinados e avaliados por meio dos parâmetros de desempenho obtidos. O programa foi disponibilizado no Apêndice C.

[1] <https://numpy.org/>
[2] <https://pandas.pydata.org/>
[3] <https://scikit-learn.org/stable/index.html>

4.5 Explicação do algoritmo melhor avaliado

Após a avaliação dos resultados, foi complementado o programa em Python para, utilizando os mesmos dados, gerar explicadores para o algoritmo melhor avaliado. Os explicadores gerados são os considerados apropriados na etapa anterior. Para isso, foram utilizadas as bibliotecas do Python *sklearn* e *shap*[4]. Esse programa se encontra disponível no apêndice C.

4.6 Otimização de treinamento por *feature importance* (FI)

Por último, foram consideradas as 10 características mais importantes dos três modelos com o melhor desempenho durante a etapa de avaliação de modelos. Com isso, todos os cinco algoritmos foram treinados novamente, por três vezes:

- Primeiro com as 10 características mais importantes para o primeiro melhor resultado;
- Segundo, com as 10 características mais importantes para o segundo melhor resultado;
- Terceiro, com as 10 características mais importantes para o terceiro melhor resultado.

O treinamento foi feito por meio da complementação do programa presente no Apêndice C. Após isso, foi feita uma comparação do desempenho dos resultados dos algoritmos, visando verificar se o treinamento com menos características prejudicaria a sua precisão e acurácia.

4.7 Resumo dos procedimentos metodológicos adotados na pesquisa

A seguir, a tabela 3 apresenta um resumo das etapas de pesquisa, contendo os objetivos e as formas de coleta e análise dos dados.

[4] <https://shap.readthedocs.io/en/latest/index.html>

Tabela 3 – Resumo das etapas de pesquisa

Etapa de Pesquisa	**Objetivo**	**Coleta de dados**	**Análise de dados**
Revisão da literatura e levantamento de trabalhos relacionados	Conhecer a temática	Pesquisa bibliográfica	Análise de Conteúdo
Revisão da literatura	Analisar a evolução do conhecimento até o momento atual		
Seleção de base de dados	Escolher a base de dados a ser utilizada para detectar o estresse		
Seleção de algoritmos de aprendizado de máquina	Escolher os algoritmos de aprendizado de máquina para serem treinados e avaliados		
Seleção de explicadores	Determinar a melhor forma de explicar as decisões tomadas pelo melhor resultado de aprendizado de máquina		
Avaliação dos algoritmos selecionados	Testar os algoritmos treinados com a base de dados	Python, bibliotecas de programação	Python, bibliotecas de programação
Explicação do algoritmo melhor avaliado	Determinar quais características mais impactaram no resultado e de que forma esse impacto se deu na saída do modelo		
Otimização do treinamento de algoritmos por *Feature Importance*	Determinar o impacto de diminuir as características da base de dados baseado na importância atribuída a cada coluna		

Fonte: Própria Autora.

5 Desenvolvimento do Trabalho

Esta seção foi subdividida em três etapas: avaliação de algoritmos de aprendizado de máquina, explicação do algoritmo *Random Forest*, que apresentou melhor desempenho durante a avaliação, e o treinamento dos algoritmos com as características mais importantes.

É preciso entender, em primeiro lugar, alguns pontos importantes sobre a base de dados utilizada no treinamento dos modelos. Trata-se de um arquivo csv contendo dados pré-processados, obtido através do repositório[5] associado ao trabalho de Hosseini et al. (2022). O *dataset* possui 49 colunas, listadas e explicadas no apêndice B. Seguem os resultados das avaliações dos algoritmos treinados e demais subseções contendo as etapas de desenvolvimento do trabalho.

[5] <https://github.com/CPHSLab/Stress-Detection-in-Nurses>

5.1 Avaliação de modelos de aprendizado de máquina

Este tópico apresenta uma comparação de desempenho dos algoritmos de aprendizado de máquina selecionados, que são caixa preta.

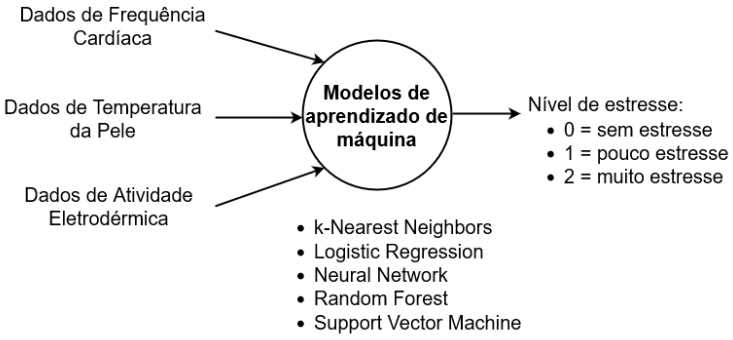
5.1.1 Critérios de avaliação de modelos

Foi elaborado um programa na linguagem Python que obtém estatísticas relacionadas ao desempenho dos modelos, empregando classes da biblioteca *sklearn* para avaliar os algoritmos selecionados no trabalho. A biblioteca *Pandas* permitiu que a base de dados *AffectiveRoad* pré-processada, fornecida por Hosseini et al. (2022), fosse transformada em um *DataFrame*, usado para treinar e testar os algoritmos. Ainda com funções da biblioteca *sklearn*, foi possível obter parâmetros que indicam a performance de cada algoritmo. Esses parâmetros foram:

- **Acurácia**: proporção de exemplos classificados corretamente;
- **AUC**: área abaixo da curva de ROC, que indica o desempenho do algoritmo comparado a um classificador aleatório;
- ***Recall***: proporção de verdadeiros positivos dentre todas as instâncias positivas da base de dados.
- ***Score* F1**: média harmônica ponderada de precisão e *recall*;

A Figura 2 mostra um diagrama indicando as entradas e as possíveis saídas dos algoritmos utilizados.

Figura 2 – Diagrama de entradas e saída dos algoritmos



Fonte: Própria autora

Dentre os dados dos biomarcadores, que totalizam 48 características de entrada, incluem média dos últimos 11 ciclos de medição, valor máximo, valor mínimo e desvio padrão dos três biomarcadores considerados. Também, existem outros dados como amplitude, duração, curtose, desvio padrão e distorção da atividade eletrodérmica, e número de picos e batimentos por segundo da frequência cardíaca. Já a saída dos modelos se resume ao nível de estresse, classificado em três níveis. O apêndice B lista os nomes de todas as características de entrada e saída juntamente com o seu significado.

5.1.2 Critérios de treinamento de modelos

Os parâmetros usados para o treinamento dos algoritmos também foram desenvolvidos empregando os recursos da biblioteca *sklearn*. No caso do *Random Forest*, alguns parâmetros foram modificados para seguir as mesmas configurações escolhidas por Hosseini et al. (2022). Em outros casos, os parâmetros foram variados e escolhidos baseando-se no desempenho dos algoritmos com a configuração de parâmetros. A seguir são listados os parâmetros de treinamento aplicados em cada modelo:

- *k-Nearest Nighbors* foi treinado com k sendo 5, métrica euclidiana e peso uniforme.
- *Logistic Regression* foi treinado com tipo de regularização L2, e C igual a 1.
- *Neural Network* foi treinado com 100 neurônios em camadas ocultas, ativação *ReLu*, e otimização Adam.
- *Random Forest* foi treinado com 100 árvores e número mínimo de ramos 5.
- *Support Vector Machine* foi treinado com custo 1, perda de regressão 0,1, *kernel* linear, e tolerância numérica 0,001.

Destaca-se que os algoritmos SVM e LR são mais comumente utilizados como classificadores binários, em aplicações com apenas duas saídas possíveis. Por esse motivo, foram utilizadas abordagens para que os modelos treinados com esses algoritmos sejam capazes de considerar três classes.

Para o algoritmo SVM, foi utilizada uma abordagem *One Vs. One*, em que 3 modelos classificadores foram treinados, cada um com dados de duas classes distintas. Em adição, o modelo LR utilizou a abordagem *One Vs. Rest*, de forma que três classificadores foram treinados, com cada um considerando uma das três saídas possíveis como positiva, e as duas restantes como negativas.

5.1.3 Resultados da avaliação dos algoritmos

Os algoritmos foram então testados, para todos os casos foram selecionados 70% dos dados para o treinamento, e os 30% restantes foram usados para testes. A divisão dos dados para treino e teste foi feita de maneira randômica pela função *train_test_split* da biblioteca *sklearn*. A avaliação se deu por meio da comparação entre as saídas esperadas e as saídas obtidas por cada modelo.

A tabela 4 mostra os resultados obtidos após avaliação dos algoritmos com os dados de teste.

Tabela 4 – Resultados da avaliação de modelos

	Modelo	**AUC**	**Acurácia**	**F1**	***Recall***
(1)	**RF**	0.994	0.955	0.943	0.933
(2)	**kNN**	0.964	0.878	0.854	0.854
(3)	**NN**	0.900	0.771	0.702	0.691
(4)	**LR**	0.747	0.610	0.453	0.496
(5)	**SVM**	0.735	0.617	0.451	0.500

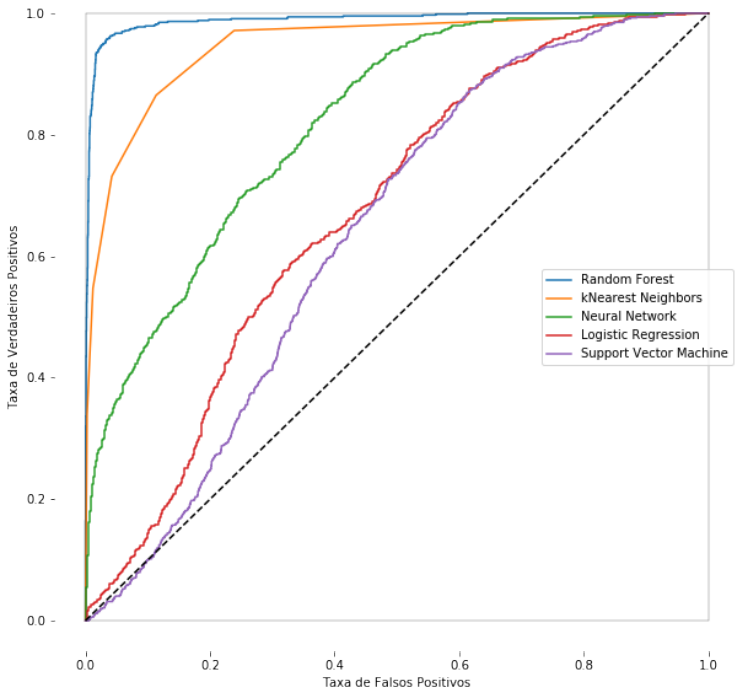
Fonte: Própria Autora.

Para todos os parâmetros em questão, um valor mais próximo de 1 indica melhor desempenho. Por esse motivo, os modelos foram classificados por ordem de AUC, como pode ser observado pela classificação atribuída. Destaca-se que os algoritmos *Logistic Regression* e *Support Vector Machine* trocariam de classificação caso o parâmetro decisivo fosse acurácia ou *recall*.

Além disso, foi traçada a curva de ROC que representa graficamente a relação entre verdadeiros positivos e falsos positivos classificados pelos modelos. A linha pontilhada no meio do gráfico representa o esperado em um classificador randômico, que faz escolhas aleatoriamente. Quanto mais próximo do contorno superior, melhor é o desempenho do modelo, e vice-versa (HOO; CANDLISH; TEARE, 2017). A análise dessa curva é comumente aplicada para avaliar classificadores com saídas binárias, como não é o caso neste trabalho. Por este motivo, foram feitas três curvas de ROC para os modelos, considerando cada uma das três saídas possíveis como positiva, enquanto as outras duas saídas são consideradas negativas para fins da avaliação.

As três curvas obtidas são muito semelhantes, por isso apenas a curva que tem a saída 1 (pouco estresse) como positiva é apresentada. A Figura 3 mostra os resultados em formato de curva de ROC.

Figura 3 – Curva de ROC dos algoritmos avaliados



Fonte: Própria autora

A partir da curva de ROC comprova-se os resultados de classificação apresentados na tabela 4. O algoritmo *Random Forest* se encontra muito próximo do contorno superior, apresentando um ótimo desempenho na classificação dos níveis de estresse. Ainda com um bom desempenho, constata-se as curvas dos algoritmos *kNearest Neighbors* e *Neural Network*, que também estão próximos do contorno superior do gráfico. Por outro lado, os

algoritmos *Support Vector Machine* e *Logistic Regression* apresentam um desempenho mais próximo a um classificador randômico, sendo menos adequados para uma aplicação real.

As demais curvas obtidas podem ser vistas no apêndice C, seção C.1.7. A seguir, são apresentados explicadores para o algoritmo *Random Forest*, em busca de esclarecimentos sobre como suas decisões são tomadas, e quais as características mais influentes.

5.2 Explicação do *Random Forest*

Como o *Random Forest* apresentou melhor resultado, os explicadores foram aplicados ao modelo previamente treinado, de como a explicar o *Random Forest*. Os explicadores foram obtidos por meio da biblioteca *shap*.

5.2.1 A biblioteca *shap*

Primeiramente, é importante salientar algumas informações referentes à biblioteca *shap*, que também foi utilizada para explicação dos modelos apresentados por Tseng et al. (2020) e Dave et al. (2020).

A biblioteca *shap* faz uso de medidas denominadas *shap values*, introduzidas por Lundberg e Lee (2017). Essas medidas são uma representação unificada de importância de características, baseadas em *Shapley Values*, que utilizam equações de teoria de jogos para obter valores. Enquanto *Shapley values* quantificam a contribuição que cada jogador traz ao jogo, *shap values* quantificam a contribuição que cada característica traz para a predição do modelo, fazendo com que *shap* seja a biblioteca de explicabilidade mais avançada até o momento (MAZZANTI, 2020).

5.2.2 Gráfico de importância de características

A Figura 4 mostra o gráfico das características mais importantes do *Random Forest* gerado com o uso da biblioteca *sklearn*. Ele traz o nome de todas as características presentes na base de dados, associados a um valor de importância relativa. Quanto maior a importância relativa, maior é o impacto da respectiva característica na saída do modelo.

O gráfico aponta que as seis características mais importantes são decisivas para a saída do modelo, pois possuem mais do que o dobro de importância que as demais. Também é possível perceber que a característica *TEMP_Mean*, média de temperatura da pele, é a característica que mais impacta o resultado. Em segundo lugar, tem-se *HRR_Mean*, que representa a média de frequência cardíaca. Vale ressaltar que o gráfico de importância de características é considerado uma explicação global, pois é válido para todo o modelo.

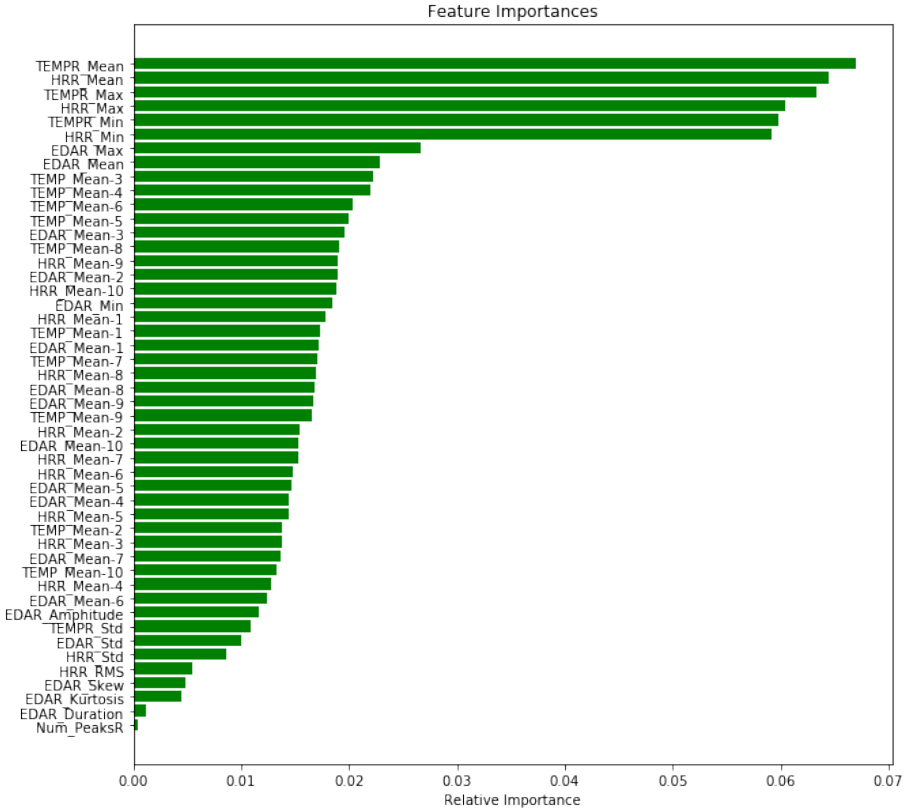
5.2.3 Gráfico de dependência parcial

A Figura 5 mostra um gráfico de dependência parcial do *Random Forest* para a característica *TEMPR_Max*, levando em consideração a característica *HRR_Min* quando a saída considerada é 2 (alto estresse).

O eixo horizontal do gráfico traz a proporção de valores da característica *TEMPR_Max*, o valor máximo da temperatura da pele. Quanto mais próximo de 0.0, mais próximo do menor valor medido para essa característica, e quanto mais próximo de 1.0, mais próximo do maior valor medido para essa característica.

De forma semelhante, as cores dos marcadores indicam a proporção do valor de *HRR_Min*, o valor mínimo de frequência cardíaca. Quanto mais azul, mais próximo do

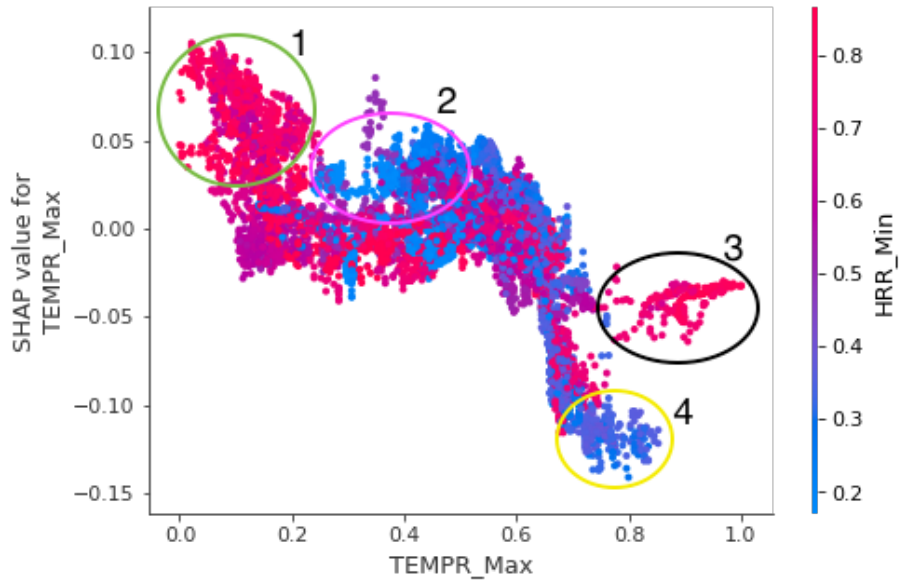
Figura 4 – *Feature Importance* do modelo *Random Forest*



Fonte: Própria autora

valor mínimo medido para essa característica, e quanto mais próximo de vermelho, mais próximo do valor máximo medido para essa característica. Dessa forma, o eixo vertical indica o impacto da relação das características *TEMPR_Max* e *HRR_Min* no modelo.

Figura 5 – Gráfico de dependência parcial do modelo *Random Forest*



Fonte: Própria autora

Para facilitar o entendimento, foram destacadas quatro regiões do gráfico que merecem atenção. São elas:

- A região 1, destacada em verde, mostra pontos vermelhos no canto superior esquerdo do gráfico. Ela indica que, quando a frequência cardíaca mínima possui um valor alto, e a temperatura da pele máxima possui um valor baixo, a característica de temperatura da pele contribui positivamente para o nível de estresse detectado ser 2 (muito estresse).
- A região 2, destacada em rosa, mostra pontos azuis, também no canto superior esquerdo, mas mais próximo do centro do gráfico. Ela indica que, a medida que o valor mínimo de frequência cardíaca diminui e o valor máximo de temperatura da pele aumenta, o impacto da temperatura da pele máxima diminui, se aproximando de 0.
- A região 3, destacada em preto, indica pontos vermelhos no lado direito do gráfico, um pouco abaixo do valor 0 de impacto. Ela indica que valores altos de frequência cardíaca mínima e de temperatura da pele máxima possuem um pequeno impacto negativo, contribuindo para o diagnóstico de pouco ou nenhum estresse.
- A região 4, destacada em amarelo, mostra pontos azuis no canto inferior direito do gráfico. Ela indica que valores altos de temperatura da pele máxima, combinados com valores baixos de frequência cardíaca mínima, contribuem para uma saída indicativa de pouco ou nenhum estresse.

Destaca-se que o gráfico de dependência parcial é um tipo de explicação local, visto que considera apenas uma saída alvo. Portanto, é válido somente quando a saída encontrada pelo modelo é a considerada na sua elaboração.

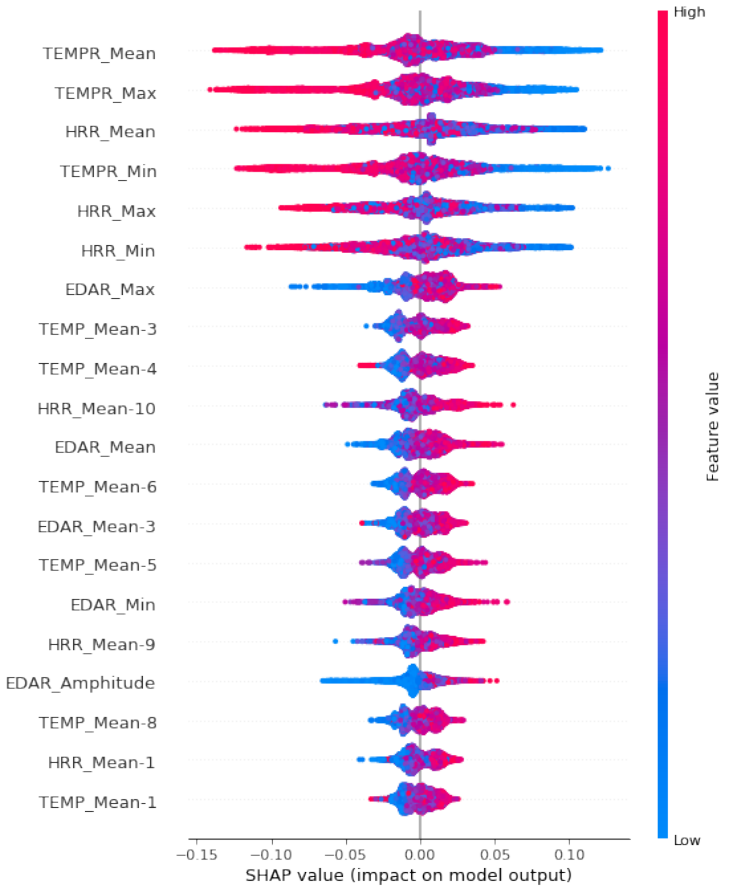
5.2.4 Gráfico de sumário

O gráfico de sumário possui esse nome pois visa "resumir" o impacto das características associado ao seu valor. Pontos vermelhos indicam que o valor da característica é alto, maior do que a maioria dos outros valores atribuídos a essa característica. De forma semelhante, pontos azuis correspondem a valores baixos, menores do que a maioria dos outros valores atribuídos a determinada característica. Cada linha corresponde a uma característica, indicada à esquerda, e cada ponto corresponde a uma instância de dado.

O eixo horizontal indica o impacto, positivo ou negativo, que a característica tem para o valor de saída considerado. No meio do gráfico existe uma linha vertical indicando impacto 0, ponto em que as características não impactam na saída do modelo. Pontos mais para a esquerda impactam negativamente na saída, e pontos para a direita impactam positivamente, contribuindo para que a saída seja a considerada.

A Figura 6 apresenta o gráfico de sumário do *Random Forest*, considerando a saída 2 (alto estresse).

Figura 6 – Gráfico de sumário do *Random Forest*



Fonte: Própria autora

Ressalta-se que as características são apresentadas em ordem de *SHAP values*, diferentemente da Figura 4, que considera os cálculos de importância da biblioteca *sklearn*. Além disso, percebe-se que as seis características mais importantes seguem o mesmo

comportamento: com valores baixos, as características contribuem para a saída de alto estresse, enquanto com valores altos existe tendência para diagnósticos de baixo ou nenhum estresse.

Dessa informação, pode-se tirar premissas como "Quanto menor é a média de temperatura da pele (*TEMPR_Mean*), mais provável é que o participante esteja estressado". A mesma premissa é verdadeira considerando as seguintes características:

- temperatura máxima da pele (*TEMPR_Max*),
- temperatura mínima da pele (*TEMPR_Min*),
- média da frequência cardíaca (*HRR_Mean*),
- frequência cardíaca máxima (*HRR_Max*),
- frequência cardíaca mínima (*HRR_Min*).

De forma semelhante, a sétima característica mais importante, atividade eletrodérmica máxima (*EDAR_Max*), apresenta um comportamento contrário, em que atividade eletrodérmica menor contribui para que a saída seja pouco ou nenhum estresse.

É notável também que, em se tratando de características menos importantes, os pontos são mais próximos da linha de nenhum impacto, e não fica clara a relação entre os valores das características e o impacto sobre a saída do modelo.

Ainda, o gráfico de sumário se trata de uma explicação local, dado que leva em consideração apenas uma das possíveis saídas. Desse modo, sua explicação somente é válida quando a saída encontrada pelo modelo é a mesma passada para a função da biblioteca *shap* que o gera.

A seguir, são apresentados os resultados do treinamento dos cinco modelos apenas com as dez características mais importantes dos três modelos com melhor desempenho.

5.3 Treinamento de algoritmos com características mais importantes

Como última etapa do desenvolvimento deste trabalho, tem-se o treinamento dos algoritmos com as 10 características mais importantes para os três resultados mais bem avaliados. O treinamento com menos características pode ajudar a melhorar o tempo de treinamento dos modelos, bem como avaliar quais características realmente são necessárias para a detecção de estresse ocupacional, permitindo a simplificação do pré-processamento dos dados. Esta investigação serve para auxiliar na compreensão sobre volume de dados necessário para a identificação dos estados de estresse. Ou seja, é possível obter resultados similares com menos características em cada um dos algoritmos estudados?

O trabalho de Singh, Thakur e Sharma (2016) traz um levantamento sobre o comportamento e fatores impactantes para cada um dos algoritmos avaliados, indicando o que seria esperado quando o modelo for treinado com menos características. Os fatores são indicados a seguir.

- *k-Nearest Neighbors* melhora a performance quando se tem menos características, pois todas devem ser visitadas durante o seu treinamento. É mais eficiente quando treinado com menos características.

- *Logistic regression* apresenta classificações melhores quando se tem mais características e mais instâncias de dados, portanto, não se beneficiaria da redução de características.
- *Neural Networks* dependem de parâmetros como neurônios ocultos e pesos iniciais. Valores inapropriados podem resultar num tempo de treinamento maior e desempenho ruim. Por outro lado, esse modelo é capaz de reconhecer vários relacionamentos entre características e saídas, tendo um bom desempenho esperado mesmo com um número menor de características.
- *Random Forest* tem uma performance melhor quando se tem menor correlação entre as características, apesar de ser ideal para problemas de larga escala com muitas características. Portanto, se espera um treinamento mais rápido, mas certa redução de desempenho quando se utiliza menos características.
- *Support Vector Machine* é simplificado quando o número de características diminui, pois diminui também o número de dimensões a serem correlacionadas durante o treinamento.

Para o treinamento, primeiramente foi necessário encontrar as 10 características mais importantes dos 3 melhores algoritmos testados no estudo: *Random Forest*, *k-Nearest Neighbors* e *Neural Network*.

A Figura 4 mostra o levantamento de importância de características do *Random Forest*. Em adição, as Figuras 7 e 8 mostram as características mais importantes para o *k-Nearest Neighbors* e a *Neural Network*. Com isso, foi possível realizar o treinamento dos cinco algoritmos novamente, utilizando somente as dez primeiras linhas de cada gráfico de *Feature Importance*.

Para filtrar as colunas desejadas dos dados, foi utilizada a função *concat* da biblioteca *pandas*, utilizada para concatenar as colunas especificadas dos dados. Dessa forma, todos os cinco modelos foram treinados novamente, e os mesmos parâmetros de desempenho foram levantados.

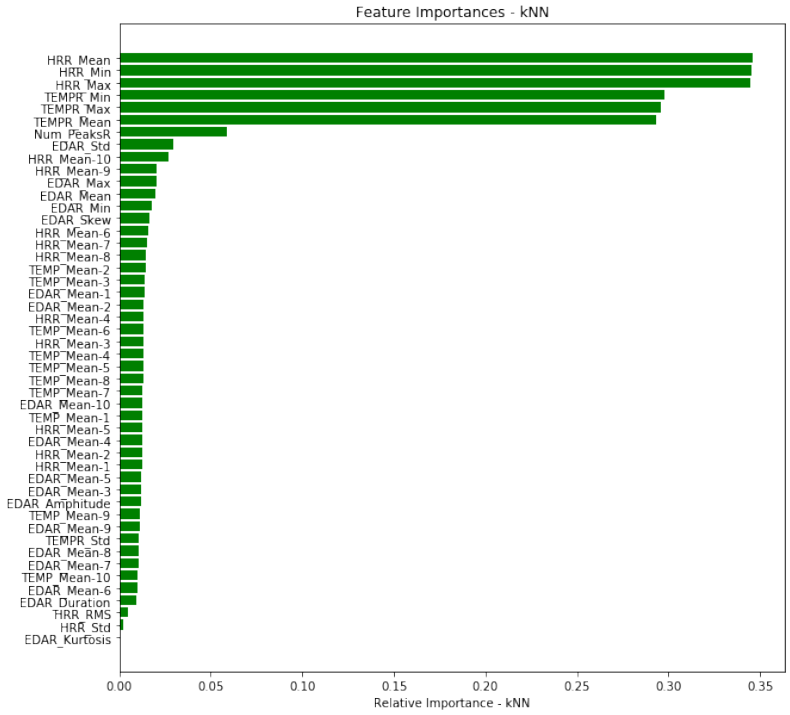
A tabela 5 mostra os parâmetros encontrados para os modelos treinados apenas com as 10 características mais importantes para o *Random Forest*.

Tabela 5 – Resultados da avaliação de modelos com as 10 características de maior impacto para *Random Forest*

	Modelo	**AUC**	**Acurácia**	**F1**	***Recall***
(1)	**RF**	0.992	0.947	0.934	0.927
(2)	**kNN**	0.990	0.942	0.929	0.928
(3)	**NN**	0.870	0.733	0.661	0.655
(4)	**LR**	0.736	0.604	0.441	0.489
(5)	**SVM**	0.735	0.613	0.449	0.498

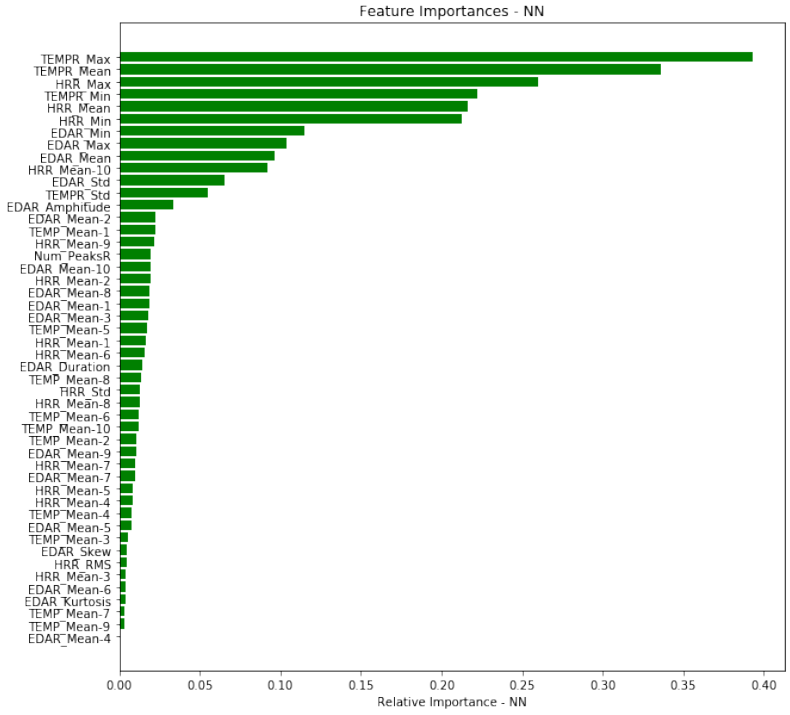
Fonte: Própria Autora.

Figura 7 – *Feature Importance* do modelo *k-Nearest Neighbors*



Fonte: Própria autora

Figura 8 – *Feature Importance* do modelo *Neural Network*



Fonte: Própria autora

É notável que, quando os resultados apresentados na tabela 5 são comparados com os resultados apresentados na tabela 4, consta-se que não houve redução drástica no desempenho de nenhum dos modelos. Por outro lado, vale destacar a melhora significativa no desempenho do modelo kNN, que se aproxima consideravelmente do modelo RF, classificado em primeiro lugar. Em adição, a classificação dos algoritmos se mantém a mesma.

A tabela 6 mostra os parâmetros encontrados para os algoritmos treinados apenas com as 10 características mais importantes para o *k-Nearest Neighbors*.

Tabela 6 – Resultados da avaliação de modelos com as 10 características de maior impacto para *k-Nearest Neighbors*

	Modelo	**AUC**	**Acurácia**	**F1**	***Recall***
(1)	**RF**	0.986	0.925	0.911	0.905
(2)	**kNN**	0.960	0.869	0.846	0.845
(3)	**NN**	0.841	0.712	0.640	0.634
(4)	**LR**	0.729	0.591	0.435	0.480
(5)	**SVM**	0.720	0.601	0.439	0.487

Fonte: Própria Autora.

Nesse caso conforme a tabela 6, todos os modelos apresentaram uma piora de desempenho em relação aos resultados da tabela 4, mas pouco notável na prática.

Por fim, a tabela 7 mostra os parâmetros encontrados para os algoritmos treinados apenas com as 10 características mais importantes para a *Neural Network*.

Tabela 7 – Resultados da avaliação de modelos com as 10 características de maior impacto para *Neural Network*

	Modelo	**AUC**	**Acurácia**	**F1**	***Recall***
(1)	**RF**	0.991	0.944	0.932	0.927
(2)	**kNN**	0.988	0.938	0.925	0.925
(3)	**NN**	0.875	0.748	0.674	0.667
(4)	**LR**	0.739	0.604	0.440	0.489
(5)	**SVM**	0.732	0.614	0.449	0.498

Fonte: Própria Autora.

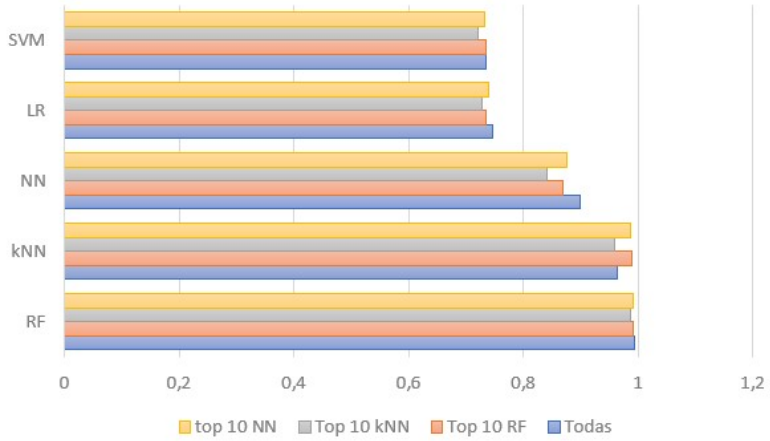
A tabela 7 aponta uma leve piora no desempenho de todos os resultados em relação a tabela 4, com exceção do modelo kNN. Assim como foi visto na tabela 5, o *k-Nearest Neighbors* apresentou uma melhora significativa de desempenho, também se aproximando do *Random Forest* na primeira posição. Esses resultados também não alteram a classificação dos algoritmos.

A Figura 9 apresenta os valores de AUC de cada algoritmo associados com as características utilizadas durante o treinamento.

6 Resultados e Discussões

Neste tópico, são apresentados os principais resultados desse trabalho, bem como se procede uma comparação e discussão com resultados de outros trabalhos presentes na

Figura 9 – Variação de AUC de cada modelo com a alteração das características de treinamento



Fonte: Própria autora

fundamentação teórica e revisão da literatura.

6.1 Resultado de avaliação de algoritmos de aprendizado de máquina

Como principal resultado da etapa de avaliação de algoritmos de aprendizado de máquina, tem-se o melhor desempenho dentre os avaliados foi o *Random Forest*. Em trabalhos relacionados (DAVE et al., 2020), o *XGBoost*, modelo de conjunto de árvores semelhante ao *Random Forest* introduzido por Chen e Guestrin (2016) foi incluído em avaliações, diferentemente desta pesquisa.

Além disso, a pesquisa de Tseng et al. (2020) traz uma avaliação de algoritmos de aprendizado de máquina que concluiu que a combinação de *XGBoost* com *Random Forest* em um único algoritmo possui um melhor desempenho dentre todos os avaliados, que incluíram SVM, LR e árvore de decisão simples. Complementarmente, na pesquisa de Tseng et al. (2020), o *XGBoost* e o *Random Forest* ficaram com o segundo e terceiro lugares respectivamente, indicando o bom desempenho dos algoritmos de conjuntos de árvores. Destaca-se que nesta pesquisa não se fez combinação de algoritmos e também não foi testado o XGBoost.

Em adição, o trabalho de Chalabianloo et al. (2022) escolheu quatro classificadores como mais promissores para o treinamento. Esses classificadores foram SVM, *Random Forest*, *Extremely Randomized Tree* e *Light Gradient Boosting Machine*, dois deles os mesmos desta pesquisa. Os dois últimos algoritmos são também baseados em conjunto de árvores, e foram adicionados ao estudo após os autores perceberem resultados promissores obtidos pelo modelo *Random Forest*. Nesse caso, o *Random Forest* ficou em segundo lugar, e o *Extremely Randomized Tree*, apresentou o melhor desempenho, foi escolhido para ser explicado. Mesmo que o *Random Forest* não tenha sido escolhido, ficou a frente dos demais, demonstrando a sua importância.

Ainda, o trabalho de Morales et al. (2022b) apresenta um levantamento de tipos de modelos de aprendizado de máquina utilizados, contando com as características analisadas para a detecção de estresse. Apesar de SVM ser um modelo predominante, presente em

muitos dos trabalhos levantados, não obteve um bom desempenho neste trabalho para a base de dados e parâmetros utilizados.

Por outro lado, o trabalho de Bahani, Moujabbir e Ramdani (2021) também realizou uma comparação entre modelos de aprendizado de máquina e o *Random Forest* apresentou um dos piores desempenhos entre os algoritmos avaliados, que incluem SVM, kNN, NN e Naive Bayes. Esses resultados destoam dos resultados obtidos pelo estudo apresentado.

6.2 Comparação de impactos de biomarcadores

Essa etapa do desenvolvimento trouxe como resultado uma explicação do *Random Forest*, esclarecendo a importância das características do modelo, bem como gráficos que associam seus valores e suas contribuições para as possíveis saídas. Ainda, é percebível que a importância de características varia dependendo do algoritmo a ser explicado.

Vale ressaltar que, dentre todos os trabalhos analisados, o único que aborda explicabilidade de estresse, cujos resultados podem ser comparados com os atingidos neste, é o trabalho de Chalabianloo et al. (2022). Já o trabalho de Hosseini et al. (2022) não apresenta explicações para as saídas obtidas, e trabalhos como o de Madanu et al. (2022), Pawar et al. (2020) e Dave et al. (2020) têm foco em áreas distintas da medicina, não oferecendo nenhuma possibilidade de comparação de explicações e impactos de biomarcadores para a detecção de estresse ocupacional.

O trabalho de Chalabianloo et al. (2022) fez uso de dados coletados em laboratório, utilizando sete dispositivos vestíveis diferentes. Esses dados também foram categorizados entre estresse baixo, médio e alto por meio de análise de contextos em que foram coletados. Por esse motivo, a quantidade e os tipos de dados são distintos dos utilizados nesse trabalho. A seleção de características foi feita por meio de eliminação recursiva de características com validação cruzada, resultando em 12 características relacionadas a frequência cardíaca e entropia. As característica mais importantes para os algoritmos treinados foram o intervalo entre batimentos cardíacos e a entropia aproximada. Posteriormente, os autores adicionaram medidas de atividade eletrodérmica, resultando no aumento do desempenho dos modelos. Porém, não é apresentada a importância de características associadas a atividade eletrodérmica, e modelos treinados com esses dados não são explicados.

O trabalho de Chalabianloo et al. (2022) também apresenta que a importância de características varia de acordo com o modelo explicado, o que é corroborado pelos levantamentos de importância de características feito neste trabalho. De forma geral, é reforçada a importância de características relacionadas a frequência cardíaca e atividade eletrodérmica para a detecção de estresse, enfatizando que o uso de mais biomarcadores, além de frequência cardíaca, contribui para que o modelo se torne mais confiável.

6.3 Redução de características de treinamento de modelos

O treinamento com menos características resultou em uma pequena redução no desempenho dos algoritmos testados, com exceção do *k-Nearest Neighbors*. Quando treinado com as 10 características mais importantes dos algoritmos *Random Forest* e *Neural Network*, o kNN apresentou uma melhora significativa do desempenho, comprovando a sua eficiência em casos de dados com menos características, como mencionado por Singh, Thakur e Sharma (2016). Apesar de melhorar o seu desempenho, em nenhum caso o kNN superou o desempenho do *Random Forest*, que se manteve em primeiro lugar em todas as avaliações

realizadas. Até o momento, no levantamento realizado não foi encontrado nenhum trabalho que utilize a importância de características para reduzir o tamanho da base de dados e otimizar o treinamento de algoritmos.

Quanto ao tempo de treinamento, não houve diferença notável. Porém, constata-se que quando o treinamento de algoritmos for feito com dados de aplicações reais, de tamanho muito maior, a diferença será mais notável. O tempo de treinamento tende a ser maior no caso de LR, e menor no caso de kNN, RF e SVM, ou indiferente no caso de NN de acordo com as características mencionadas por Singh, Thakur e Sharma (2016).

7 Conclusão e Trabalhos Futuros

O presente trabalho teve como objetivo avaliar modelos de aprendizado de máquina treinados com dados estatísticos de biomarcadores retirados da base de dados *AffectiveRoad* para detectar diferentes níveis de estresse. Para alcançar tal objetivo, pelo levantamento bibliográfico foram identificadas as vantagens da aplicação de inteligência artificial explicável na área da medicina, bem como características de aplicações explicáveis para diferenciar modelos caixa preta e caixa transparente. Em adição, foram indetificados alguns algoritmos caixa preta utilizados na detecção de estresse em trabalhos anteriores, bem como explicadores adequados para esclarecer o comportamento desses modelos. Por fim, foi feito um estudo sobre os biomarcadores associados com estresse, e sua forma de coleta, para determinar quais biomarcadores são mais predominantemente aplicados em sistemas detectores de estresse.

A realização desse trabalho seguiu com revisão da literatura para levantamento de trabalhos relacionados, e da seleção de algoritmos caixa preta de aprendizado de máquina e explicadores a serem utilizados. Logo depois, foram feitos programas em Python para avaliar e explicar os modelos de aprendizado de máquina selecionados, treinados com uma base de dados pré-processada disponível publicamente. Após isso, o melhor desempenho foi explicado por meio de explicadores considerados ideais durante a etapa de revisão bibliográfica. Por fim, as características mais importantes dos três algoritmos que apresentaram o melhor desempenho foram utilizadas para um novo treinamento dos modelos, visando avaliar a diferença de desempenho quando treinados com menos características.

Quanto aos resultados desse trabalho, a primeira etapa apontou o *Random Forest* como melhor avaliado, seguido pelos *k-Nearest Neighbors* e *Neural Network*. Sendo assim, explicadores foram aplicados sobre o *Random Forest*, apontando importância de características, dependências parciais entre características e sumário do impacto das características nas saídas dependendo dos seus valores.

O levantamento de importância de características aponta que dados relacionados a temperatura da pele e frequência cardíaca possuem maior importância para o modelo. Além disso, o gráfico de sumário do *Random Forest* aponta que valores altos de temperatura da pele e de frequência cardíaca apontam para pouco ou nenhum estresse. Por outro lado, uma alta atividade eletrodérmica é um indicativo de estresse, ainda que possua menos importância para o modelo. Ainda, o gráfico de dependência parcial aponta que, ainda que o valor de temperatura da pele seja alto, o valor da frequência cardíaca pode aumentar ou diminuir o seu impacto no modelo. São destacadas regiões de destaque do gráfico de dependência parcial, visando demonstrar como o valor de uma característica influencia no impacto do valor de outra característica.

O treinamento dos algoritmos com menos características resultou em uma pequena piora no desempenho dos modelos treinados, com exceção do *k-Nearest Neighbors*, que teve um aumento significativo de performance quando treinado com as 10 características mais importantes para o *Random Forest* e a*Neural Network*. Destaca-se que a diferença de desempenho dos algoritmos treinados com 10 ou 48 características é insignificante na prática, o que indica que usar todas as características não é vantajoso ao se considerar a relação custo-benefício.

Apesar de existirem estudos relacionados a explicabilidade de diagnósticos na área médica, esse trabalho é inovador por trazer avaliação e explicação de alguns algortimos focados no diagnóstico de estresse ocupacional em profissionais da saúde, trazendo análises a partir de explicadores como *Feature Importance* e *Summary Plot*, com a adição do gráfico de dependência parcial, notavelmente ausente em todos os trabalhos relacionados a estresse analisados. Em adição, não foi encontrado até o momento, na literatura analisada nenhum trabalho que realizasse a otimização do treinamento de algoritmos por meio da seleção de características baseada na sua importância, algo que é apresentado, de forma inédita, no presente trabalho, demonstrando assim, contribuições para o avanço do conhecimento sobre este tema. Além disso, o trabalho apresenta ferramentas que podem ser utilizadas por organizações de saúde para a detecção do estresse ocupacional, na prática, assegurando a acurácia dos resultados.

Por fim, esse estudo apresenta algumas limitações. Em primeiro lugar, o treinamento dos algoritmos de aprendizado de máquina foi feito utilizando apenas características estatísticas derivadas de três biomarcadores, sendo que existem mais nove biomarcadores que já foram utilizados para a detecção de estresse em outros trabalhos. Em segundo lugar, alguns algoritmos já utilizados para diagnósticos na área da saúde não fizeram parte da avaliação, e não foi feita nenhuma combinação de algoritmos para avaliação final. Ademais, não foi utilizada nenhuma característica de fator emocional para o treinamento dos modelos, dificultando a distinção entre eustresse e distresse. Além disso, não foi feita a tradução de explicações de modo que as mesmas possam ser facilmente entendidas por profissionais da saúde, bem como há nenhuma integração com sistema de recomendação para remediar o estresse detectado. Para tanto, explicações locais, válidas para apenas uma instância de dados, podem ser aplicadas e traduzidas para uma sentença em linguagem natural, de modo a tornar o entendimento dessas explicações muito mais rápido e fácil.

Dadas as limitações mencionadas anteriormente, pode-se sugerir alguns trabalhos futuros. Sugere-se a adição de dados obtidos por um sensor químico, capaz de medir taxas do hormônio cortisol no sangue como descrito na investigação de vestíveis para identificação de estresse em (MORALES et al., 2022). Também, é possível melhorar a base de dados de treinamento com a adição de características emocionais robustas, que podem ser obtidas através de questionários, para que o estado emocional do profissional seja levado em consideração para a detecção de estresse. Sugere-se também a inclusão de outros algoritmos e combinados para a avaliação, visando comparar mais modelos em busca de melhores resultados. E para finalizar, os resultados do estudo pode integrar com um sistema de recomendação inteligente que, após uma detecção de estresse alto, ofereça sugestões de psicólogos ou psiquiatras com horários de atendimento, utilizando explicações locais facilmente entendíveis como sentenças em linguagem natural conforme apresentado por (MORALES et al., 2022b). Inclusive, recomenda-se o uso de multiagentes para a coleta de dados fisiológicos, emocionais e recomendações inteligentes, evitando que a presença de agentes estressores no ambiente de trabalho afete a saúde física e mental dos profissionais de saúde.

Referências

ADADI, A.; BERRADA, M. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, Institute of Electrical and Electronics Engineers Inc., v. 6, p. 52138–52160, 9 2018. ISSN 21693536. Citado na página [8].

AMMAR, N.; SHABAN-NEJAD, A. Explainable artificial intelligence recommendation system by leveraging the semantics of adverse childhood experiences: Proof-of-concept prototype development. *JMIR Medical Informatics*, JMIR Publications Inc., v. 8, n. 11, 11 2020. ISSN 22919694. Citado na página [20].

ANTWARG, L. et al. Explaining anomalies detected by autoencoders using Shapley Additive Explanations[Formula presented]. *Expert Systems with Applications*, Elsevier Ltd, v. 186, 12 2021. ISSN 09574174. Citado na página [22].

BAHANI, K.; MOUJABBIR, M.; RAMDANI, M. An accurate fuzzy rule-based classification systems for heart disease diagnosis. *Scientific African*, Elsevier B.V., v. 14, 11 2021. ISSN 24682276. Citado (2) vezes nas páginas [20 e 37].

BENFANTE, A. et al. *Traumatic Stress in Healthcare Workers During COVID-19 Pandemic: A Review of the Immediate Impact*. [S.l.]: Frontiers Media S.A., 2020. Citado na página [7].

BIAU, G.; SCORNET, E. A random forest guided tour. *Test*, Springer New York LLC, v. 25, n. 2, p. 197–227, 6 2016. ISSN 11330686. Citado na página [13].

CHALABIANLOO, N. et al. Application level performance evaluation of wearable devices for stress classification with explainable AI. *Pervasive and Mobile Computing*, Elsevier B.V., v. 87, 12 2022. ISSN 15741192. Citado (6) vezes nas páginas [8, 15, 16, 21, 36 e 37].

CHEN, T.; GUESTRIN, C. XGBoost: A scalable tree boosting system. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. [S.l.]: Association for Computing Machinery, 2016. v. 13-17-August-2016, p. 785–794. ISBN 9781450342322. Citado na página [36].

DAVE, D. et al. *Explainable AI meets Healthcare: A Study on Heart Disease Dataset*. [S.l.], 2020. Disponível em: <https://www.kaggle.com/cherngs/>. Citado (5) vezes nas páginas [8, 19, 28, 36 e 37].

DORAN, D.; SCHULZ, S.; BESOLD, T. R. What Does Explainable AI Really Mean? A New Conceptualization of Perspectives. 10 2017. Disponível em: <http://arxiv.org/abs/1710.00794>. Citado (4) vezes nas páginas [9, 10, 11 e 14].

GHASSEMI, M.; OAKDEN-RAYNER, L.; BEAM, A. L. *The false hope of current approaches to explainable artificial intelligence in health care*. [S.l.], 2021. v. 3, 745–50 p. Disponível em: <www.thelancet.com/>. Citado (2) vezes nas páginas [8 e 18].

GUIDOTTI, R. et al. A survey of methods for explaining black box models. *ACM Computing Surveys*, Association for Computing Machinery, v. 51, n. 5, 8 2018. ISSN 15577341. Citado (8) vezes nas páginas [9, 10, 11, 12, 13, 14, 15 e 22].

GUNNING, D. Explainable artificial intelligence (xai). *Defense advanced research projects agency (DARPA), nd Web*, v. 2, n. 2, p. 1, 2017. Citado na página [8].

HOLZINGER, A. et al. What do we need to build explainable AI systems for the medical domain? 12 2017. Disponível em: <http://arxiv.org/abs/1712.09923>. Citado (3) vezes nas páginas [8, 14 e 17].

HOO, Z. H.; CANDLISH, J.; TEARE, D. What is an ROC curve? *Emergency Medicine Journal*, BMJ Publishing Group, v. 34, n. 6, p. 357–359, 6 2017. ISSN 14720213. Citado na página [27].

HOSSAIN, M. S.; MUHAMMAD, G.; GUIZANI, N. Explainable AI and mass surveillance system-based healthcare framework to combat COVID-I9 like pandemics. *IEEE Network*, Institute of Electrical and Electronics Engineers Inc., v. 34, n. 4, p. 126–132, 7 2020. ISSN 1558156X. Citado na página [19].

HOSSEINI, S. et al. A multimodal sensor dataset for continuous stress detection of nurses in a hospital. *Scientific Data*, Nature Research, v. 9, n. 1, 12 2022. ISSN 20524463. Citado (8) vezes nas páginas [8, 15, 20, 21, 24, 25, 26 e 37].

INAM, R. et al. *Explainable AI – how humans can trust AI*. 2021. Citado (3) vezes nas páginas [9, 10 e 11].

Laurence Bardin. Análise de conteúdo. *Edições 70*, p. –229, 2011. Citado na página [21].

LAVALLEY, M. P. *Logistic regression*. 2008. 2395–2399 p. Citado na página [14].

LINARDATOS, P.; PAPASTEFANOPOULOS, V.; KOTSIANTIS, S. *Explainable ai: A review of machine learning interpretability methods*. [S.l.]: MDPI AG, 2021. 1–45 p. Citado (2) vezes nas páginas [9 e 11].

LOPEZ-MARTINEZ, D.; EL-HAOUIJ, N.; PICARD, R. Detection of Real-world Driving-induced Affective State Using Physiological Signals and Multi-view Multi-task Machine Learning. 7 2019. Disponível em: <http://arxiv.org/abs/1907.09929>. Citado (4) vezes nas páginas [8, 20, 21 e 22].

LUNDBERG, S.; LEE, S.-I. A Unified Approach to Interpreting Model Predictions. 5 2017. Disponível em: <http://arxiv.org/abs/1705.07874>. Citado na página [28].

MADANU, R. et al. Explainable AI (XAI) Applied in Machine Learning for Pain Modeling: A Review. *Technologies*, MDPI AG, v. 10, n. 3, p. 74, 6 2022. Citado (2) vezes nas páginas [18 e 37].

MAZZANTI, S. *SHAP Values Explained Exactly How You Wished Someone Explained to You*. 2020. Citado na página [28].

MILLER, T. *Explanation in artificial intelligence: Insights from the social sciences*. [S.l.]: Elsevier B.V., 2019. 1–38 p. Citado na página [10].

MONTAVON, G.; SAMEK, W.; MÜLLER, K. R. *Methods for interpreting and understanding deep neural networks*. [S.l.]: Elsevier Inc., 2018. 1–15 p. Citado (5) vezes nas páginas [9, 11, 13, 14 e 22].

MORALES, A. et al. Occupational stress monitoring using biomarkers and smartwatches: A systematic review. *Sensors*, v. 22, n. 17, 2022. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/22/17/6633>. Citado (4) vezes nas páginas [15, 16, 18 e 39].

MORALES, A. S.; OURIQUE, F. d. O.; CAZELLA, S. C. A comprehensive review on the challenges for intelligent systems related with internet of things for medical decision. In: _____. *Enhanced Telemedicine and e-Health: Advanced IoT Enabled Soft Computing Framework*. Cham: Springer International Publishing, 2021. p. 221–240. ISBN 978-3-030-70111-6. Disponível em: <https://doi.org/10.1007/978-3-030-70111-6_11>. Citado na página [9].

MORALES, A. S. et al. A biomarker-based model to assist the identification of stress in health workers involved in coping with covid-19. In: _____. *The Science behind the COVID Pandemic and Healthcare Technology Solutions*. Cham: Springer International Publishing, 2022. p. 485–500. ISBN 978-3-031-10031-4. Disponível em: <https://doi.org/10.1007/978-3-031-10031-4_22>. Citado na página [15].

MORALES, A. S. et al. Exploring interpretable machine learning methods and biomarkers to classifying occupational stress of the health workers. In: _____. *Machine Learning for Smart Environments/Cities: An IoT Approach*. Cham: Springer International Publishing, 2022. p. 105–124. ISBN 978-3-030-97516-6. Disponível em: <https://doi.org/10.1007/978-3-030-97516-6_6>. Citado (7) vezes nas páginas [7, 13, 15, 18, 22, 36 e 39].

MURDOCH, W. J. et al. Interpretable machine learning: definitions, methods, and applications. 1 2019. Disponível em: <http://arxiv.org/abs/1901.04592http://dx.doi.org/10.1073/pnas.1900654116>. Citado na página [14].

NOHARA, Y. et al. Explanation of machine learning models using shapley additive explanation and application for real data in hospital. *Computer Methods and Programs in Biomedicine*, Elsevier Ireland Ltd, v. 214, 2 2022. ISSN 18727565. Citado na página [14].

OSÓRIO, F. L. et al. Risk and Protective Factors for the Mental Health of Brazilian Healthcare Workers in the Frontline of COVID-19 Pandemic. *Frontiers in Psychiatry*, Frontiers Media S.A., v. 12, 7 2021. ISSN 16640640. Citado na página [7].

PABLO, G. Salazar de et al. *Impact of coronavirus syndromes on physical and mental health of health care workers: Systematic review and meta-analysis*. [S.l.]: Elsevier B.V., 2020. 48–57 p. Citado na página [7].

PAWAR, U. et al. *Explainable AI in Healthcare*. [S.l.], 2020. Citado (5) vezes nas páginas [9, 10, 11, 19 e 37].

SINGH, A.; THAKUR, N.; SHARMA, A. *A Review of Supervised Machine Learning Algorithms*. [S.l.: s.n.], 2016. 1310–1315 p. ISBN 9789380544205. Citado (4) vezes nas páginas [13, 32, 37 e 38].

TJOA, E.; GUAN, C. A Survey on Explainable Artificial Intelligence (XAI): Towards Medical XAI. 7 2019. Disponível em: <http://arxiv.org/abs/1907.07374http://dx.doi.org/10.1109/TNNLS.2020.3027314>. Citado (2) vezes nas páginas [8 e 17].

TSENG, P. Y. et al. Prediction of the development of acute kidney injury following cardiac surgery by machine learning. *Critical Care*, BioMed Central, v. 24, n. 1, 7 2020. ISSN 1466609X. Citado (4) vezes nas páginas [8, 20, 28 e 36].

VOS, G. et al. Machine Learning for Stress Monitoring from Wearable Devices: A Systematic Literature Review. 9 2022. Disponível em: <http://arxiv.org/abs/2209.15137>. Citado (3) vezes nas páginas [8, 21 e 22].

ZHANG, Z. Introduction to machine learning: K-nearest neighbors. *Annals of Translational Medicine*, AME Publishing Company, v. 4, n. 11, 6 2016. ISSN 23055847. Citado na página [13].

Apêndice

A Tabela de acrônimos

Acrônimo	**Significado**
EDA	*Electrodermal Activity*
HR	*Heart Rate*
TEMP	*skin Temperature*
BVP	*Blood Volume Pulse*
ECG	*Electrocardiogram*
EMG	*Electromyogram*
BT	*Body Temperature*
RESP	*Respiration*
ACC	*Three-axis Acceleration*
GSR	*Galvanic Skin Response*
HRV	*Heart Rate Variation*
EEG	*Electroencephalogram*
RF	*Random Forest*
kNN	*k-Nearest Neighbors*
SVM	*Support Vector Machine*
NLM	*Non-linear Methods*
NN	*Neural Network*
DNN	*Deep Neural Network*
LR	*Logistic Regression*
SA	*Sensitive Analysis*
DT	*Decision Tree*
STD	*Simple Taylor Decomposition*
PDP	*Partial Dependency Plot*
SP	*Summary Plot*
FI	*Feature Importance*
SM	*Saliency Mask*
AM	*Activation Maximization*
DR	*Decision Rules*
PS	*Prototype Selection*

B Colunas da base de dados utilizada

Coluna	Significado
EDAR_Amphitude	Diferena entre os valores máximo e mínimo de atividade eletrodérmica medidos durante o ciclo de medição
EDAR_Duration	Duração da atividade eletrodérmica medida durante o ciclo de medição
EDAR_Kurtosis	Curtose da atividade eletrodérmica, indica a "cauda"da curva de distribuição, servindo como probabilidade do dado ser real ou ruído
EDAR_Max	Valor máximo medido de atividade eletrodérmica num dado ciclo de medição
EDAR_Mean	Média dos valores medidos de atividade eletrodérmica durante um ciclo de medição
EDAR_Mean-10	Média dos valores de atividade eletrodérmica medida há dez ciclos de medição
EDAR_Mean-9	Média dos valores de atividade eletrodérmica medida há nove ciclos de medição
EDAR_Mean-8	Média dos valores de atividade eletrodérmica medida há oito ciclos de medição
EDAR_Mean-7	Média dos valores de atividade eletrodérmica medida há sete ciclos de medição
EDAR_Mean-6	Média dos valores de atividade eletrodérmica medida há seis ciclos de medição
EDAR_Mean-5	Média dos valores de atividade eletrodérmica medida há cinco ciclos de medição
EDAR_Mean-4	Média dos valores de atividade eletrodérmica medida há quatro ciclos de medição
EDAR_Mean-3	Média dos valores de atividade eletrodérmica medida há três ciclos de medição
EDAR_Mean-2	Média dos valores de atividade eletrodérmica medida há dois ciclos de medição
EDAR_Mean-1	Média dos valores de atividade eletrodérmica medida há um ciclos de medição
EDAR_Min	Valor mínimo medido de atividade eletrodérmica durante um ciclo de medição
EDAR_Std	Desvio padrão de atividade eletrodérmica durante um ciclo de medição
EDAR_Skew	Distorção das medições de atividade eletrodérmica durante um ciclo de medicina
HRR_Max	Valor máximo de frequência cardíaca medido durante um ciclo de medição
HRR_Mean	Média dos valores medidos de frequência cardíaca durante um ciclo de medição
HRR_Mean-10	Média dos valores de frequência cardíaca medida há dez ciclos de medição
HRR_Mean-9	Média dos valores de frequência cardíaca medida há nove ciclos de medição
HRR_Mean-8	Média dos valores de frequência cardíaca medida há oito ciclos de medição

Coluna	Significado
HRR_Mean-7	Média dos valores de frequência cardíaca medida há sete ciclos de medição
HRR_Mean-6	Média dos valores de frequência cardíaca medida há seis ciclos de medição
HRR_Mean-5	Média dos valores de frequência cardíaca medida há cinco ciclos de medição
HRR_Mean-4	Média dos valores de frequência cardíaca medida há quatro ciclos de medição
HRR_Mean-3	Média dos valores de frequência cardíaca medida há três ciclos de medição
HRR_Mean-2	Média dos valores de frequência cardíaca medida há dois ciclos de medição
HRR_Mean-1	Média dos valores de frequência cardíaca medida há um ciclos de medição
HRR_Min	Valor mínimo medido de frequência cardíaca durante um ciclo de medição
HRR_Std	Desvio padrão de frequência cardíaca durante um ciclo de medição
HRR_RMS	Número de ciclos (batimentos) por segundo medidos durante um ciclo de medição
Num_PeaksR	Números de picos da temperatura da pele medidos durante um ciclo de medição
TEMPR_Max	Valor máximo de temperatura da pele medido durante um ciclo de medição
TEMPR_Mean	Média dos valores medidos de temperatura da pele durante um ciclo de medição
TEMP_Mean-10	Média dos valores de temperatura da pele medida há dez ciclos de medição
TEMP_Mean-9	Média dos valores de temperatura da pele medida há nove ciclos de medição
TEMP_Mean-8	Média dos valores de temperatura da pele medida há oito ciclos de medição
TEMP_Mean-7	Média dos valores de temperatura da pele medida há sete ciclos de medição
TEMP_Mean-6	Média dos valores de temperatura da pele medida há seis ciclos de medição
TEMP_Mean-5	Média dos valores de temperatura da pele medida há cinco ciclos de medição
TEMP_Mean-4	Média dos valores de temperatura da pele medida há quatro ciclos de medição
TEMP_Mean-3	Média dos valores de temperatura da pele medida há três ciclos de medição
TEMP_Mean-2	Média dos valores de temperatura da pele medida há dois ciclos de medição
TEMP_Mean-1	Média dos valores de temperatura da pele medida há um ciclos de medição

Coluna	Significado
TEMPR_Min	Valor mínimo medido de temperatura da pele durante um ciclo de medição
TEMPR_Std	Desvio padrão de temperatura da pele durante um ciclo de medição
Stress	Nível de estresse a ser classificado, podendo ser 0 (sem estresse), 1 (pouco estresse) e 2 (muito estresse)

C Programa em Python desenvolvido

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = [10, 10]
import shap

from scipy.stats import kurtosis, skew
from scipy.signal import find_peaks

from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC

from sklearn.preprocessing import MinMaxScaler, OneHotEncoder
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.feature_selection import SelectFromModel
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, f1_score, precision_score,␣
 ↪recall_score, classification_report, roc_auc_score, roc_curve, auc
from sklearn.inspection import (partial_dependence,␣
 ↪PartialDependenceDisplay, permutation_importance)

import random
import ruptures as rpt
```

C.1 Treinando modelos de aprendizado de máquina

C.1.1 Transformar arquivo csv em DataFrame

[2]:
```
df_lag = pd.read_csv("C:/Users/milena/Documents/tcc/dataset/
 ↪combinedlagEDA.csv")
train_set = df_lag.iloc[:,0:48]
labels = df_lag.iloc[:,48:49]

train, test, train_labels, test_labels = train_test_split(train_set,␣
 ↪labels, test_size=0.3, random_state=30)
```

C.1.2 *k-Nearest Nighbors*

[3]:
```
k_nearest_neighbors = KNeighborsClassifier(n_neighbors=5,␣
↪metric="minkowski", weights="uniform")

k_nearest_neighbors.fit(train, train_labels.values.ravel())

y_pred_knn = k_nearest_neighbors.predict(test)
```

[4]:
```
f1score   = f1_score        (test_labels, y_pred_knn, average = 'macro')
recall    = recall_score    (test_labels, y_pred_knn, average = 'macro')
accuracy  = accuracy_score  (test_labels, y_pred_knn)
AUC       = roc_auc_score   (test_labels.values.ravel(),␣
↪k_nearest_neighbors.predict_proba(test), multi_class='ovr')

print('auc =', AUC)
print('acc =', accuracy)
print('f1 =', f1score)
print('recall =', recall)
```

```
auc = 0.9638160217720122
acc = 0.8778789501874665
f1 = 0.8544328715993337
recall = 0.8540278142357861
```

C.1.3 *Logistic Regression*

[5]:
```
logistic_regression = LogisticRegression(penalty="l2", C=1.0,␣
↪max_iter=999)

logistic_regression.fit(train, train_labels.values.ravel())

y_pred_lr = logistic_regression.predict(test)
```

[6]:
```
f1score   = f1_score        (test_labels, y_pred_lr, average = 'macro')
recall    = recall_score    (test_labels, y_pred_lr, average = 'macro')
accuracy  = accuracy_score  (test_labels, y_pred_lr)
AUC       = roc_auc_score   (test_labels.values.ravel(),␣
↪logistic_regression.predict_proba(test), multi_class='ovr')

print('auc =', AUC)
print('acc =', accuracy)
print('f1 =', f1score)
print('recall =', recall)
```

```
auc = 0.7472221544626391
acc = 0.610337439742903
```

```
f1 = 0.4534351582466029
recall = 0.4965674019596338
```

C.1.4 *Neural Network*

[7]:
```
neural_network = MLPClassifier(hidden_layer_sizes=(100,),␣
↪activation="relu", solver="adam", max_iter=999)

neural_network.fit(train, train_labels.values.ravel())

y_pred_nn = neural_network.predict(test)
```

[8]:
```
f1score   = f1_score        (test_labels, y_pred_nn, average = 'macro')
recall    = recall_score    (test_labels, y_pred_nn, average = 'macro')
accuracy  = accuracy_score  (test_labels, y_pred_nn)
AUC       = roc_auc_score   (test_labels.values.ravel(), neural_network.
↪predict_proba(test), multi_class='ovr')

print('auc =', AUC)
print('acc =', accuracy)
print('f1 =', f1score)
print('recall =', recall)
```

```
auc = 0.9009343586468708
acc = 0.7715586502410284
f1 = 0.7021793902746284
recall = 0.6915208140374115
```

C.1.5 *Random Forest*

[9]:
```
random_forest = RandomForestClassifier(n_estimators=100,max_depth=15)

random_forest.fit(train, train_labels.values.ravel())

y_pred_rf = random_forest.predict(test)
```

[10]:
```
f1score   = f1_score        (test_labels, y_pred_rf, average = 'macro')
recall    = recall_score    (test_labels, y_pred_rf, average = 'macro')
accuracy  = accuracy_score  (test_labels, y_pred_rf)
AUC       = roc_auc_score   (test_labels.values.ravel(), random_forest.
↪predict_proba(test), multi_class='ovr')

print('auc =', AUC)
print('acc =', accuracy)
print('f1 =', f1score)
print('recall =', recall)
```

```
auc = 0.9941924111259378
acc = 0.9550080342795929
f1 = 0.9427457550016455
recall = 0.9333256807971191
```

C.1.6 *Support Vector Machine*

[11]:
```
support_vector_machine = SVC(C=1.0, gamma=0.1, kernel="linear", tol=0.
↪001, max_iter=9999, probability=True)

support_vector_machine.fit(train, train_labels.values.ravel())

y_pred_svm = support_vector_machine.predict(test)
```

[12]:
```
f1score   = f1_score        (test_labels, y_pred_svm, average = 'macro')
recall    = recall_score    (test_labels, y_pred_svm, average = 'macro')
accuracy  = accuracy_score  (test_labels, y_pred_svm)
AUC       = roc_auc_score   (test_labels.values.ravel(),␣
↪support_vector_machine.predict_proba(test), multi_class='ovr')

print('auc =', AUC)
print('acc =', accuracy)
print('f1 =', f1score)
print('recall =', recall)
```

```
auc = 0.7352568615079594
acc = 0.6170326727370112
f1 = 0.45116522976196965
recall = 0.5001786228997965
```

C.1.7 Fazer curva de ROC

Calcula probabilidade de predição das três saídas possíveis

[13]:
```
k_nearest_neighbors_pred_prob_0 = k_nearest_neighbors.
↪predict_proba(test)[:,0]
logistic_regression_pred_prob_0 = logistic_regression.
↪predict_proba(test)[:,0]
neural_network_pred_prob_0 = neural_network.predict_proba(test)[:,0]
random_forest_pred_prob_0 = random_forest.predict_proba(test)[:,0]
support_vector_machine_pred_prob_0 = support_vector_machine.
↪predict_proba(test)[:,0]
```

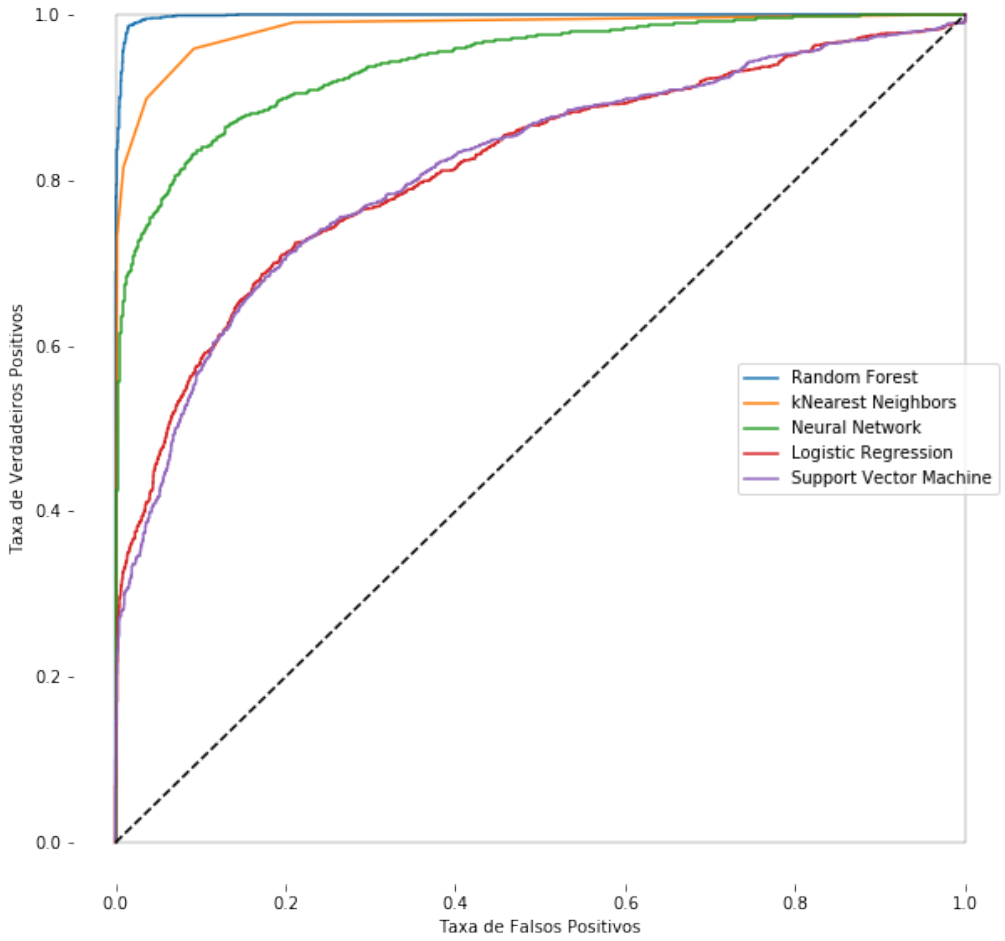
[14]:
```
k_nearest_neighbors_pred_prob_1 = k_nearest_neighbors.
↪predict_proba(test)[:,1]
logistic_regression_pred_prob_1 = logistic_regression.
↪predict_proba(test)[:,1]
neural_network_pred_prob_1 = neural_network.predict_proba(test)[:,1]
```

```
random_forest_pred_prob_1 = random_forest.predict_proba(test)[:,1]
support_vector_machine_pred_prob_1 = support_vector_machine.
 ↪predict_proba(test)[:,1]
```

[15]:
```
k_nearest_neighbors_pred_prob_2 = k_nearest_neighbors.
 ↪predict_proba(test)[:,2]
logistic_regression_pred_prob_2 = logistic_regression.
 ↪predict_proba(test)[:,2]
neural_network_pred_prob_2 = neural_network.predict_proba(test)[:,2]
random_forest_pred_prob_2 = random_forest.predict_proba(test)[:,2]
support_vector_machine_pred_prob_2 = support_vector_machine.
 ↪predict_proba(test)[:,2]
```

Saída 0 como positiva

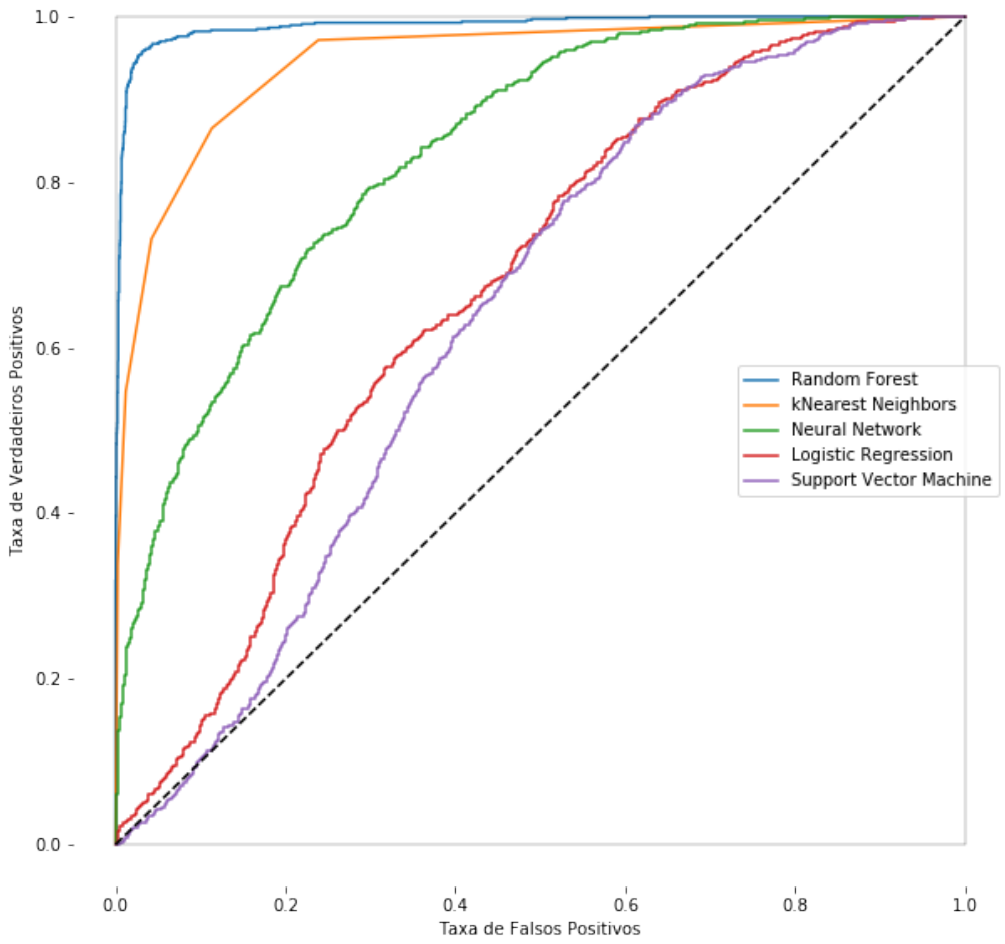
[16]:
```
fpr, tpr, threshold = roc_curve(test_labels.values.ravel(),␣
 ↪random_forest_pred_prob_0, pos_label=0)
plt.plot(fpr, tpr, label='Random Forest')

fpr, tpr, threshold = roc_curve(test_labels.values.ravel(),␣
 ↪k_nearest_neighbors_pred_prob_0, pos_label=0)
plt.plot(fpr, tpr, label='kNearest Neighbors')

fpr, tpr, threshold = roc_curve(test_labels.values.ravel(),␣
 ↪neural_network_pred_prob_0, pos_label=0)
plt.plot(fpr, tpr, label='Neural Network')

fpr, tpr, threshold = roc_curve(test_labels.values.ravel(),␣
 ↪logistic_regression_pred_prob_0, pos_label=0)
plt.plot(fpr, tpr, label='Logistic Regression')

fpr, tpr, threshold = roc_curve(test_labels.values.ravel(),␣
 ↪support_vector_machine_pred_prob_0, pos_label=0)
plt.plot(fpr, tpr, label='Support Vector Machine')

plt.plot([0,1], [0,1], 'k--')
plt.plot([0,0], [0,1], color="black", linewidth=0.25)
plt.plot([0,1], [1,1], color="black", linewidth=0.25)
plt.plot([0,1], [0,0], color="black", linewidth=0.25)
plt.plot([1,1], [0,1], color="black", linewidth=0.25)
plt.xlabel('Taxa de Falsos Positivos')
plt.ylabel('Taxa de Verdadeiros Positivos')
plt.legend(prop={'size': 10})

for pos in ['right', 'top', 'bottom', 'left']:
   plt.gca().spines[pos].set_visible(False)

plt.show()
```



Saída 1 como positiva

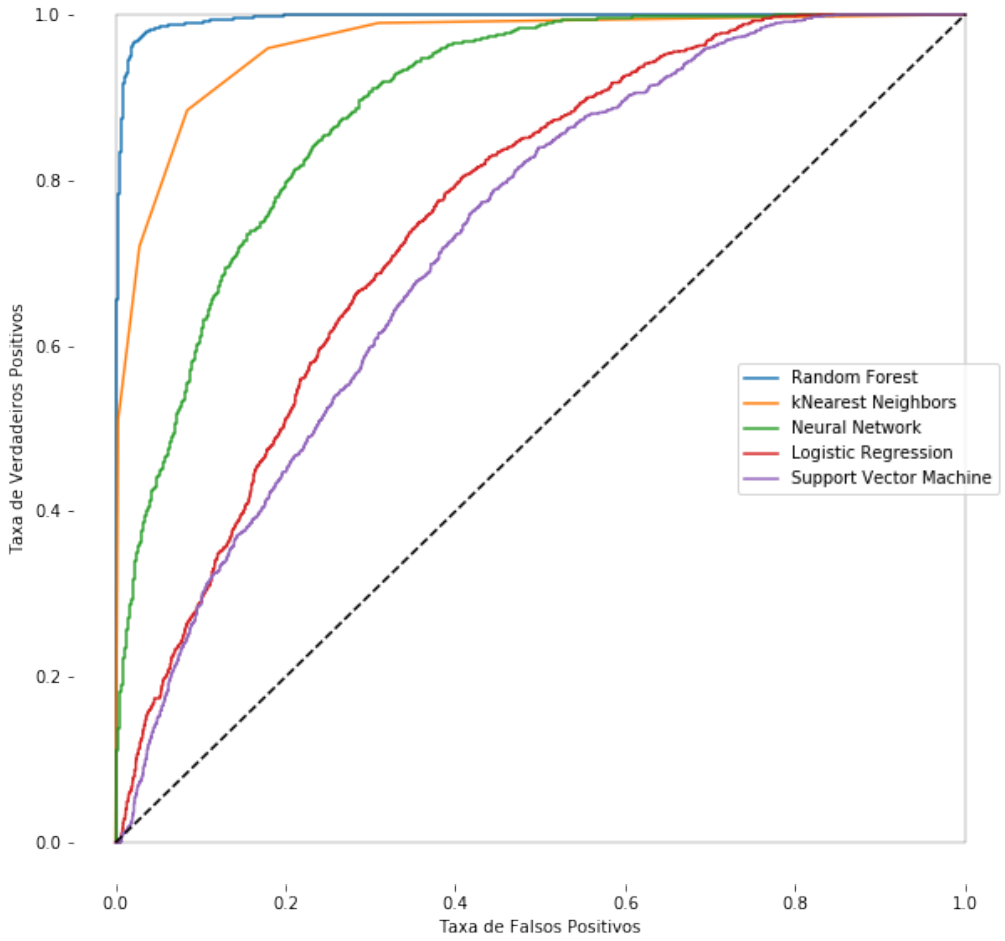
[17]:
```
fpr, tpr, threshold = roc_curve(test_labels.values.ravel(),␣
 ↪random_forest_pred_prob_1, pos_label=1)
plt.plot(fpr, tpr, label='Random Forest')

fpr, tpr, threshold = roc_curve(test_labels.values.ravel(),␣
 ↪k_nearest_neighbors_pred_prob_1, pos_label=1)
plt.plot(fpr, tpr, label='kNearest Neighbors')

fpr, tpr, threshold = roc_curve(test_labels.values.ravel(),␣
 ↪neural_network_pred_prob_1, pos_label=1)
plt.plot(fpr, tpr, label='Neural Network')

fpr, tpr, threshold = roc_curve(test_labels.values.ravel(),␣
 ↪logistic_regression_pred_prob_1, pos_label=1)
plt.plot(fpr, tpr, label='Logistic Regression')
```

```
fpr, tpr, threshold = roc_curve(test_labels.values.ravel(),␣
 ↪support_vector_machine_pred_prob_1, pos_label=1)
plt.plot(fpr, tpr, label='Support Vector Machine')

plt.plot([0,1], [0,1], 'k--')
plt.plot([0,0], [0,1], color="black", linewidth=0.25)
plt.plot([0,1], [1,1], color="black", linewidth=0.25)
plt.plot([0,1], [0,0], color="black", linewidth=0.25)
plt.plot([1,1], [0,1], color="black", linewidth=0.25)
plt.xlabel('Taxa de Falsos Positivos')
plt.ylabel('Taxa de Verdadeiros Positivos')
plt.legend(prop={'size': 10})

for pos in ['right', 'top', 'bottom', 'left']:
   plt.gca().spines[pos].set_visible(False)

plt.show()
```



Saída 2 como positiva

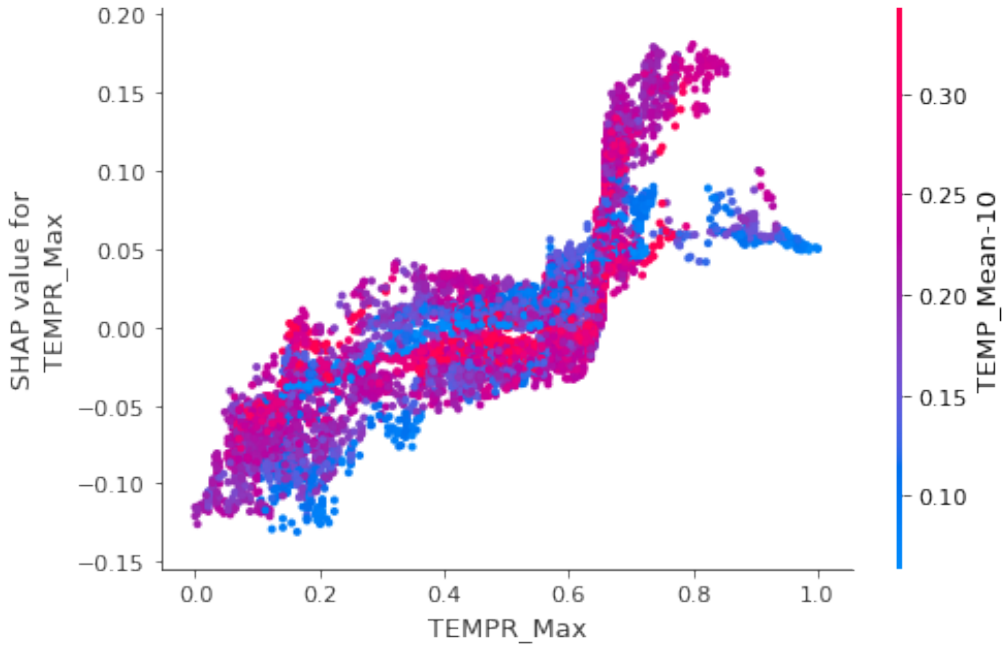
[18]:
```
fpr, tpr, threshold = roc_curve(test_labels.values.ravel(),␣
↪random_forest_pred_prob_2, pos_label=2)
plt.plot(fpr, tpr, label='Random Forest')

fpr, tpr, threshold = roc_curve(test_labels.values.ravel(),␣
↪k_nearest_neighbors_pred_prob_2, pos_label=2)
plt.plot(fpr, tpr, label='kNearest Neighbors')

fpr, tpr, threshold = roc_curve(test_labels.values.ravel(),␣
↪neural_network_pred_prob_2, pos_label=2)
plt.plot(fpr, tpr, label='Neural Network')

fpr, tpr, threshold = roc_curve(test_labels.values.ravel(),␣
↪logistic_regression_pred_prob_2, pos_label=2)
plt.plot(fpr, tpr, label='Logistic Regression')

fpr, tpr, threshold = roc_curve(test_labels.values.ravel(),␣
↪support_vector_machine_pred_prob_1, pos_label=2)
plt.plot(fpr, tpr, label='Support Vector Machine')

plt.plot([0,1], [0,1], 'k--')
plt.plot([0,0], [0,1], color="black", linewidth=0.25)
plt.plot([0,1], [1,1], color="black", linewidth=0.25)
plt.plot([0,1], [0,0], color="black", linewidth=0.25)
plt.plot([1,1], [0,1], color="black", linewidth=0.25)
plt.xlabel('Taxa de Falsos Positivos')
plt.ylabel('Taxa de Verdadeiros Positivos')
plt.legend(prop={'size': 10})

for pos in ['right', 'top', 'bottom', 'left']:
   plt.gca().spines[pos].set_visible(False)

plt.show()
```



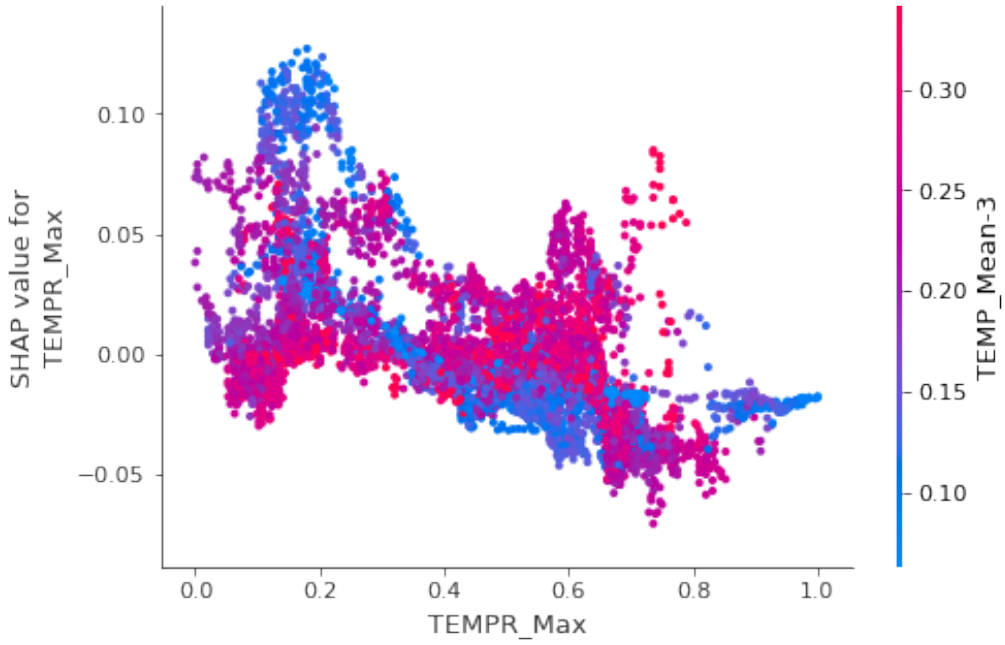
C.2 Explicando melhor modelo de Aprendizado de máquina (*Random Forest*)

[19]:
```
explainer = shap.TreeExplainer(random_forest)
shap_values = explainer.shap_values(train)
shap_obj = explainer(train)
```

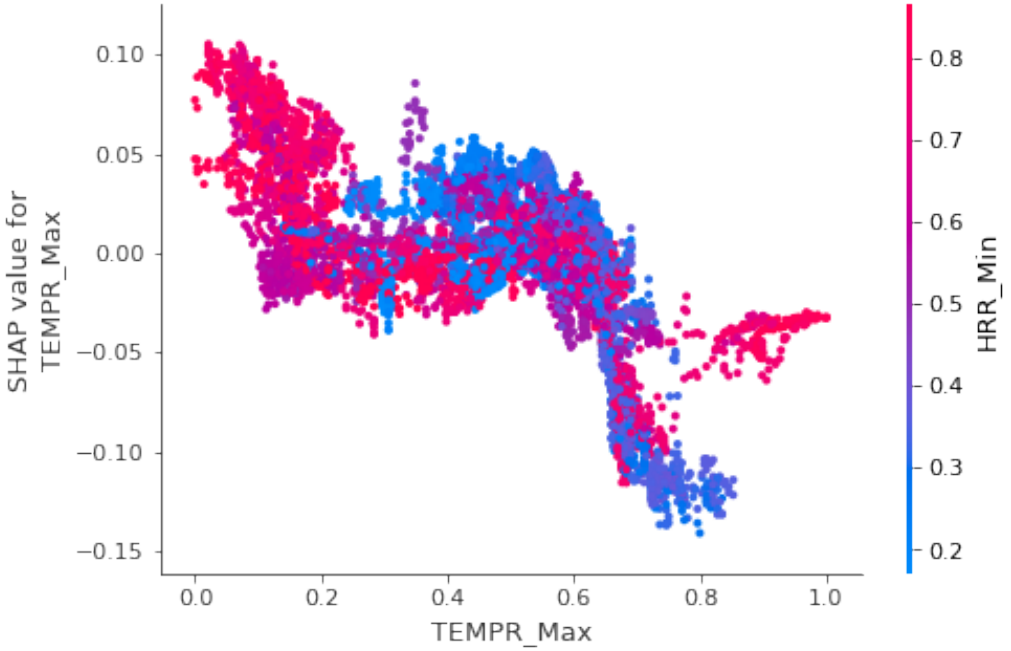
[20]:
```
shap.dependence_plot("TEMPR_Max", shap_values[0], train.values,␣
↪feature_names=train.columns)
```



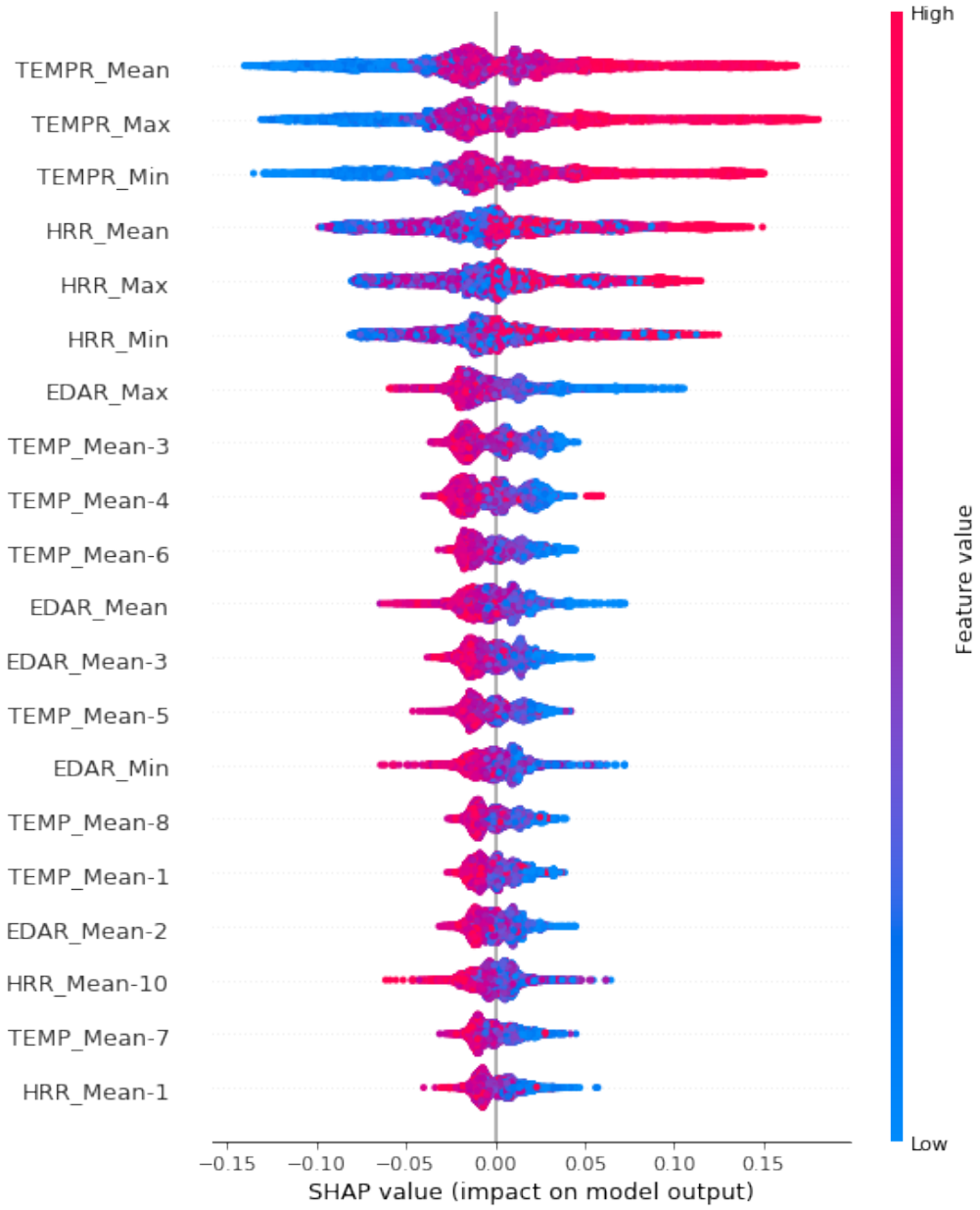
[21]:
```
shap.dependence_plot("TEMPR_Max", shap_values[1], train.values,␣
↪feature_names=train.columns)
```



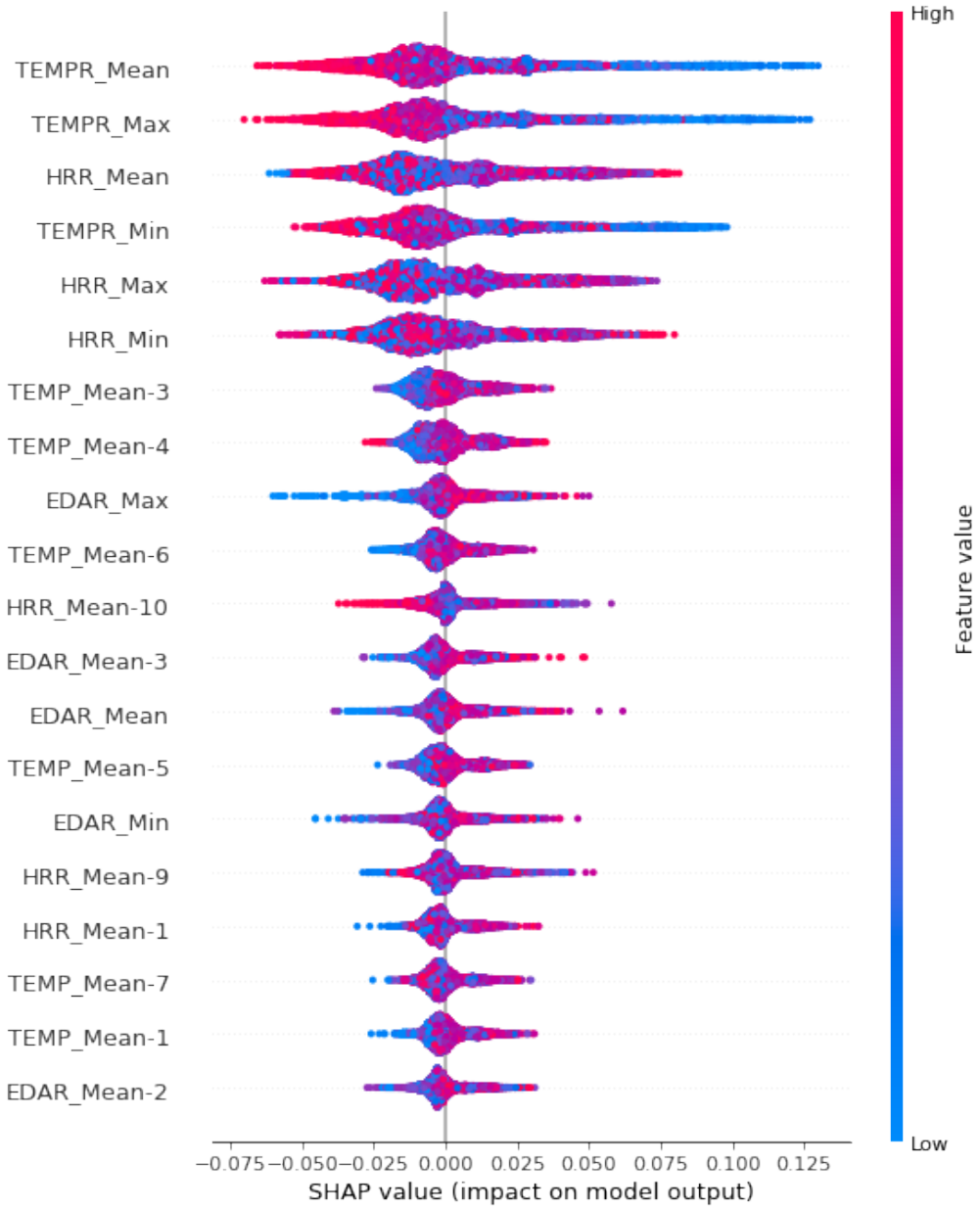
[22]:
```
shap.dependence_plot("TEMPR_Max", shap_values[2], train.values,␣
↪feature_names=train.columns)
```



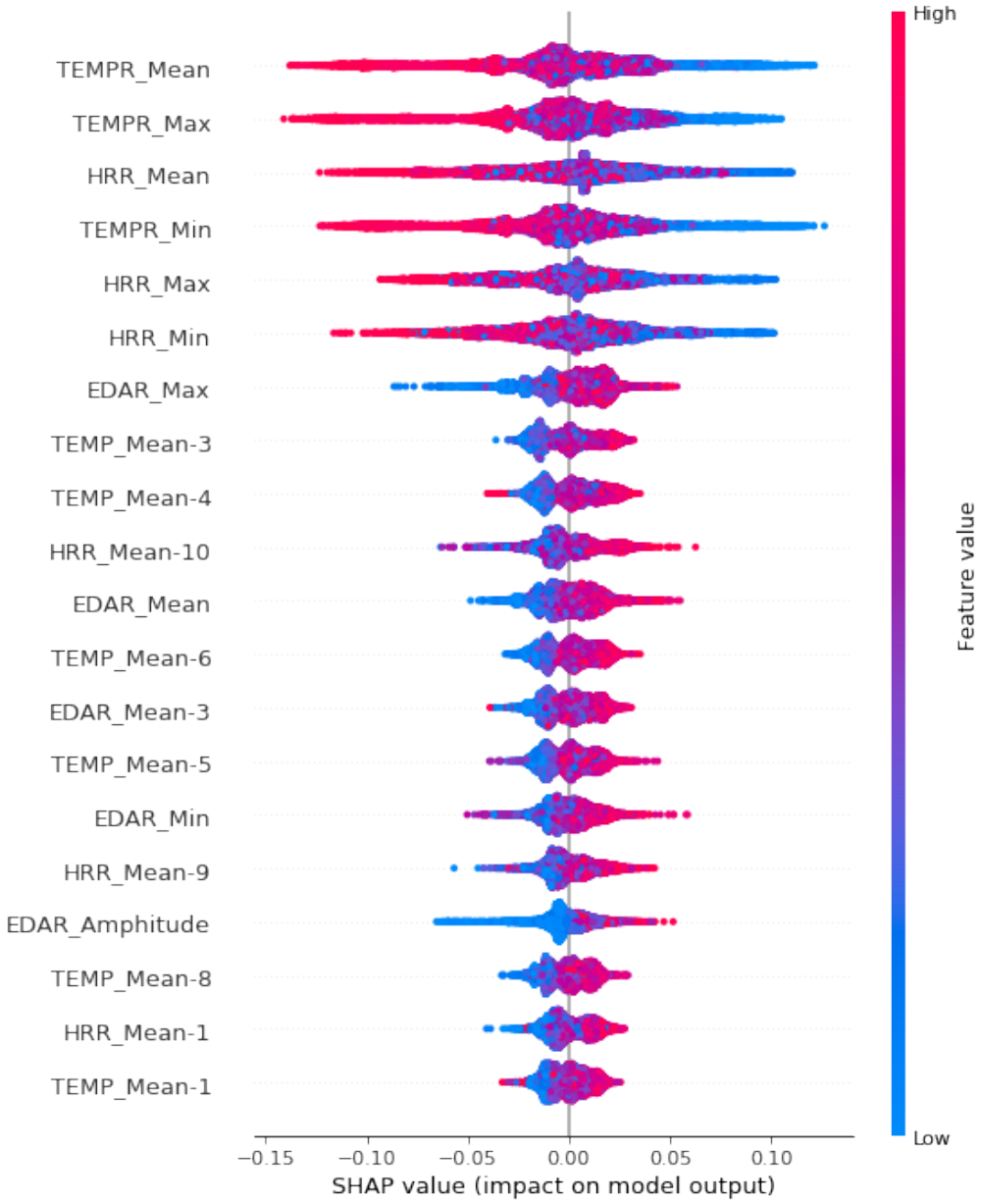
[23]:
```
shap.summary_plot(shap_values[0], train.values, feature_names=train.
↪columns)
```



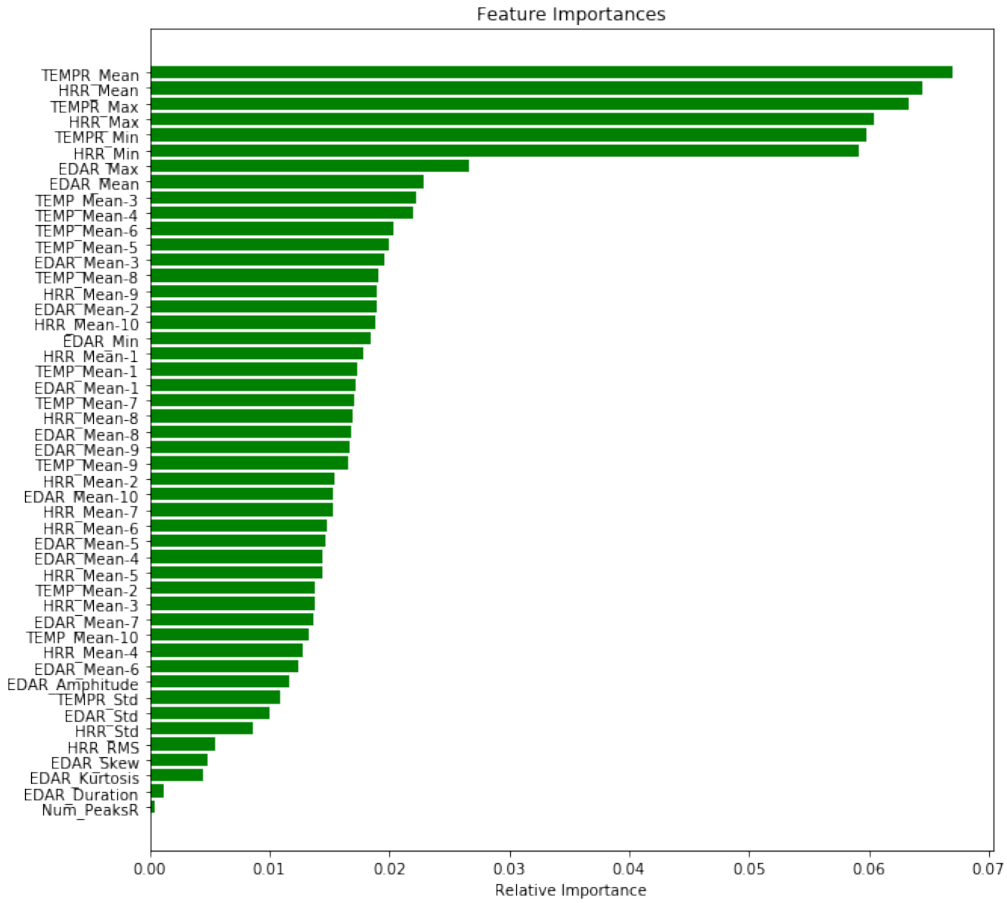
[24]:
```
shap.summary_plot(shap_values[1], train.values, feature_names=train.
↪columns)
```



[25]:
```
shap.summary_plot(shap_values[2], train.values, feature_names=train.
↪columns)
```



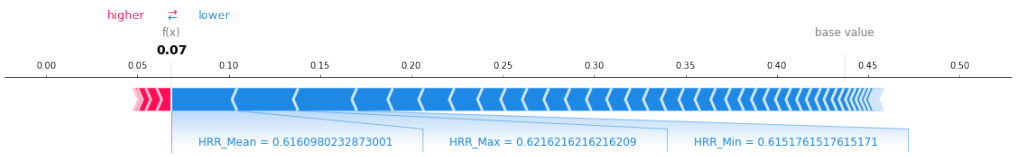
[26]:
```
importances = random_forest.feature_importances_
indices = np.argsort(importances)
features = df_lag.columns
plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], color='g',␣
 ↪align='center')
plt.yticks(range(len(indices)), [features[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```



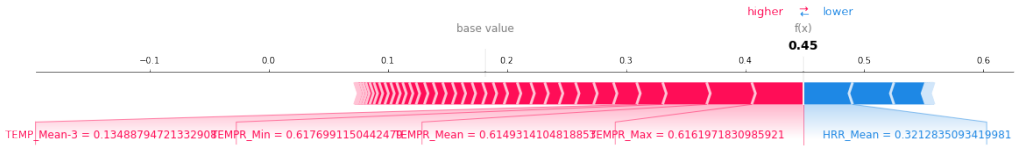
[27]:
```
explainer.expected_value
```

[27]:
```
array([0.43726208, 0.18178854, 0.38094937])
```

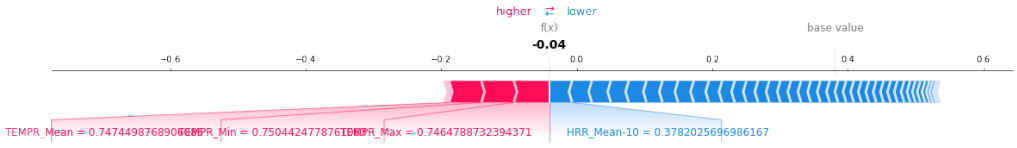
[28]:
```
i=180
shap.force_plot(explainer.expected_value[0], shap_values[0][i], train.
↪values[i], feature_names = train.columns, matplotlib=1)
```



[29]:
```
j=1985
shap.force_plot(explainer.expected_value[1], shap_values[0][j], train.
 ↪values[j], feature_names = train.columns, matplotlib=1)
```



[30]:
```
i=3567
shap.force_plot(explainer.expected_value[2], shap_values[0][i], train.
 ↪values[i], feature_names = train.columns, matplotlib=1)
```

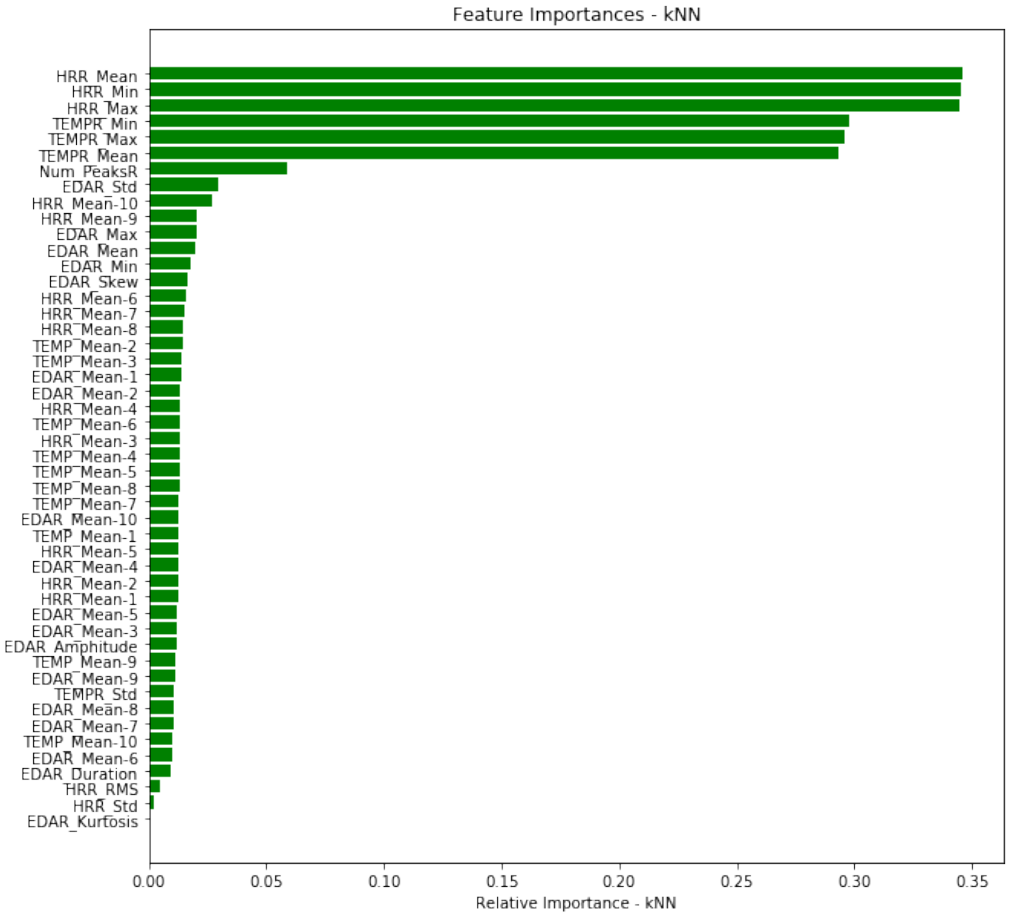


C.3 Treinando novamente com top 10 características

C.3.1 Pegar top 10 caracteísticas dos top 3 modelos

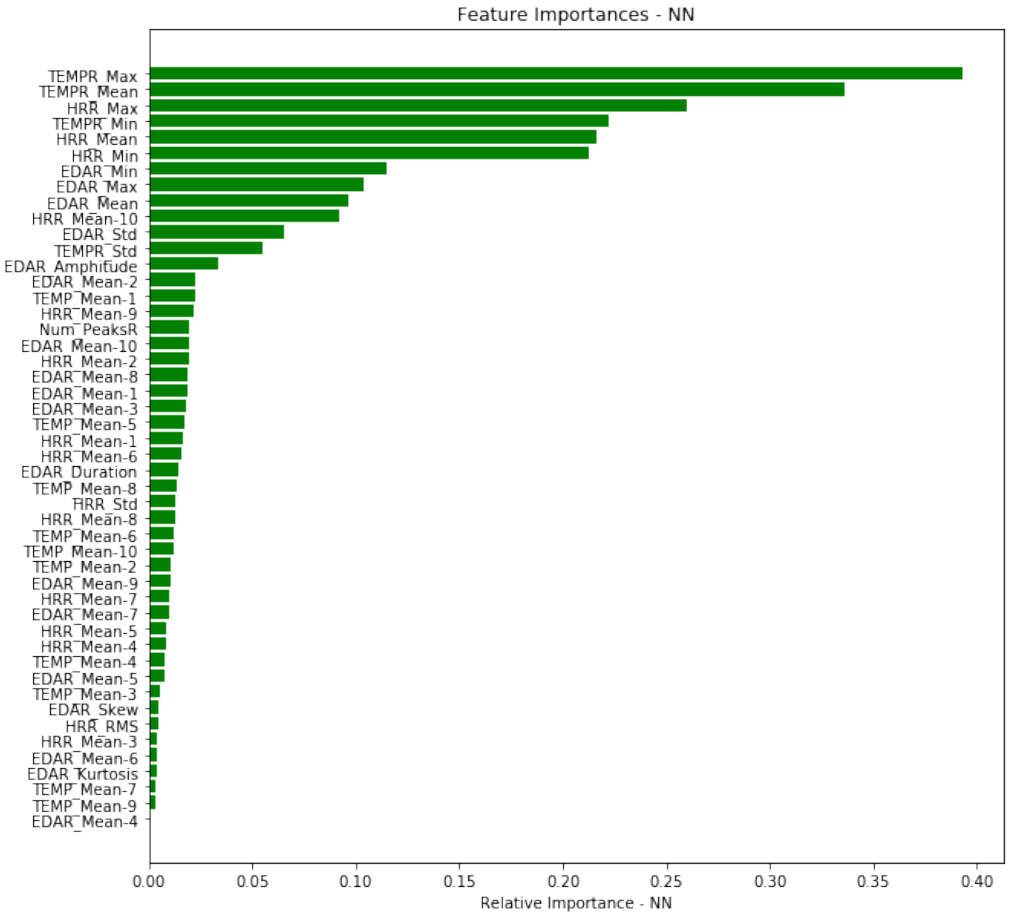
k-Nearest Neighbors

[31]:
```
results_knn = permutation_importance(k_nearest_neighbors, train,␣
 ↪train_labels, scoring='neg_mean_squared_error')
importances_knn = results_knn.importances_mean
plt.title('Feature Importances - kNN')
indices_knn = np.argsort(importances_knn)
plt.barh(range(len(indices_knn)), importances_knn[indices_knn],␣
 ↪color='g', align='center')
plt.yticks(range(len(indices_knn)), [features[i] for i in indices_knn])
plt.xlabel('Relative Importance - kNN')
plt.show()
```



Neural Network

[32]:
```
results_nn = permutation_importance(neural_network, train,␣
↪train_labels, scoring='neg_mean_squared_error')
importances_nn = results_nn.importances_mean
plt.title('Feature Importances - NN')
indices_nn = np.argsort(importances_nn)
plt.barh(range(len(indices_nn)), importances_nn[indices_nn], color='g',␣
↪align='center')
plt.yticks(range(len(indices_nn)), [features[i] for i in indices_nn])
plt.xlabel('Relative Importance - NN')
plt.show()
```



C.3.2 Treinar todos os modelos com top 10 características

Top 10 características do *Random Forest*

[33]:
```
dh_top10_RF = pd.concat([df_lag["TEMPR_Mean"], df_lag["HRR_Mean"],␣
↪df_lag["TEMPR_Max"], df_lag["HRR_Max"], df_lag["TEMPR_Min"],
                         df_lag["HRR_Min"], df_lag["EDAR_Max"],␣
↪df_lag["EDAR_Mean"], df_lag["TEMP_Mean-3"], df_lag["TEMP_Mean-4"],
                        df_lag["Stress"]], axis=1)
dh_top10_RF.head()
```

[33]:

	TEMPR_Mean	HRR_Mean	TEMPR_Max	HRR_Max	TEMPR_Min	HRR_Min	EDAR_Max
0	0.821491	0.641552	0.823944	0.643243	0.821239	0.639566	0.107022
1	0.827471	0.642973	0.830986	0.643243	0.828319	0.639566	0.103630
2	0.832395	0.643921	0.834507	0.645946	0.831858	0.640921	0.101679

```
3    0.837759  0.645952   0.839789  0.645946   0.842478  0.644986  0.
↪099644
4    0.843123  0.646764   0.848592  0.648649   0.846018  0.644986  0.
↪098796
```

	EDAR_Mean	TEMP_Mean-3	TEMP_Mean-4	Stress
0	0.105191	0.112588	0.108121	0.0
1	0.102822	0.117063	0.112588	0.0
2	0.101157	0.119653	0.117063	0.0
3	0.099952	0.119543	0.119653	0.0
4	0.099298	0.119534	0.119543	0.0

[34]:
```
train_set_top10RF = dh_top10_RF.iloc[:,0:10]
labels_top10RF = dh_top10_RF.iloc[:,10:11]

train_top10RF, test_top10RF, train_labels_top10RF, test_labels_top10RF␣
↪= train_test_split(train_set_top10RF, labels_top10RF, test_size=0.3,␣
↪random_state=30)
```

[35]:
```
k_nearest_neighbors_top10RF = KNeighborsClassifier(n_neighbors=5,␣
↪metric="minkowski", weights="uniform")

k_nearest_neighbors_top10RF.fit(train_top10RF, train_labels_top10RF.
↪values.ravel())

y_pred_knn_top10RF = k_nearest_neighbors_top10RF.predict(test_top10RF)
```

[36]:
```
f1score   = f1_score        (test_labels_top10RF, y_pred_knn_top10RF,␣
↪average = 'macro')
recall    = recall_score    (test_labels_top10RF, y_pred_knn_top10RF,␣
↪average = 'macro')
accuracy  = accuracy_score  (test_labels_top10RF, y_pred_knn_top10RF)
AUC       = roc_auc_score   (test_labels_top10RF.values.ravel(),␣
↪k_nearest_neighbors_top10RF.predict_proba(test_top10RF),␣
↪multi_class='ovr')

print("KNN with RF's top 10 features:")
print('auc =', AUC)
print('acc =', accuracy)
print('f1 =', f1score)
print('recall =', recall)
```

```
KNN with RF's top 10 features:
auc = 0.990360069738936
acc = 0.9429566148901982
f1 = 0.9293324049778787
recall = 0.9283986249030205
```

[37]:
```
logistic_regression_top10RF = LogisticRegression(penalty="l2", C=1.0,␣
↪max_iter=999)

logistic_regression_top10RF.fit(train_top10RF, train_labels_top10RF.
↪values.ravel())

y_pred_lr_top10RF = logistic_regression_top10RF.predict(test_top10RF)
```

[38]:
```
f1score   = f1_score        (test_labels_top10RF, y_pred_lr_top10RF,␣
↪average = 'macro')
recall    = recall_score    (test_labels_top10RF, y_pred_lr_top10RF,␣
↪average = 'macro')
accuracy  = accuracy_score  (test_labels_top10RF, y_pred_lr_top10RF)
AUC       = roc_auc_score   (test_labels_top10RF.values.ravel(),␣
↪logistic_regression_top10RF.predict_proba(test_top10RF),␣
↪multi_class='ovr')

print("LR with RF's top 10 features:")
print('auc =', AUC)
print('acc =', accuracy)
print('f1 =', f1score)
print('recall =', recall)
```

```
LR with RF's top 10 features:
auc = 0.7365922072218525
acc = 0.6041778253883235
f1 = 0.44105466653636904
recall = 0.4890237789960484
```

[39]:
```
neural_network_top10RF = MLPClassifier(hidden_layer_sizes=(100,),␣
↪activation="relu", solver="adam", max_iter=999)

neural_network_top10RF.fit(train_top10RF, train_labels_top10RF.values.
↪ravel())

y_pred_nn_top10RF = neural_network_top10RF.predict(test_top10RF)
```

[40]:
```
f1score   = f1_score        (test_labels_top10RF, y_pred_nn_top10RF,␣
↪average = 'macro')
recall    = recall_score    (test_labels_top10RF, y_pred_nn_top10RF,␣
↪average = 'macro')
accuracy  = accuracy_score  (test_labels_top10RF, y_pred_nn_top10RF)
AUC       = roc_auc_score   (test_labels_top10RF.values.ravel(),␣
↪neural_network_top10RF.predict_proba(test_top10RF), multi_class='ovr')

print("NN with RF's top 10 features:")
print('auc =', AUC)
```

```
print('acc =', accuracy)
print('f1 =', f1score)
print('recall =', recall)
```

```
NN with RF's top 10 features:
auc = 0.8702960502069351
acc = 0.7337975361542581
f1 = 0.6614761499197628
recall = 0.6556414826719518
```

[41]:
```
random_forest_top10RF =␣
↪RandomForestClassifier(n_estimators=100,max_depth=15)

random_forest_top10RF.fit(train_top10RF, train_labels_top10RF.values.
↪ravel())

y_pred_rf_top10RF = random_forest_top10RF.predict(test_top10RF)
```

[42]:
```
f1score   = f1_score        (test_labels_top10RF, y_pred_rf_top10RF,␣
↪average = 'macro')
recall    = recall_score    (test_labels_top10RF, y_pred_rf_top10RF,␣
↪average = 'macro')
accuracy  = accuracy_score  (test_labels_top10RF, y_pred_rf_top10RF)
AUC       = roc_auc_score   (test_labels_top10RF.values.ravel(),␣
↪random_forest_top10RF.predict_proba(test_top10RF), multi_class='ovr')

print("RF with RF's top 10 features")
print('auc =', AUC)
print('acc =', accuracy)
print('f1 =', f1score)
print('recall =', recall)
```

```
RF with RF's top 10 features
auc = 0.9925303643656429
acc = 0.9475093733261918
f1 = 0.9342790342915813
recall = 0.9276126233459759
```

[43]:
```
support_vector_machine_top10RF = SVC(C=1.0, gamma=0.1, kernel="linear",␣
↪tol=0.001, max_iter=9999, probability=True)

support_vector_machine_top10RF.fit(train_top10RF, train_labels_top10RF.
↪values.ravel())

y_pred_svm_top10RF = support_vector_machine_top10RF.
↪predict(test_top10RF)
```

[44]:
```
f1score   = f1_score        (test_labels_top10RF, y_pred_svm_top10RF,␣
 ↪average = 'macro')
recall    = recall_score    (test_labels_top10RF, y_pred_svm_top10RF,␣
 ↪average = 'macro')
accuracy  = accuracy_score  (test_labels_top10RF, y_pred_svm_top10RF)
AUC       = roc_auc_score   (test_labels_top10RF.values.ravel(),␣
 ↪support_vector_machine_top10RF.predict_proba(test_top10RF),␣
 ↪multi_class='ovr')

print("SVM with RF's top 10 features")
print('auc =', AUC)
print('acc =', accuracy)
print('f1 =', f1score)
print('recall =', recall)
```

```
SVM with RF's top 10 features
auc = 0.7353231147017336
acc = 0.613551151580075
f1 = 0.44897370261969427
recall = 0.49764552867692685
```

Top 10 características do *k-Nearest Neighbor*

[45]:
```
dh_top10_kNN = pd.concat([df_lag["HRR_Mean"], df_lag["HRR_Min"],␣
 ↪df_lag["HRR_Max"], df_lag["TEMPR_Min"], df_lag["TEMPR_Max"],
                           df_lag["TEMPR_Mean"], df_lag["Num_PeaksR"],␣
 ↪df_lag["EDAR_Std"], df_lag["TEMP_Mean-10"], df_lag["TEMP_Mean-9"],
                           df_lag["Stress"]], axis=1)
dh_top10_kNN.head()
```

[45]:

	HRR_Mean	HRR_Min	HRR_Max	TEMPR_Min	TEMPR_Max	TEMPR_Mean	Num_PeaksR
0	0.641552	0.639566	0.643243	0.821239	0.823944	0.821491	0.0
1	0.642973	0.639566	0.643243	0.828319	0.830986	0.827471	0.0
2	0.643921	0.640921	0.645946	0.831858	0.834507	0.832395	0.0
3	0.645952	0.644986	0.645946	0.842478	0.839789	0.837759	0.0
4	0.646764	0.644986	0.648649	0.846018	0.848592	0.843123	0.0

	EDAR_Std	TEMP_Mean-10	TEMP_Mean-9	Stress
0	0.035656	0.084598	0.089846	0.0
1	0.023788	0.089846	0.094890	0.0
2	0.018717	0.094890	0.098347	0.0
3	0.011283	0.098347	0.101107	0.0

```
4  0.005735       0.101107       0.104240       0.0
```

[46]:
```
train_set_top10kNN = dh_top10_kNN.iloc[:,0:10]
labels_top10kNN = dh_top10_kNN.iloc[:,10:11]

train_top10kNN, test_top10kNN, train_labels_top10kNN,␣
↪test_labels_top10kNN = train_test_split(train_set_top10kNN,␣
↪labels_top10kNN, test_size=0.3, random_state=30)
```

[47]:
```
k_nearest_neighbors_top10kNN = KNeighborsClassifier(n_neighbors=5,␣
↪metric="minkowski", weights="uniform")

k_nearest_neighbors_top10kNN.fit(train_top10kNN, train_labels_top10kNN.
↪values.ravel())

y_pred_knn_top10kNN = k_nearest_neighbors_top10kNN.
↪predict(test_top10kNN)
```

[48]:
```
f1score   = f1_score        (test_labels_top10kNN, y_pred_knn_top10kNN,␣
↪average = 'macro')
recall    = recall_score    (test_labels_top10kNN, y_pred_knn_top10kNN,␣
↪average = 'macro')
accuracy  = accuracy_score  (test_labels_top10kNN, y_pred_knn_top10kNN)
AUC       = roc_auc_score   (test_labels_top10kNN.values.ravel(),␣
↪k_nearest_neighbors_top10kNN.predict_proba(test_top10kNN),␣
↪multi_class='ovr')

print("KNN with kNN's top 10 features:")
print('auc =', AUC)
print('acc =', accuracy)
print('f1 =', f1score)
print('recall =', recall)
```

```
KNN with kNN's top 10 features:
auc = 0.9606830298054806
acc = 0.8693090519550081
f1 = 0.8466160611574279
recall = 0.8450254258112829
```

[49]:
```
logistic_regression_top10kNN = LogisticRegression(penalty="l2", C=1.0,␣
↪max_iter=999)

logistic_regression_top10kNN.fit(train_top10kNN, train_labels_top10kNN.
↪values.ravel())

y_pred_lr_top10kNN = logistic_regression_top10kNN.predict(test_top10kNN)
```

[50]:
```
f1score   = f1_score        (test_labels_top10kNN, y_pred_lr_top10kNN,␣
 ↪average = 'macro')
recall    = recall_score    (test_labels_top10kNN, y_pred_lr_top10kNN,␣
 ↪average = 'macro')
accuracy  = accuracy_score  (test_labels_top10kNN, y_pred_lr_top10kNN)
AUC       = roc_auc_score   (test_labels_top10kNN.values.ravel(),␣
 ↪logistic_regression_top10kNN.predict_proba(test_top10kNN),␣
 ↪multi_class='ovr')

print("LR with kNN's top 10 features:")
print('auc =', AUC)
print('acc =', accuracy)
print('f1 =', f1score)
print('recall =', recall)
```

```
LR with kNN's top 10 features:
auc = 0.7288210797605883
acc = 0.5915907873594001
f1 = 0.4350843591070494
recall = 0.4796565130977762
```

[51]:
```
neural_network_top10kNN = MLPClassifier(hidden_layer_sizes=(100,),␣
 ↪activation="relu", solver="adam", max_iter=999)

neural_network_top10kNN.fit(train_top10kNN, train_labels_top10kNN.
 ↪values.ravel())

y_pred_nn_top10kNN = neural_network_top10kNN.predict(test_top10kNN)
```

[52]:
```
f1score   = f1_score        (test_labels_top10kNN, y_pred_nn_top10kNN,␣
 ↪average = 'macro')
recall    = recall_score    (test_labels_top10kNN, y_pred_nn_top10kNN,␣
 ↪average = 'macro')
accuracy  = accuracy_score  (test_labels_top10kNN, y_pred_nn_top10kNN)
AUC       = roc_auc_score   (test_labels_top10kNN.values.ravel(),␣
 ↪neural_network_top10kNN.predict_proba(test_top10kNN),␣
 ↪multi_class='ovr')

print("NN with kNN's top 10 features:")
print('auc =', AUC)
print('acc =', accuracy)
print('f1 =', f1score)
print('recall =', recall)
```

```
NN with kNN's top 10 features:
auc = 0.8410757762935264
acc = 0.7129084092126406
```

```
f1 = 0.6406334279129124
recall = 0.6338855714948552
```

[53]:
```
random_forest_top10kNN =␣
↪RandomForestClassifier(n_estimators=100,max_depth=15)

random_forest_top10kNN.fit(train_top10kNN, train_labels_top10kNN.values.
↪ravel())

y_pred_rf_top10kNN = random_forest_top10kNN.predict(test_top10kNN)
```

[54]:
```
f1score   = f1_score        (test_labels_top10kNN, y_pred_rf_top10kNN,␣
↪average = 'macro')
recall    = recall_score    (test_labels_top10kNN, y_pred_rf_top10kNN,␣
↪average = 'macro')
accuracy  = accuracy_score  (test_labels_top10kNN, y_pred_rf_top10kNN)
AUC       = roc_auc_score   (test_labels_top10kNN.values.ravel(),␣
↪random_forest_top10kNN.predict_proba(test_top10kNN),␣
↪multi_class='ovr')

print("RF with kNN's top 10 features")
print('auc =', AUC)
print('acc =', accuracy)
print('f1 =', f1score)
print('recall =', recall)
```

```
RF with kNN's top 10 features
auc = 0.9862309625308666
acc = 0.9255490091055169
f1 = 0.9112650609328079
recall = 0.9054826407097271
```

[55]:
```
support_vector_machine_top10kNN = SVC(C=1.0, gamma=0.1,␣
↪kernel="linear", tol=0.001, max_iter=9999, probability=True)

support_vector_machine_top10kNN.fit(train_top10kNN,␣
↪train_labels_top10kNN.values.ravel())

y_pred_svm_top10kNN = support_vector_machine_top10kNN.
↪predict(test_top10kNN)
```

[56]:
```
f1score   = f1_score        (test_labels_top10kNN, y_pred_svm_top10kNN,␣
↪average = 'macro')
recall    = recall_score    (test_labels_top10kNN, y_pred_svm_top10kNN,␣
↪average = 'macro')
accuracy  = accuracy_score  (test_labels_top10kNN, y_pred_svm_top10kNN)
```

```
AUC       = roc_auc_score   (test_labels_top10kNN.values.ravel(),␣
 ↪support_vector_machine_top10kNN.predict_proba(test_top10kNN),␣
 ↪multi_class='ovr')

print("SVM with kNN's top 10 features")
print('auc =', AUC)
print('acc =', accuracy)
print('f1 =', f1score)
print('recall =', recall)
```

```
SVM with kNN's top 10 features
auc = 0.7208123274003527
acc = 0.6012319228709159
f1 = 0.43940427270240984
recall = 0.48707042825479174
```

Top 10 características do *Neural Network*

[57]:
```
dh_top10_NN = pd.concat([df_lag["TEMPR_Max"], df_lag["TEMPR_Mean"],␣
 ↪df_lag["HRR_Max"], df_lag["TEMPR_Min"], df_lag["HRR_Mean"],
                          df_lag["HRR_Min"], df_lag["EDAR_Min"],␣
 ↪df_lag["EDAR_Max"], df_lag["EDAR_Mean"], df_lag["TEMP_Mean-10"],
                          df_lag["Stress"]], axis=1)
dh_top10_NN.head()
```

[57]:

	TEMPR_Max	TEMPR_Mean	HRR_Max	TEMPR_Min	HRR_Mean	HRR_Min	EDAR_Min	EDAR_Max	EDAR_Mean	TEMP_Mean-10	Stress
0	0.823944	0.821491	0.643243	0.821239	0.641552	0.639566	0.104202	0.107022	0.105191	0.084598	0.0
1	0.830986	0.827471	0.643243	0.828319	0.642973	0.639566	0.102238	0.103630	0.102822	0.089846	0.0
2	0.834507	0.832395	0.645946	0.831858	0.643921	0.640921	0.101213	0.101679	0.101157	0.094890	0.0
3	0.839789	0.837759	0.645946	0.842478	0.645952	0.644986	0.100359	0.099644	0.099952	0.098347	0.0
4	0.848592	0.843123	0.648649	0.846018	0.646764	0.644986	0.100188	0.098796	0.099298	0.101107	0.0

[58]:
```
train_set_top10NN = dh_top10_NN.iloc[:,0:10]
labels_top10NN = dh_top10_NN.iloc[:,10:11]
```

```
train_top10NN, test_top10NN, train_labels_top10NN, test_labels_top10NN␣
↪= train_test_split(train_set_top10NN, labels_top10NN, test_size=0.3,␣
↪random_state=30)
```

[59]:
```
k_nearest_neighbors_top10NN = KNeighborsClassifier(n_neighbors=5,␣
↪metric="minkowski", weights="uniform")

k_nearest_neighbors_top10NN.fit(train_top10NN, train_labels_top10NN.
↪values.ravel())

y_pred_knn_top10NN = k_nearest_neighbors_top10NN.predict(test_top10NN)
```

[60]:
```
f1score   = f1_score        (test_labels_top10NN, y_pred_knn_top10NN,␣
↪average = 'macro')
recall    = recall_score    (test_labels_top10NN, y_pred_knn_top10NN,␣
↪average = 'macro')
accuracy  = accuracy_score  (test_labels_top10NN, y_pred_knn_top10NN)
AUC       = roc_auc_score   (test_labels_top10NN.values.ravel(),␣
↪k_nearest_neighbors_top10NN.predict_proba(test_top10NN),␣
↪multi_class='ovr')

print("KNN with NN's top 10 features:")
print('auc =', AUC)
print('acc =', accuracy)
print('f1 =', f1score)
print('recall =', recall)
```

```
KNN with NN's top 10 features:
auc = 0.9886808858493908
acc = 0.9384038564542047
f1 = 0.9253882270534346
recall = 0.9256332180477944
```

[61]:
```
logistic_regression_top10NN = LogisticRegression(penalty="l2", C=1.0,␣
↪max_iter=999)

logistic_regression_top10NN.fit(train_top10NN, train_labels_top10NN.
↪values.ravel())

y_pred_lr_top10NN = logistic_regression_top10NN.predict(test_top10NN)
```

[62]:
```
f1score   = f1_score        (test_labels_top10NN, y_pred_lr_top10NN,␣
↪average = 'macro')
recall    = recall_score    (test_labels_top10NN, y_pred_lr_top10NN,␣
↪average = 'macro')
accuracy  = accuracy_score  (test_labels_top10NN, y_pred_lr_top10NN)
```

```
AUC       = roc_auc_score   (test_labels_top10NN.values.ravel(),␣
 ↪logistic_regression_top10NN.predict_proba(test_top10NN),␣
 ↪multi_class='ovr')

print("LR with NN's top 10 features:")
print('auc =', AUC)
print('acc =', accuracy)
print('f1 =', f1score)
print('recall =', recall)
```

```
LR with NN's top 10 features:
auc = 0.7394360509576758
acc = 0.6039100160685592
f1 = 0.4407112844808463
recall = 0.48872489091650023
```

[63]:
```
neural_network_top10NN = MLPClassifier(hidden_layer_sizes=(100,),␣
 ↪activation="relu", solver="adam", max_iter=999)

neural_network_top10NN.fit(train_top10NN, train_labels_top10NN.values.
 ↪ravel())

y_pred_nn_top10NN = neural_network_top10NN.predict(test_top10NN)
```

[64]:
```
f1score   = f1_score        (test_labels_top10NN, y_pred_nn_top10NN,␣
 ↪average = 'macro')
recall    = recall_score    (test_labels_top10NN, y_pred_nn_top10NN,␣
 ↪average = 'macro')
accuracy  = accuracy_score  (test_labels_top10NN, y_pred_nn_top10NN)
AUC       = roc_auc_score   (test_labels_top10NN.values.ravel(),␣
 ↪neural_network_top10NN.predict_proba(test_top10NN), multi_class='ovr')

print("NN with NN's top 10 features:")
print('auc =', AUC)
print('acc =', accuracy)
print('f1 =', f1score)
print('recall =', recall)
```

```
NN with NN's top 10 features:
auc = 0.8756088838042072
acc = 0.7485270487412962
f1 = 0.673606919814695
recall = 0.6668949382370738
```

[65]:
```
random_forest_top10NN =␣
 ↪RandomForestClassifier(n_estimators=100,max_depth=15)
```

```
random_forest_top10NN.fit(train_top10NN, train_labels_top10NN.values.
 ↪ravel())

y_pred_rf_top10NN = random_forest_top10NN.predict(test_top10NN)
```

[66]:
```
f1score   = f1_score        (test_labels_top10NN, y_pred_rf_top10NN,␣
 ↪average = 'macro')
recall    = recall_score    (test_labels_top10NN, y_pred_rf_top10NN,␣
 ↪average = 'macro')
accuracy  = accuracy_score  (test_labels_top10NN, y_pred_rf_top10NN)
AUC       = roc_auc_score   (test_labels_top10NN.values.ravel(),␣
 ↪random_forest_top10NN.predict_proba(test_top10NN), multi_class='ovr')

print("RF with NN's top 10 features")
print('auc =', AUC)
print('acc =', accuracy)
print('f1 =', f1score)
print('recall =', recall)
```

```
RF with NN's top 10 features
auc = 0.9913573220709063
acc = 0.9440278521692554
f1 = 0.931641130600339
recall = 0.9267740862805661
```

[67]:
```
support_vector_machine_top10NN = SVC(C=1.0, gamma=0.1, kernel="linear",␣
 ↪tol=0.001, max_iter=9999, probability=True)

support_vector_machine_top10NN.fit(train_top10NN, train_labels_top10NN.
 ↪values.ravel())

y_pred_svm_top10NN = support_vector_machine_top10NN.
 ↪predict(test_top10NN)
```

[68]:
```
f1score   = f1_score        (test_labels_top10NN, y_pred_svm_top10NN,␣
 ↪average = 'macro')
recall    = recall_score    (test_labels_top10NN, y_pred_svm_top10NN,␣
 ↪average = 'macro')
accuracy  = accuracy_score  (test_labels_top10NN, y_pred_svm_top10NN)
AUC       = roc_auc_score   (test_labels_top10NN.values.ravel(),␣
 ↪support_vector_machine_top10NN.predict_proba(test_top10NN),␣
 ↪multi_class='ovr')

print("SVM with NN's top 10 features")
print('auc =', AUC)
print('acc =', accuracy)
print('f1 =', f1score)
```

```
print('recall =', recall)
```

```
SVM with NN's top 10 features
auc = 0.7323050740865438
acc = 0.614354579539368
f1 = 0.44926832067750017
recall = 0.49801924526855207
```