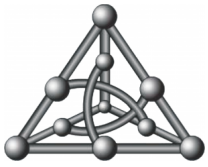


UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL

Programa de Pós-Graduação em Computação Aplicada

Rafael Gonçalves de Oliveira Viana

Classificação de Hipertensos Utilizando o Sinal da Variabilidade da Frequência Cardíaca (VFC) com Auxílio de Inteligência Artificial

Campo Grande - MS

2022

Rafael Gonçalves de Oliveira Viana

Classificação de Hipertensos Utilizando o Sinal da Variabilidade da Frequência Cardíaca (VFC) com Auxílio de Inteligência Artificial

Dissertação apresentada como requisito parcial para obtenção do título de Mestre em Computação Aplicada, ao Programa de Pós-Graduação em Computação Aplicada, da Universidade Federal de Mato Grosso do Sul.

Programa de Pós-Graduação em Computação Aplicada

Universidade Federal de Mato Grosso do Sul

Orientador: Prof. Dr. Milton Ernesto Romero Romero

Coorientador: Profa. Dra. Vera Regina Fernandes da Silva Marães

Campo Grande - MS

2022

Rafael Gonçalves de Oliveira Viana

Classificação de Hipertensos Utilizando o Sinal da Variabilidade da Frequência Cardíaca (VFC) com Auxílio de Inteligência Artificial

Dissertação apresentada como requisito parcial para obtenção do título de Mestre em Computação Aplicada, ao Programa de Pós-Graduação em Computação Aplicada, da Universidade Federal de Mato Grosso do Sul.

Campo Grande - MS, 30 de Novembro de 2022:

Prof. Dr. Milton Ernesto Romero Romero
Orientador
Universidade Federal de Mato Grosso do Sul

Profa. Dra. Vera Regina Fernandes da Silva Marães
Coorientadora
Universidade de Brasília

Profa. Dra. Andréa Teresa Riccio Barbosa
Examinador Interno
Universidade Federal de Mato Grosso do Sul

Prof. Dr. Evandro Mazina Martins
Examinador Interno
Universidade Federal de Mato Grosso do Sul

Campo Grande - MS
2022

Aos meus imprescindíveis ... Dedicatória

Agradecimentos

Agradeço a minha esposa Jéssica e minha irmã Bianca, que vem me apoiando nas mais diversas dificuldades.

Agradeço aos professores: Milton, Evandro e Vera, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender. Ao Evandro que cooperou substancialmente, a Verá por disponibilizar todo seu conhecimento e ao Milton por me orientar nesta fase de minha formação acadêmica.

"Ora, nós sabemos que Deus faz que todas as suas obras cooperem para o bem daqueles que amam a Deus, os que são os chamados segundo o seu propósito."

(Romanos 8:28)

Resumo

O Sistema Nervoso Autônomo é um dos responsáveis pelos ajustes cardiovasculares e seu realce pode ser aferido, utilizando a Variabilidade de Frequência Cardíaca(VFC) pelas ondas R, presente no Eletrocardiograma. Atualmente a hipertensão é altamente prevalente em todos os países. Graças a equipamentos inteligentes vestíveis, uma aglomerado de dados começou a ser coletado e armazenado, entretanto pouco explorados. Novos métodos de aferição da hipertensão utilizando os dados armazenados ajudaria no controle cardiovascular de uma grande quantidade de pessoas. Nesta pesquisa, os dados de VFC coletados por equipamentos vestíveis de baixo custo e de fácil aquisição, foram processados e obteve-se um resultado de 75% de acurácia ao utilizar inteligência artificial na classificação de pacientes hipertensos medicados e não hipertensos (saudáveis). Este resultado provavelmente esta relacionado a eficácia do medicamento utilizado pelos pacientes. Novas pesquisas visando candidatos que possuem pressão alterada (não hipertensos), deve ser realizada, a fim de extrair características destes, ao não utilizar medicamento de controle de pressão arterial.

Palavras-chave: Heart Rate Variability, Autonomic Nervous System, Artificial Intelligence, Hypertension.

Abstract

The Autonomic Nervous System is one of those responsible for cardiovascular adjustments, and its balance can be measured using the Heart Rate Variability (HRV) by the R waves, present in the Electrocardiogram. Currently, hypertension is highly prevalent in all countries. Thanks to smart wearable equipment, a cluster of data began to be collected and stored, however little explored. New methods of measuring hypertension using the stored data would help in the cardiovascular control of a large number of people. In this research, HRV data collected by low-cost and easily acquired wearable equipment were processed and an accuracy of 75% was obtained when using artificial intelligence to classify medicated and non-hypertensive (healthy) hypertensive patients. This result is probably due to the non-efficacy of the drug used by the patients, due to the low dosage or by the body of the same. New research aimed at candidates who have altered blood pressure (non-hypertensive), should be carried out in order to extract characteristics from these, when not using blood pressure control medication.

Keywords: Heart Rate Variability, Autonomic Nervous System, Artificial Intelligence, Hypertension

Lista de ilustrações

Figura 1 – Equipamento galvanômetro (ARRIBAS; PAZ, 1993). 14
Figura 2 – Ondas P,Q,R,S,T e U do eletrocardiograma (REGIS; CALDEIRA; GURJAO, 2016). 15
Figura 3 – Modelo MLP, fonte: Autor 17
Figura 4 – Modelo CNN (DEVELOPER,) 18
Figura 5 – Polar WearLin (fonte: Autor) 19
Figura 6 – Imagem digital, cada bloco representa um pixel (modificada, (GONZALEZ; WOODS,)). 20
Figura 7 – Dimensão Tempo e Frequência dividindo o mesmo campo de percepção (KALHARA et al., 2017). 22
Figura 8 – Transformada PONS (fonte: Autor) 23
Figura 9 – Modelo CRISP-DM (MARBÁN; MARISCAL; SEGOVIA, 2009) 25
Figura 10 – Representação gráfica do R-R em milissegundos (fonte: Autor) 26
Figura 11 – Representação gráfica do R-R por batimento (fonte: Autor) 26
Figura 12 – Identificação de *Outliers* pela técnica de AED (fonte: Autor) 28
Figura 13 – Modelo de remoção de *outlier* (fonte: Autor) 29
Figura 14 – Sinal R-R antes e depois da remoção de *outliers* (fonte: Autor) 29
Figura 15 – Seleção de sinal estável (fonte: Autor) 30
Figura 16 – Sinais Sintéticos em Azul e Vermelho original (fonte: Autor). 31
Figura 17 – Formatação de conjunto de sinal R-R (fonte: Autor) 31
Figura 18 – Imagem R-R bidimensional (fonte: Autor) 32
Figura 19 – Imagem R-R bidimensional replicada (fonte: Autor) 33
Figura 20 – Imagens da classe normal e hipertenso, modelo MO (fonte: Autor) . . . 33
Figura 21 – Imagens da classe normal e hipertenso, modelo MOP (fonte: Autor) . . 34
Figura 22 – Imagens da classe normal e hipertenso, modelo MEIM (fonte: Autor) . 34
Figura 23 – Imagens da classe normal e hipertenso, modelo MEO (fonte: Autor) . . 35
Figura 24 – Imagens da classe normal e hipertenso, modelo MEP (fonte: Autor) . . 35
Figura 25 – Modelo de CNN (fonte: Autor) 36
Figura 26 – Fluxo do treinamento (fonte: Autor) 36
Figura 27 – Fluxo de Dados 37
Figura 28 – Fluxo de Predição (fonte: Autor) 37
Figura 29 – Página de treinamento em desenvolvimento (fonte: Autor) 38
Figura 30 – Resultado do treinamento, *acurracy* por *epoch* do modelo MEIM 39

Lista de tabelas

Tabela 1 – Resultado MEIM 39

Lista de abreviaturas e siglas

SNA	*Sistema Nervoso Autônomo*
VFC	Variabilidade de Frequência Cardíaca
HRV	*Heart Rate Variability*
IA	Inteligência Artificial
ECG	Eletrocardiograma
RNA	Redes Neurais Artificiais
MLP	*Multi-Layer Perceptron*
CNN	*Convolution Neural Network*
PET	Tomografia por Emissão de Pósitrons
TF	Transformadas de Fourier
PONS	*Prometheus Orthonormal*
CRISP-DM	*Cross-Industry Standard Process for Data Mining*
AED	Análise Exploratória de Dados

Sumário

1 INTRODUÇÃO 12

2 TRABALHOS RELACIONADOS 13

3 FUNDAMENTAÇÃO TEÓRICA 14
3.1 Eletrocardiograma ECG 14
3.2 Sistema Nervoso Autônomo - SNA 15
3.3 Inteligência Artificial estado atual e tendências 16
3.4 Modelos de IA 16
3.5 Inteligência Artificial na Medicina 18
3.6 Warable Devices 19
3.7 Representação de Imagem Digital 20
3.8 Transformada de Fourier 21
3.9 *Prometheus Orthonormal - PONS* 21
3.10 Sintetização de Sinal 23

4 METODOLOGIA 25
4.1 Compreensão do Negócio - *Business understanding* 25
4.2 Compreensão dos Dados - *Data Understanding* 27
4.3 Preparação dos Dados - *Data Preparation* 31
4.4 Modelagem - Modeling 33
4.5 Avaliação - Evaluation 36
4.6 Aplicação Web - *Deployment* 38

5 RESULTADOS 39

6 CONCLUSÃO 40

REFERÊNCIAS 41

7 CÓDIGO FONTE 43
7.1 Biblioteca de pré-processamento 43

1 Introdução

A hipertensão é uma doença associada a problemas cardíacos e renais, é altamente prevalente em praticamente todos os países. Calcula-se que pelo menos 50 milhões de norte-americanos são hipertensos e estudos brasileiros têm mostrado prevalência entre 12% e 35% em diferentes regiões. Sabe-se que os indivíduos portadores de hipertensão arterial têm maior risco para desenvolver doença arterial coronariana, além de frequentemente agregarem diversos fatores de risco cardiovascular (SENEVIRATNE et al., 2017).

O Sistema Nervoso Autônomo (SNA) controla em parte o sistema cardiovascular, o qual fornece nervos aferentes e eferentes ao coração, na forma de terminações simpáticas por todo o miocárdio e parassimpáticas para o nódulo sinusal, o miocárdio atrial e o nódulo atrioventricular. A Variabilidade da Frequência Cardíaca (VFC) ou *Heart Rate Variability* (HRV) é um marcador promissor resultante de dois intervalos consecutivos de períodos instantâneos de ondas R, derivado do sinal do eletrocardiograma, especificamente do complexo "PQRST" assim este marcador consegue descrever como o SNA está modulando o coração. A análise da VFC, permite uma avaliação do controle da frequência cardíaca, na qual se ajusta constantemente para os desafios diários (CARVALHO, 2002).

A Inteligência Artificial está cada dia mais presente na medicina, desde predição de câncer ou até mesmo estimação do ângulo da articulação do joelho (LIMA et al.,). Utilização de equipamentos inteligentes vestíveis (equipamentos que interpretam sinais biológico em marcadores vitais), estão cada dia mais disseminados na sociedade atual, pelo custo benefício, assim, conseguir criar sistemas de processamentos inteligentes que consigam associar mais patologias ou situações clínicas, é de grande vantagem para usuários destes equipamentos (LOBO, 2018).

Esta pesquisa utiliza o sinal R-R, derivado no eletrocardiograma para classificação de hipertensão, uma vez que VFC possui relação direta com a modulação do SNA. Relata também, a criação de novos modelos, que permite a visualização dos sinais R-R, para a avaliação desses modelos foi utilizada uma CNN, que classifica pessoas hipertensas medicadas de não hipertensas.

2 Trabalhos Relacionados

Atualmente, o uso de *wearable devices* tem sido introduzido na medicina, obtendo informações contínuas sobre glicemia, Eletrocardiograma e movimento, por exemplo, sistemas que injetam insulina automaticamente, dar uma descarga elétrica no coração do paciente, ou até mesmo realizar a dosagem em horários específicos para um paciente de Parkinson. Assim, essas informações podem ser transmitidas para o médico pelo celular do próprio paciente (LOBO, 2017). Em (LUDERMIR, 2021), sistemas médicos que cruzam dados de remédios prescritos a pacientes, já vem evitando interações ou doses inadequadas, para determinados pacientes que possuem problemas clinos em suas fichas médicas, como por exemplo, problemas renais e hepáticos.

Em (LIMA et al.,), uma rede neural SOM (*self-organizing maps*) e MLP (*multi-layer perceptrons*), foram utilizados para classificar pessoas normais de hipertensos e chagásicos. Em (MARÃES, 2018) um estudo referente a técnicas de para processamento da VFC por CNNs, enquanto que em (GONçALVES, 2019) uma CNN foi desenvolvida para realizar a classificação de pessoas hipertensas utilizando transformação *walevet* e uma CNN.

3 Fundamentação Teórica

3.1 Eletrocardiograma ECG

O ECG é utilizado para registrar a atividade elétrica do coração, que é expressa pelo traçado e suas variações nas ondas do eletrocardiograma. Sendo assim, trata-se do exame mais utilizado para auxiliar no diagnóstico de doenças cardíacas (MORSCH, 2018).

Para captura das ondas, são utilizados eletrodos, que são pequenos dispositivos em formato de ventosa, *clip* ou pinça, que captam e transmitem os estímulos elétricos cardíacos durante a eletrocardiografia. Eles são posicionados conforme a recomendação clínica para um exame eficiente. O mais comum é a utilização de quatro eletrodos periféricos, que são fixados nos braços e tornozelos do paciente, enquanto outros seis são posicionados no tórax, no total eles capturam 12 ângulos diferentes, ou 12 derivações. Para uma melhor captura dos sinais é importante verificar se a pele está limpa e depilada antes de fixar os eletrodos, também é de costume utilizar gel condutor para uma melhor captura dos sinais elétricos gerados pelo coração (MORSCH, 2018). No eletrocardiograma ou ECG, capturado pelo equipamento galvanômetro (Figura 1), existem seis ondas que o formam e que são capturadas durante o exame, sendo elas: P, Q, R, S, T e U.

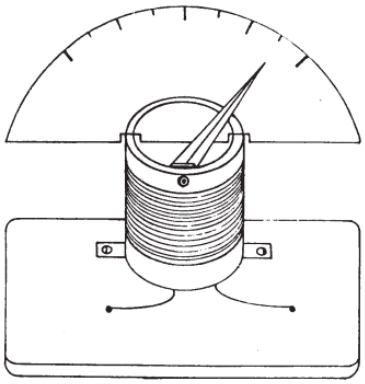


Figura 1 – Equipamento galvanômetro (ARRIBAS; PAZ, 1993).

Cada sinal, intervalo ou complexo, corresponde a uma fase da passagem dos impulsos elétricos responsáveis pela atividade elétrica cardíaca e consequentemente funções do coração, exemplos de cada sinal pode ser visualizada, na Figura 2.

As ondas do ECG precisam ser interpretadas, para obter-se uma avaliação do coração. Quando estão normais possuem um ritmo sinusal, o intervalo entre a onda P e a onda R dura entre 0,12 e 0,20 segundos, evidenciando que o impulso elétrico está seguindo a trajetória normal. Já quando as ondas ECG possuem alterações, o ritmo cardíaco fica caótico, pode ser observado que as ondas P,T e complexo QRS são quase indiscerníveis

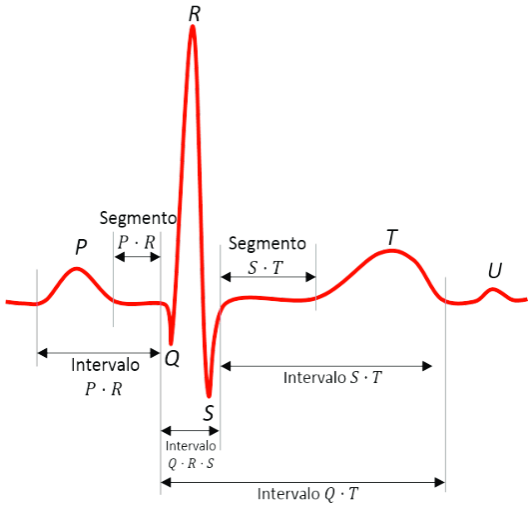


Figura 2 – Ondas P,Q,R,S,T e U do eletrocardiograma (REGIS; CALDEIRA; GURJAO, 2016).

nos traçados, expressando uma fibrilação ventricular, podendo relacionar patologias como hipertensão.

3.2 Sistema Nervoso Autônomo - SNA

O coração é um órgão que apresenta células com ritmicidade própria, capazes de gerar potenciais de ação, responsáveis pelo estabelecimento da frequência cardíaca, cujo controle é feito em parte pelo SNA. Esses ajustes estão divididos em duas áreas denominadas como simpática que atua sobre o miocárdio e uma parassimpática cuja atuação se dá sobre o nó sinoatrial, miocárdio atrial e o nó atrioventricular.

Os ramos simpático e parassimpáticos controlam o ritmo dos batimentos do coração, atuando diretamente no nódulo sinoatrial, no qual é nosso marcapasso natural e possuem as células mais rápidas do miocárdio, que se contraem periodicamente por natureza. Essas duas áreas se comportam inversas uma a outra, onde a ação simpática promove o aumento da frequência cardíaca, enquanto a parassimpática, promove a sua diminuição. Assim, é de se esperar que exista modificações na frequência cardíaca em respostas a estímulos fisiológicos e ambientais como por exemplo: sono, exercício físico, estresse, respiração, obesidade, alterações hemodinâmicas, metabólicas, e variações ocasionadas por patologias do sistema cardiovascular como hipertensão que esta pesquisa busca associar (VANDERLEI et al., 2009).

3.3 Inteligência Artificial estado atual e tendências

As técnicas de Inteligência Artificial (IA), já fazem mais de 70 anos que foram definidas, nos primórdios dos anos 1950, quando Alan Turing publicou o artigo: *"Computing Machinery and Intelligence"* (TURING, 2009), e o termo IA foi cunhado numa conferência no Dartmouth College em 1956, mas somente agora com poder computacional e quantidade de dados disponíveis, começaram a ser utilizadas em problemas cada vez mais complexos.

Atualmente máquinas estão aprendendo desde dirigir automóveis (Google e Tesla) a interpretar linguagens naturais (Google Tradutor) possibilitando traduções em diversas línguas. Na área da saúde vem sendo utilizada para diagnósticos automáticos que são corretos e precisos. Em particular à empresa IFlytek criou um robô que passou no exame nacional para licenciamento de médicos da China, vale ressaltar que o robô não tem o objetivo de substituir humanos é sim ajudar em registros de sintomas, analisar imagens de tomografia computadorizadas, aumentando assim, a eficiência de médicos (LUDERMIR, 2021).

Os sistemas de visão computacional vem evoluindo a cada dia. Sistemas para reconhecimento de imagens vem ganhando qualidade no reconhecimento de objetos. Em 2014 o erro de classificação de imagens utilizando Redes Neurais Profundas (um tipo de IA) foi de 7,3%, e em 2016 foi de 3,6% , superando o desempenho obtido na base do ImageNet, que foi de 5,1%. Hoje já existem computadores ganhando campeonatos de campeões como foi o caso do jogo Go. Em 2017, o AlphaGo Master, aprendeu a jogar por treinamento e ganhou do campeão mundial Ke Jie.

Não é possível programar o computador para ser campeão em jogos como o Go, nem para desempenhar as inúmeras tarefas que os computadores desempenham atualmente, entretanto graças ao Aprendizado de Máquina (AM) que o computador está adquirindo novas habilidades. As técnicas de AM permitem que o computadores aprenda por exemplos, ou seja, aprende por meio dos dados, assim podemos dizer que o AM tornou-se a chave para colocar conhecimento nos computadores. Como pode-se observar a utilização da (IA) é uma provável solução para problemas que não podem ser resolvidos com soluções algorítmicas mas que possuem muitos dados para esse problemas, possibilitando o treinamento de algoritmos que utilizam o AM (LUDERMIR, 2021).

3.4 Modelos de IA

Redes Neurais Artificiais (RNA), buscam simular o comportamento de neurônios biológicos e podem modelar problemas complexos (HAYKIN, 2001).

Uma RNA recebe informações $(x_1, x_2, ...x_m)$. Outro termo utilizado é o $biasb_k$ que é adicionado na Equação (3.1), para fornecer maior liberdade na ponderação dos dados de

entrada, bem como maior acessibilidade da rede.

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{3.1}$$

Uma função de ativação φ é aplicada a este sinal para limitar y_k, o sinal da camada de saída, evitando adições progressivas de acordo com a Equação (3.2):

$$y_k = \varphi(u_k + bk) \tag{3.2}$$

Multi-Layer Perceptron (MLP) é uma RNA que possui mais camadas (ocultas) do que camadas de entrada e saída. Geralmente, uma camada oculta pode resolver qualquer problema contínuo. Uma MLP(Figura 3, ilustra a arquitetura) usa aprendizagem supervisionado e para atualizar os pesos, usa o algoritmo de retropropagação que é baseado na correção de erros. Assim, as variáveis de entrada são ponderadas de forma a minimizar o erro final (KLEINA et al., 2022).

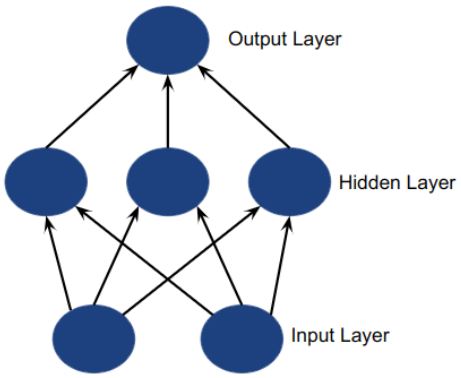


Figura 3 – Modelo MLP, fonte: Autor

Redes Neurais Convolucionais (do inglês *Convolution Neural Network* - CNN), tem aplicação em vários campos da visão computacional, alguns dos quais incluem reconhecimento facial, sistemas biométricos, carros autônomos, detecção de emoções, restauração de imagens, robótica, entre outros. Os algoritmos de *Deep Learning* alcançaram grande progresso no campo da visão computacional. *Deep Learning* é uma implementação das redes neurais artificiais com múltiplas camadas ocultas para imitar as funções do córtex cerebral humano (CHAUHAN; GHANSHALA; JOSHI, 2018).

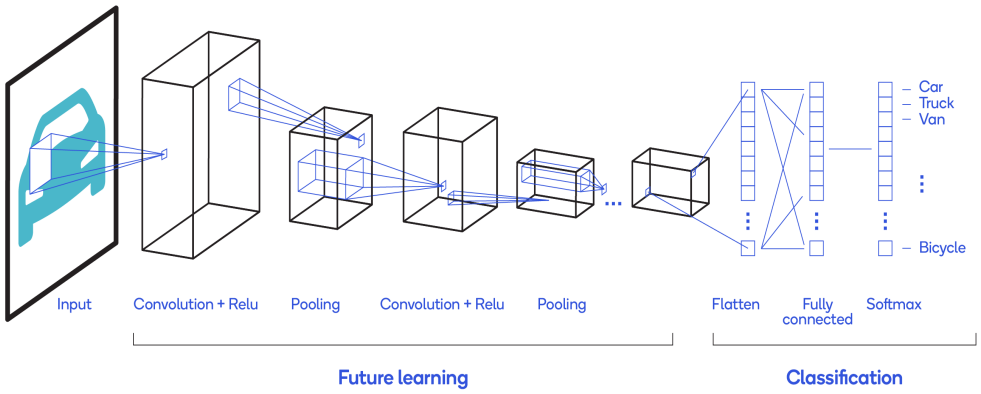


Figura 4 – Modelo CNN (DEVELOPER,)

3.5 Inteligência Artificial na Medicina

O advento do computador trouxe uma enorme capacidade de ampliação do conhecimento do homem, aumentando sua capacidade de calcular e armazenar informações. Ao mesmo tempo, houve um grande avanço das tecnologias voltadas ao diagnóstico pela imagem: o ecocardiograma, tornará o estetoscópio obsoleto, o ultrassom, que substituirá palpação e a percussão do abdome, a ressonância, que tornará os exames de radiologia contrastada obsoleto, e a Tomografia por Emissão de Pósitrons (PET), visam substituir exames clínicos dos pacientes e a relação médico-paciente. A Inteligência Artificial (IA) na medicina é o uso de computadores que, processam um grande volume de dados e seguindo uma metodologia definida por especialistas na área, conseguem propor soluções para problemas médicos, o primeiro trabalho de IA voltada a medicina foi no artigo de Shortlife em 1963 (SHORTLIFFE, 1993).

Computadores podem utilizar sua capacidade de recuperação de informação aliada ao seu processamento para gerar probabilidades de diagnósticos baseados nas IA com algoritmos de decisão estabelecidos e que podem-se auto-modificar em decorrência de resultados obtidos (*self improvement*). Os dados utilizados nos processamentos podem ser obtidos de diversas formas, como em prontuários médicos ou por intermédio de equipamentos de captura de sinais vitais (wearable devices) (LOBO, 2017). Nos estudos de (RIMOLDI, 1988) pode-se compreender que o número de anamnese, exame físico e exames complementares requisitados por internos, residentes e especialistas para resolver um caso varia significativamente indicando a importância da experiência na proposição de possíveis diagnósticos. O reconhecimento de padrões (*pattern recognition*) e de critérios de diagnóstico (combinação de sintomas, sinais e resultados de exames) é usado na determinação de um diagnóstico correto.

O desenvolvimento de sistemas de IA para tomada de decisão, estão sendo cada dia mais usual na medicina moderna, o supercomputador da IBM Watson e o *Deep Mind* da Google vem sendo utilizado para auxiliar médicos nas tomadas de decisões clínicas, analisando dados dos pacientes, permitindo gerar alertas sobre sua evolução, evitando medicações contra indicadas ou conflitantes, além de realizar processamento de imagens de câncer de pele (LUDERMIR, 2021). Um dos problemas que conseguimos observar nas técnicas de IA voltadas à na medicina, é que elas não explicam como chegaram no resultado (*know-why*), apenas diz que chegou em um resultado (*know-what*). Outro desafio que vem sendo enfrentado e problemas de privacidade no compartilhamento dos dados, assim, gradualmente o *big data* vem sendo criada na área de medicina, proporcionando bases de dados sólidas como o *Nacional Health Service*, da Inglaterra (LOBO, 2017).

3.6 Warable Devices

Como é recordado em 3.5, os *big datas* atualmente estão sendo gerados e graças a usuários estão mostrando um interesse crescente em usar praticamente um dispositivo móvel que promete melhorar a qualidade de vida de uma forma que os *smartphones* sozinhos não podem alcançar. Esses dispositivos, que incluem relógios inteligentes, pulseiras, óculos inteligentes, jóias inteligentes, roupas eletrônicas, adesivos de pele e assim por diante, geralmente são chamados simplesmente de *wearables*. Um exemplo são equipamentos que capturar de forma prática (graças utilização de um único equipamento “Polar WearLink") pequeno e discreto) os sinais de um ECG 3.1, assim, armazenar realizar *upload* para um sistema de IA. Para mais informações sobre os *wearables*, consulte a (SENEVIRATNE et al., 2017)

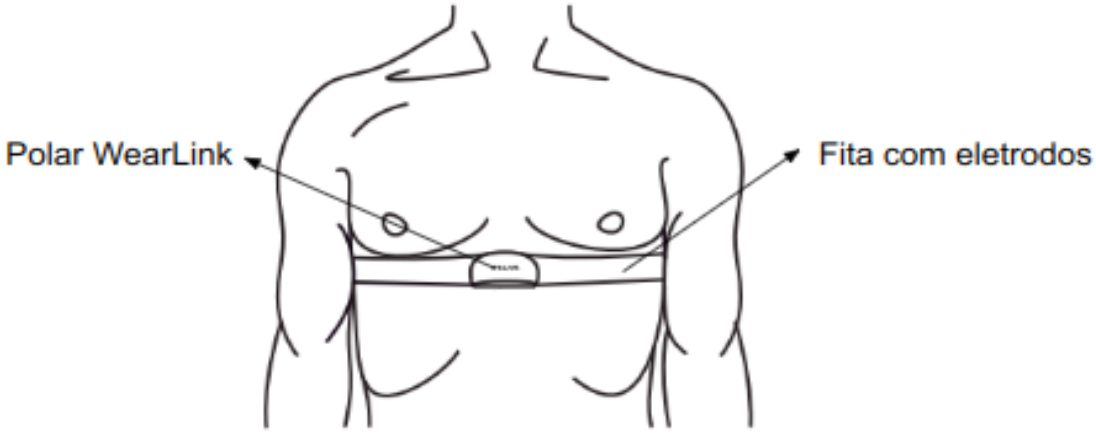


Figura 5 – Polar WearLin (fonte: Autor)

3.7 Representação de Imagem Digital

Uma imagem digital pode ser representada pela função $f(s,t)$, por duas variáveis contínuas, s e t. Convertemos essa função em uma imagem digital por meio da amostragem e quantização, assim, uma imagem digital é a amostragem da imagem contínua em uma matriz 2-D, $f(x,y)$, contendo M linhas e N colunas, sendo que (x,y) são coordenadas discretas. Para fins de praticidade e clareza na notação, pode-se utilizar números inteiros para essas coordenadas discretas:

$$x = 0, 1, 2, ..., M\check{}1 e y = 0, 1, 2, ..., N\check{}1. \tag{3.3}$$

Uma imagem no geral possui um valor em quaisquer coordenadas (x,y) é expresso por $f(x,y)$, onde x e y são números inteiros. Uma imagem digital expande pelas coordenadas de uma imagem é chamada de domínio espacial, com x e y sendo chamadas de variáveis espaciais e coordenadas espaciais.

A representação gráfica é útil ao trabalhar com conjuntos em escala de cinza cujos elementos são expressos em um grupo de três variáveis na forma (x,y,z), onde x e y são coordenadas espaciais e z é o valor da intensidade f nas coordenadas (x,y). A variável z pode possuir intensidades de 0 à 255, onde 0 representa negro e 255 branco, valores entre esses extremos são tonalidades de cinza, sendo possível um total de 256 tons de cinza.

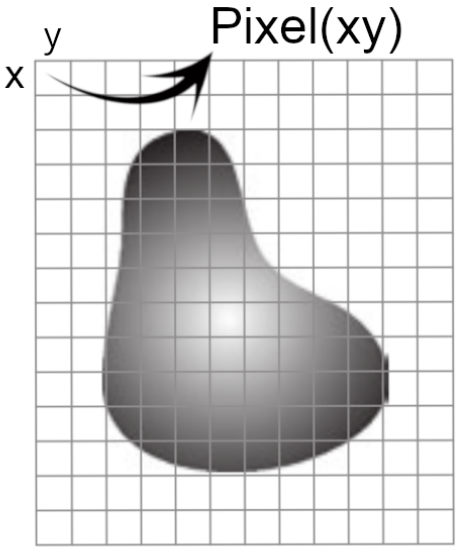


Figura 6 – Imagem digital, cada bloco representa um pixel (modificada, (GONZALEZ; WOODS,)).

Na Figura 6, podemos ver uma imagem em escala de cinza, cada posição da imagem (pixel) representada pelos blocos é equivalente a variáveis x,y enquanto a variável z representa a iluminação/intensidade da variação de cinza.

3.8 Transformada de Fourier

Alguns problemas são difíceis de solucionar diretamente, podem ser mais facilmente resolvidos em outro domínio e posteriormente realizando uma transformada inversa retorna ao domínio original resolvido. Algumas operações tornam-se mais simples e esclarecedoras se trabalharmos no domínio da frequência, domínio este, conseguido a partir das Transformadas de Fourier (TF). O matemático e físico francês Jean Baptiste Joseph Fourier, demonstrou que qualquer forma de onda pode ser representada por uma somatória de senóides e cossenóides de diferentes frequências, amplitude e fases, sendo assim, a TF, decompõe um sinal em suas componentes elementares seno e cosseno (FECHINE; COMPUTAÇÃO,).

Segundo Fourier, qualquer função *f(x)* pode ser escrita na forma da soma de uma série de funções seno e cosseno da seguinte forma geral:

$$f(x) = a_0 + a_1 sen(x) + a_2 sen(2x) + a_3 sen(3x) + b_1 cos(x) + b_2 cos(2x) + b_3 cos(3x) \quad (3.4)$$

Assim, para transformada de fourier, podemos utilizar a seguinte fórmula matemática para obtermos as componentes de frequência:

$$F[f(x)] \equiv F(\omega_x) = \sum_{-\infty}^{\infty} f(x) e^{-2i\omega_x x} dx \quad (3.5)$$

Já para retornarmos para dimensão do tempo, podemos utilizar sua transformada inversa pela seguinte fórmula:

$$f(t) = F^{-1}(F(\omega)) = \frac{1}{2\pi} \sum_{-\infty}^{\infty} F(\omega) e^{i\omega t} d\omega \quad (3.6)$$

Na Figura 7, pode-se observar a dimensão tempo e frequência dividindo o mesmo campo de percepção, ao depender do objetivo um campo sobressaíram melhor, por exemplo se quisermos obter os componentes de frequência de um sinal no tempo, basta olhar sua transformada e obter suas componentes.

3.9 *Prometheus Orthonormal - PONS*

O conjunto Ortonormal Prometheus (PONS) representa uma construção de sequência do tipo Hadamard com propriedades vantajosas de autocorrelação, espectro e relação de potência de pico para média (DELIC; BYRNES; OSTHEIMER, 2004). Sua utilização faz com que a energia em um sinal seja distribuído por todo ele.

Inicialmente a Figura 8a com resolução $128x128$ é convertida utilizando PONS, resultando na Figura 8b. Ao Recortar qualquer quadrante $64x64$ da imagem e aplicarmos

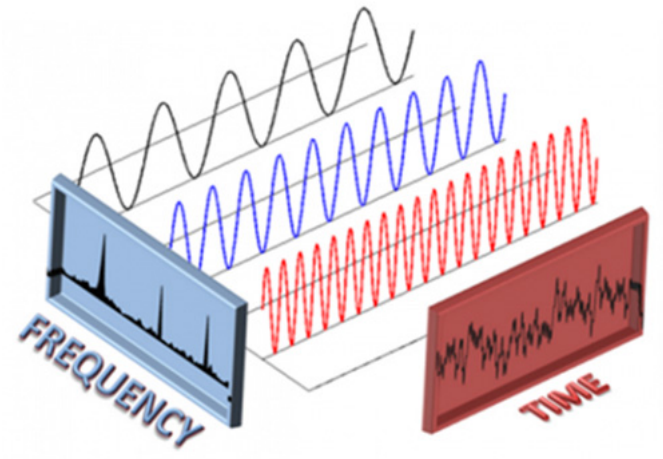
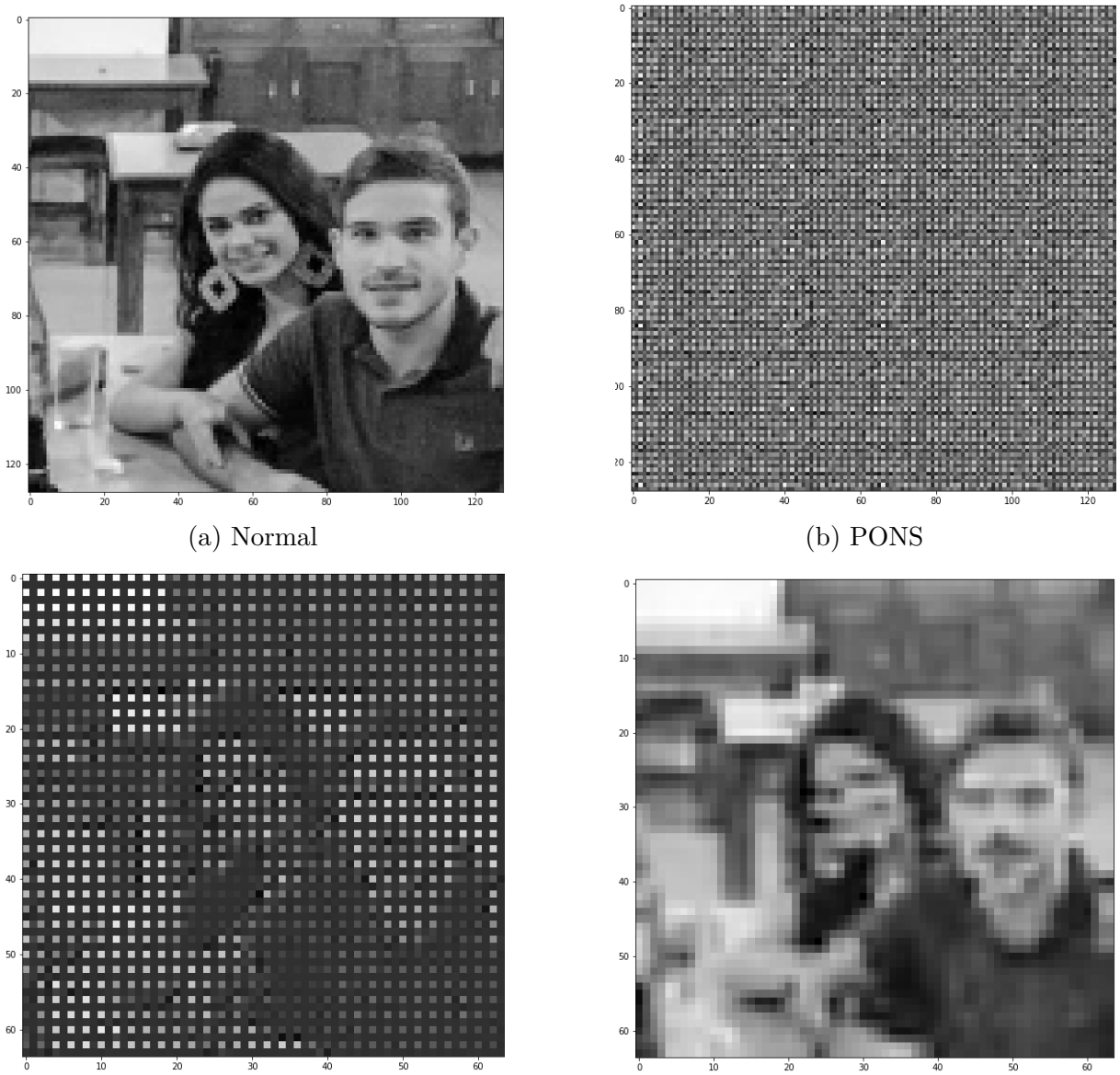


Figura 7 – Dimensão Tempo e Frequência dividindo o mesmo campo de percepção (KALHARA et al., 2017).

novamente PONS, obtemos a Figura 8c, que pós realizar um processamento de passa baixa resulta na Figura 8d. Esse procedimento demonstra que essa transformada distribui as componentes de frequência, que após qualquer recorte é possível recuperar uma imagem(no qual poderia ser um sinal unidimensional) muito próxima à original.



(a) Normal

(b) PONS

(c) Removendo altas frequências

(d) Imagens Recuperadas

Figura 8 – Transformada PONS (fonte: Autor)

Para mais informações referente a transformada PONS, consultar (DELIC; BYRNES; OSTHEIMER, 2004).

3.10 Sintetização de Sinal

Com a popularização da *Deep Learning* a dificuldade em encontrar bases de dados para criação de modelos foi ficando mais evidente, surgindo a necessidade de criação de bases de dados, podendo ou não utilizar dados reais, o desenvolvimento de bases de dados é uma tarefa difícil e demorada, tendo uma complexidade ainda maior quando os dados precisam ser rotulados (NIKOLENKO, 2019).

Para essa finalidade alguns programas semi-automatizados fazem a criação de dados,

tais como o Synner e o Chatette 3 criam conjuntos de palavras, para *chat boots*, esses sistemas permitem que dados sejam gerados a partir de *templates* e especificações definidas pelo desenvolvedor responsável por criar a base de dados Cabendo ao desenvolvedor pensar da melhor forma de criar a base de dados e garantir que os dados sintéticos representam os dados reais (NIKOLENKO, 2019).

4 Metodologia

O modelo de pesquisa CRISP-DM (*Cross-Industry Standard Process for Data Mining*) foi adotado nesta pesquisa. Este é utilizado para mineração de dados e descoberta de conhecimento, composto por seis etapas: Compreensão do Negócio, Compreensão dos Dados, Preparação dos Dados, Modelagem, Avaliação e Aplicação. O ciclo de vida do modelo pode ser observado na Figura 9, para mais informações sobre o modelo, consultar (MARBÁN; MARISCAL; SEGOVIA, 2009).

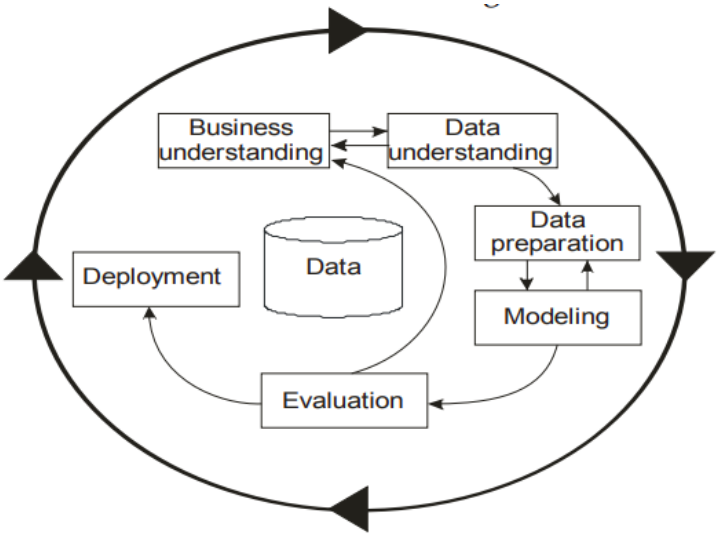


Figura 9 – Modelo CRISP-DM (MARBÁN; MARISCAL; SEGOVIA, 2009)

4.1 Compreensão do Negócio - *Business understanding*

Na seção 3.1, foi demonstrado como sinais elétrico gerados pelo batimento do coração são coletados, este tem com objetivo extrair os sinais R-R, que serão utilizados para mensurar o equilíbrio do SNA seção 3.2. O sinal R-R pode ser visualizado graficamente de duas formas em ms ou batimentos: Na primeira, o eixo Y (sua amplitude) é representado por um valor em ms, ou seja, o tempo entre as ondas R, no eixo X também representado em ms é igual a soma dos valores dos Y antecessores (como o eixo X é casual ao Y, ele inicia em 0), pode-se observar na Figura 10, que o valor de Y, no eixo X instante 0 é igual $550ms$, sendo assim o valor do próximo eixo X será $0ms + 550ms$, assim sucessivamente.

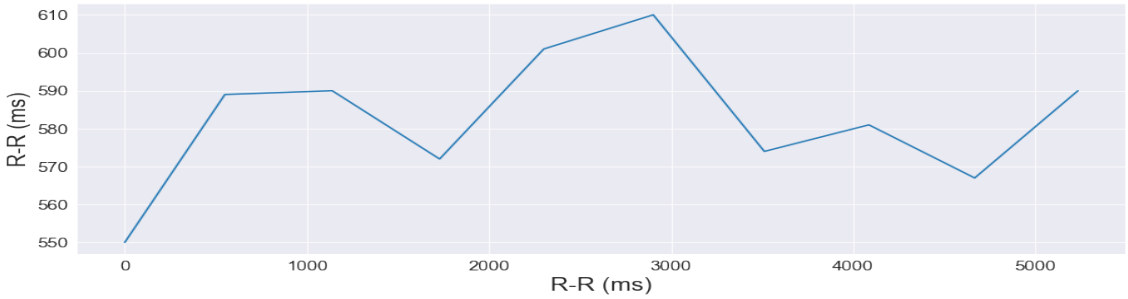


Figura 10 – Representação gráfica do R-R em milissegundos (fonte: Autor)

Já na segunda representação o eixo Y, representa a amplitude em ms, ou seja, a duração do sinal R e no eixo X o número do batimento correspondente, iniciando em 0 e sendo incrementado em 1 por sinal R, essa comportamento pode ser observado na Figura 11.

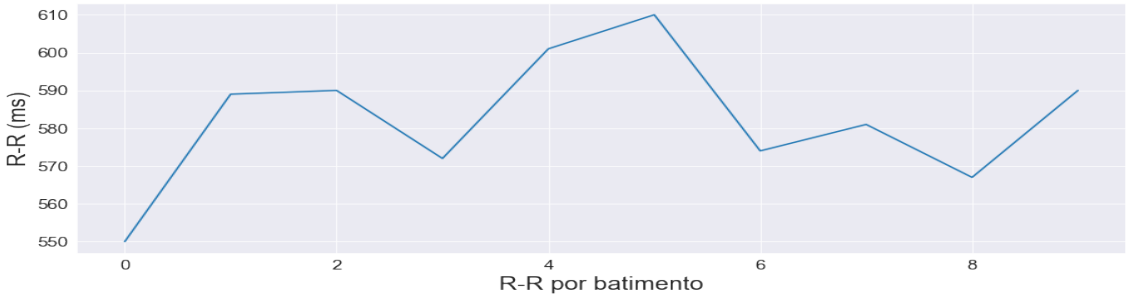


Figura 11 – Representação gráfica do R-R por batimento (fonte: Autor)

Para a análise da VFC no domínio do tempo, assim denominada por expressar os resultados em unidade de tempo (milissegundos), mede-se cada intervalo R-R normal (batimentos sinusais), a partir daí, com base em métodos estatísticos (média, desvio padrão dos intervalos R-R), calculam-se os índices. Este trabalho utilizou os seguintes índices para processamento dos valores R-R:

1. **Média**: valor da média aritmética do sinal R-R;
2. **Mediana**: valor da mediana aritmética do sinal R-R;
3. **STD**: desvio padrão, expresso em ms;
4. **SDNN**: desvio-padrão da média de todos os intervalos RR normais em gravação habitual de 24h;
5. **RMSSD**: raiz quadrada da média do quadrado das diferenças entre os intervalos R-R normais adjacentes, em um intervalo de tempo, expresso em ms;

6. **pNN50**: representa a porcentagem dos intervalos R-R adjacentes com diferença de duração maior que 50ms.

O índices RMSSD e pNN50 representam a atividade parassimpática, pois são encontrados a partir da análise de intervalos R-R adjacentes, encontrados em sinais de baixa duração, uma vez que a coleta nesta pesquisa possuem 4 minutos em média (VANDERLEI et al., 2009).

Além da dimensão tempo na medicina utiliza-se a frequência (Transformada de Fourier, vista na seção 3.8) para avaliar as alterações da variação cardíaca, examinando as potencias de cada componente de frequência. Como visto na seção 3.2, um equilíbrio entre as baixas e altas frequências, sinalizam um coração saudável, enquanto alterações tanto na baixa ou alta frequência, demonstram a incapacidade de adaptabilidade do coração. Visando a utilização da frequência nas analises desta pesquisa foi levantado os seguintes índices:

1. **Power VLF**: representa a potência total das *Very Low Frequency* do sinal R-R.
2. **Power LF**: representa a potência total das *Low Frequency* do sinal R-R.
3. **Power HF**: representa a potência total das *High Frequency* do sinal R-R.
4. **Power Total**: representa a potência total das frequências do sinal R-R.
5. **LF/HF**: representa o equilíbrio entre *Low Frequency* e *High Frequency*.
6. **Peak VLF**: representa quantidade de batimento em *Very Low Frequency* do sinal R-R.
7. **Peak LF**: representa quantidade de batimento em *Low Frequency* do sinal R-R.
8. **Peak HF**: representa quantidade de batimento em *High Frequency* do sinal R-R.
9. **Fraction LF**: representa a normalização total das *Low Frequency* do sinal R-R.
10. **Fraction HF**: representa a normalização total das *High Frequency* do sinal R-R.

4.2 Compreensão dos Dados - *Data Understanding*

O *dataset* de sinais *R-R*, foram classificados previamente por uma equipe técnica nas seguintes classes: hipertensos(medicados[1]) repouso e saudável repouso, sendo um total de $80, 30$ consecutivamente. Nesta pesquisa foi realizada a remoção de sinais *R-R* que

[1] Os pacientes receberam diferentes medicamentos e possivelmente possuem comportamento arterial como de pacientes normais

foram consideradas como *outliers*, para essa finalidade de identificação destes artefatos, utilizou-se o método de Análise Exploratória de Dados de Tukey(Figura 12), ou como é conhecida *AED*. Essa técnica consiste na utilização de quartis para identificar possíveis *outliers*, nesta pesquisa os valores identificados foram classificados como "Valores para Correção"ou apenas VC.

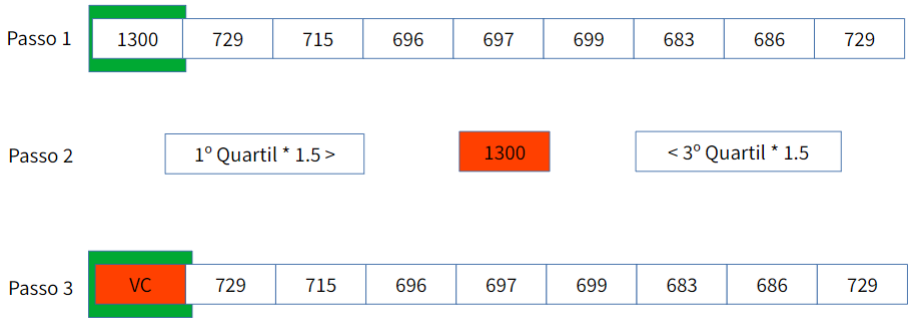


Figura 12 – Identificação de *Outliers* pela técnica de AED (fonte: Autor)

Após a identificação por *AED*, a substituição destes *outliers* por valores de sinais *R-R* que não destoam a realidade da sequência é de extrema importância, uma vez que esses são gerados pelo músculo cardíaco e respeitam uma limitação física do mesmo como apresentado na seção 3.1. Para esse propósito, a técnica adotada foi de janela deslizante, que centraliza o valor a ser substituído, capturando sinais à direita e à esquerda. Caso o valor a ser substituído seja o primeiro (n) ou o último, a sequência se comportará como uma lista cíclica capturando o próximo valor da sequência. Somente valores considerados como não *outliers* são adicionados na janela, ao fim da seleção os valores dentro da janela são utilizados para sintetizar o sinal *R-R* perdido. Os cálculos de média e mediana foram utilizados na sintetização dos valores originais dos sinais.

Janela utilizando média: A média dos valores dentro da janela, faz com que o sinal *R-R* sintetizado seja um valor dentro do *range* possível pelo músculo do coração, entretanto quanto maior a janela, maior vai ser o viés (achatamento ou suavização do ponto) desse valor, sendo assim, janelas menores que 3, deve ser utilizado.

Janela utilizando mediana: Essa técnica busca sintetizar um novo sinal utilizando os valores mais conhecidos do músculo do coração utilizando a janela, assim, janelas maiores que ou igual a cinco são sugeridas, uma vez que mais valores possíveis serão capturados pela janela deslizante.

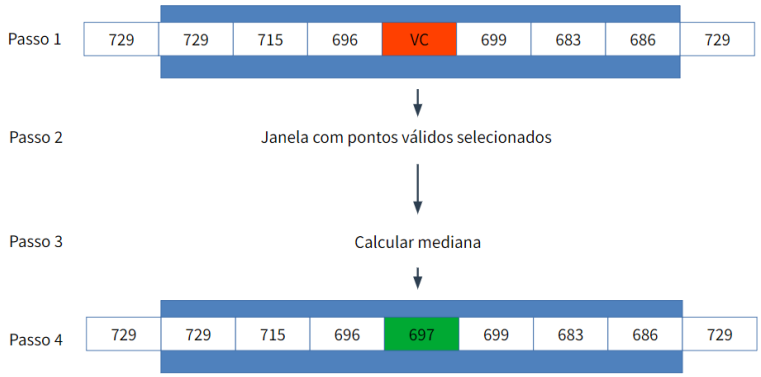
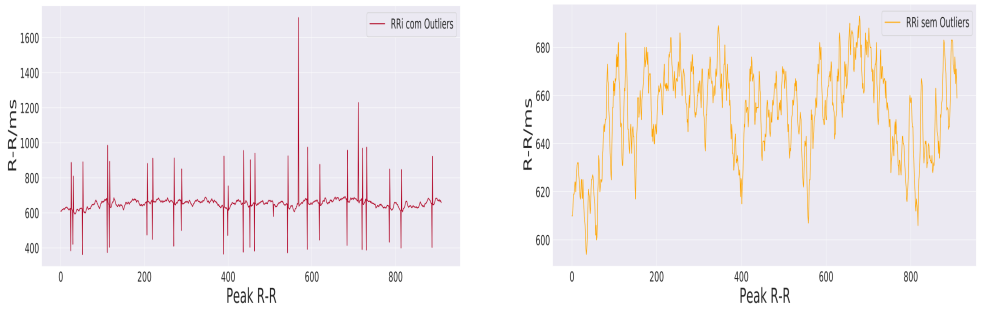


Figura 13 – Modelo de remoção de *outlier* (fonte: Autor)

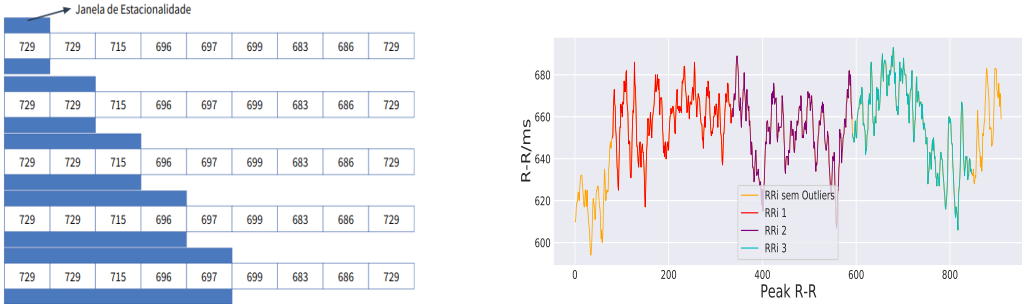
A janela utilizando mediana foi adotada por se aproximar de valores originais esperados, pode-se ver na Figura 13 uma ilustração de seu comportamento, removendo e adicionando valores próximos da realidade (similaridade com o coração). Para representar como ficaram os sinais R-R, gerou-se figuras com a remoção e correção dos sinais R-R, Figura 14a pode-se ver um sinal R-R de um paciente da classe de saudáveis com *outliers* presentes e sem modificação, já na Figura 14b pode-se ver o mesmo sinal R-R da Figura 14a da classe de saudáveis sem *outliers* presente.



(a) Sinal com *Outiliers* (b) Sinal sem *Outiliers*

Figura 14 – Sinal R-R antes e depois da remoção de *outliers* (fonte: Autor)

Outro ponto é selecionar intervalos estáveis do sinal R-R, para isso, foi desenvolvido um detector de estacionalidade, no qual uma janela de estacionalidade é criada, inicialmente um elemento é selecionado e adicionado a ela, posteriormente calcula-se a média e o desvio padrão entre ela e o próxima valor a ser adicionado, caso esse esteja dentro de um fator de 0.2%± de diferença da média e desvio padrão (ou seja, o valor candidato a ser adicionado, não distorcer a estacionalidade) e adicionado a janela, caso o valor não seja aceito, a janela é reiniciada e o valor candidato é adicionado à janela, reiniciando os valores dentro da janela, esse procedimento ocorre até 256 valores estarem dentro da janela, ou todo o sinal ser percorrido, pode-se ver um exemplo da janela na Figura 15a. Neste procedimento foram ignorados os primeiros 80 sinais R-R, isto para que os sinais capturados enquanto o paciente se preparava para realização do exame, não distorça o sinal. Na Figura 15b, pode-se observar três trechos, contendo 256 posições cada, dentro do sinal R-R sem *outlier*.

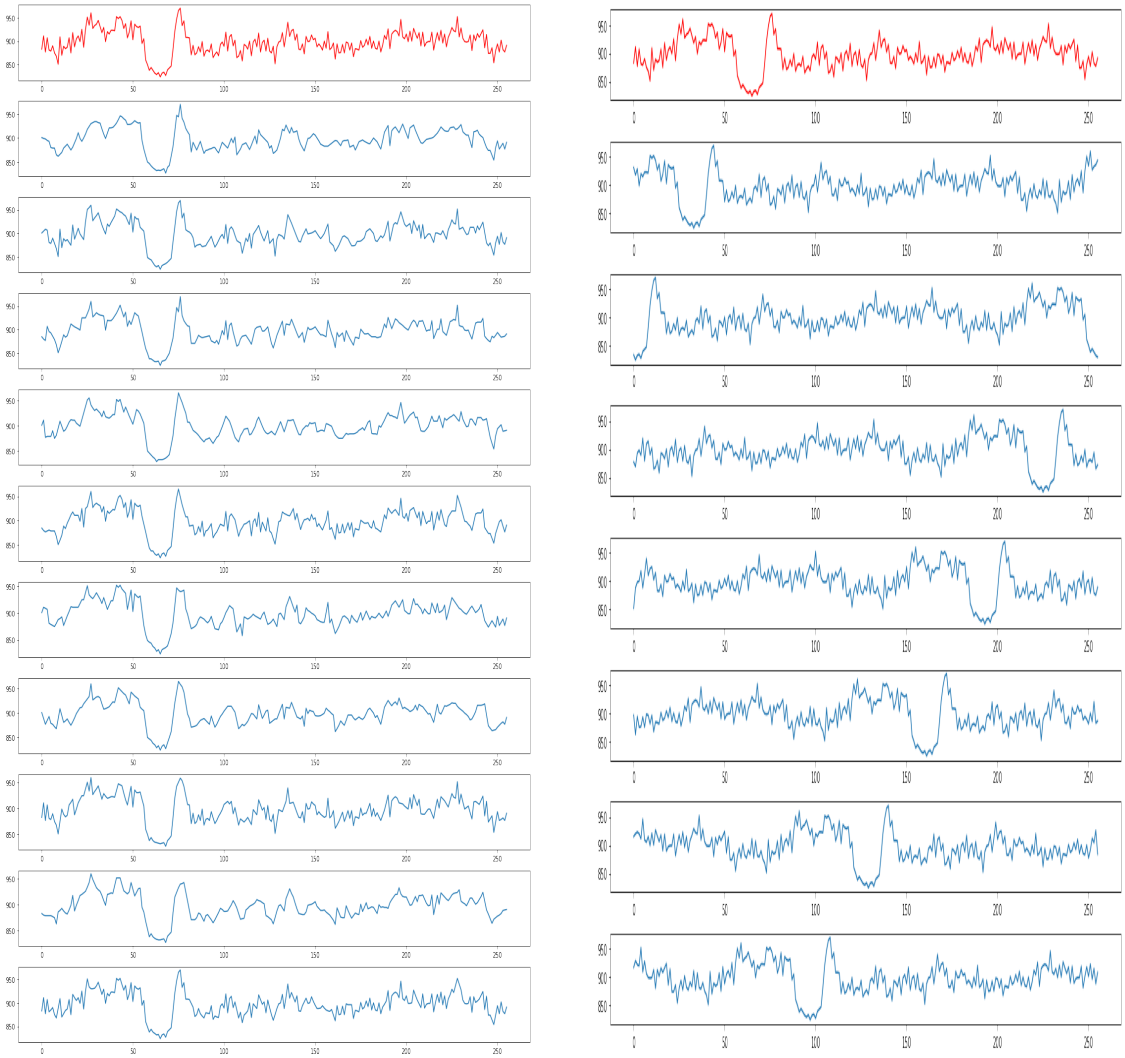


(a) Janela de seleção de sinal estável (b) Seleção de sinais estáveis

Figura 15 – Seleção de sinal estável (fonte: Autor)

Como citado na seção 3.10, alguns problemas no treinamento de RNAs, são dados insuficientes para o treinamento, nesta pesquisa esse é um problema encontrado nos dados de conjuntos R-R. Entretanto, como observado na seção 3.1, sinais derivados do coração são únicos e representam o músculo cardíaco de cada indivíduo, tornando uma geração sintética de sinal um pouco mais complexa, na qual deve-se respeitar a limitação e funcionamento de cada sistema cardiovascular. Deste modo, duas estratégias foram criadas para sintetização de novos conjuntos, elas foram combinadas para criar uma extensa quantidade de dados, a primeira intitulada de Ruído Mediano e a segunda *Array* Circular.

Na primeira a função de substituição de valor R-R de janela mediana é utilizada para sintetização de novos sinais R, aleatoriamente um entre os N é marcado como valor para correção, que posteriormente é substituído pela janela mediana (captura 2 sinais R antecessores e posteriores ao mercado), esse procedimento ocorre 100 vezes para cada conjunto original, assim modificando-o aleatoriamente, uma lista de 10 novos conjunto R-R é criada para cada conjunto original, como pode-se observar na 16a, na qual o primeiro conjunto em vermelho (original), deu origem a nove conjuntos em azul. Na segunda, cada conjunto R-R original dá origem a oito novos conjuntos, no qual cada novo conjunto busca adiantar em 64 posições o sinal, criando um círculo, esse procedimento busca criar novos conjuntos a partir do mesmo conjunto original, apenas modificando suas posições sem perder os sinais R que o compõe, na Figura 16b observa-se esse comportamento ao criar novos sinais, no qual cada um em azul é derivado do sinal original em vermelho.



(a) Ruído Mediano (b) Circular

Figura 16 – Sinais Sintéticos em Azul e Vermelho original (fonte: Autor).

4.3 Preparação dos Dados - *Data Preparation*

Nesta pesquisa um dos objetivo é a criação de imagens bidimensionais dos conjuntos de sinais R-R, essas imagens serão utilizadas no treinamento da IA. Um conjunto de valores: $R_0, R_2, ..., R_n$, podem ser transformados em uma imagem ao dividir esse conjunto em linhas como o exemplo da Figura 17, cada linha posteriormente e realocada em ordem, uma após a outra.

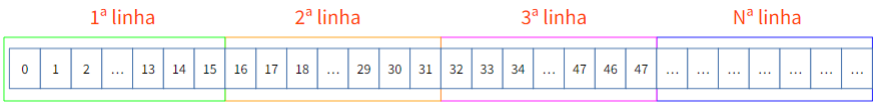


Figura 17 – Formatação de conjunto de sinal R-R (fonte: Autor)

Na Figura 18a e 18b pode-se observar como a imagem será ordenada nas coordenadas (x, y), os valores inicia-se da esquerda para direita, um conjunto de 256 valores é transformado

em uma imagem com 16 linhas e 16 colunas (x, y), ou seja $16x16$, iniciando em 0 e terminado em 255. Na Figura 18c os valores originais do sinal R-R são normalizados para representar a faixa de cinza, ou seja como cada pixel pode ter uma valor de 0 a 255 discretos e inteiros, os valores originais do sinal R-R são normalizados nesta faixa de valores, o menor valor do *dataset* é normalizada em 0 e o maior em 255, sendo que 0 é preto e 255 é branco, valores entre essa faixa são tons de cinza. Após essa transformação (quebra de linha e normalização) uma imagem em tons de cinza é criada, um exemplo de transformação pode ser observada na Figura 18d.

Y

X	0	1	2	3	4	.....	.....	.....	12	13	14	15	16
0	0	1	2	3	4	.....	.....	.....	11	12	13	14	15
1	16	17	18	19	20	.....	.....	.....	27	28	29	30	31
2	32	33	34	35	36	.....	.....	.....	43	44	45	46	47
3	48	49	50	51	52	.....	.....	.....	59	60	61	62	63
4	64	65	66	67	68	.....	.....	.....	75	76	77	78	79
...	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....
...	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....
...	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....
12	176	177	178	179	180	.....	.....	.....	187	188	189	190	191
13	192	193	194	195	196	.....	.....	.....	203	204	205	206	207
14	208	209	210	211	212	.....	.....	.....	219	220	221	222	223
15	224	225	226	227	228	.....	.....	.....	235	236	237	238	239
16	240	241	242	243	244	.....	.....	.....	251	252	253	254	255

(a) Posições

Y

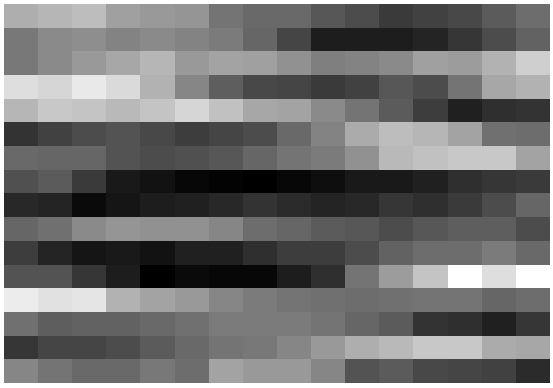
X	0	1	2	3	4	.....	.....	.....	12	13	14	15	16
0	569	571	573	573	565	.....	.....	.....	537	539	541	546	551
1	554	559	560	557	559	.....	.....	.....	529	531	536	542	548
2	554	559	563	567	571	.....	.....	.....	559	565	564	570	578
3	582	580	585	581	570	.....	.....	.....	545	542	553	587	570
4	571	576	575	572	575	.....	.....	.....	546	538	530	534	535
...	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....
...	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....
...	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....
12	544	544	536	529	521	.....	.....	.....	564	575	591	586	591
13	586	583	584	570	566	.....	.....	.....	552	553	553	549	551
14	552	547	548	548	550	.....	.....	.....	546	535	534	530	536
15	536	540	540	542	546	.....	.....	.....	572	576	576	568	567
16	558	553	550	550	554	.....	.....	.....	546	541	540	539	533

(b) R-R original novas posições

Y

X	0	1	2	3	4	.....	.....	.....	12	13	14	15	16
0	174	182	189	160	58	.....	.....	.....	65	72	91	109	120
1	138	142	131	138	29	.....	.....	.....	36	54	76	98	120
2	138	153	167	182	138	.....	.....	.....	153	167	182	138	160
3	156	178	207	222	214	.....	.....	.....	233	218	178	87	76
4	182	200	196	185	196	.....	.....	.....	91	61	32	47	51
...	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....
...	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....
...	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....
12	83	83	54	29	0	.....	.....	.....	156	196	255	236	255
13	236	225	229	178	163	.....	.....	.....	112	116	116	102	109
14	112	94	98	98	105	.....	.....	.....	91	51	47	32	54
15	54	69	69	76	91	.....	.....	.....	185	200	200	171	167
16	134	116	105	105	120	.....	.....	.....	91	72	69	65	43

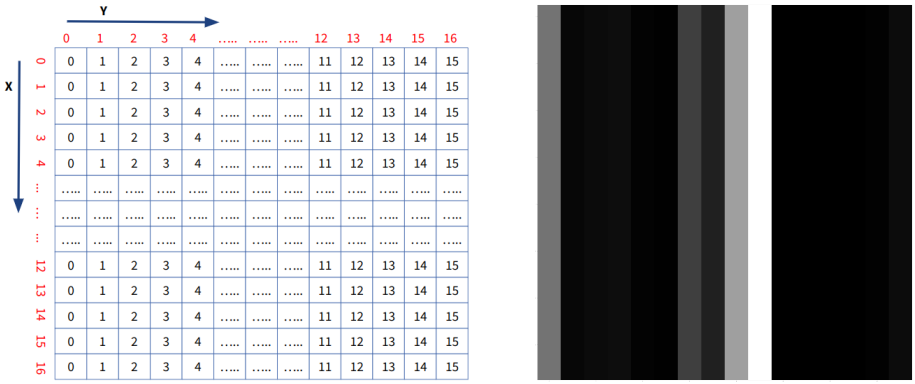
(c) Imagem R-R normalizada



(d) Imagem R-R

Figura 18 – Imagem R-R bidimensional (fonte: Autor)

Outro modelo proposto de imagem foi criado utilizando os 16 índices(seis na dimensão do tempo e dez na frequência na mesma ordem enumerada na seção 4.1) extraídos do sinal R-R, esses índices foram replicados 15 vezes(Figura 19a), criando uma matriz 16x16, assim uma imagem bidimensional Figura 19b, representa os índices utilizados na medicina, essa replicação cria uma redundância com objetivo de remover ruídos no treinamento da IA.



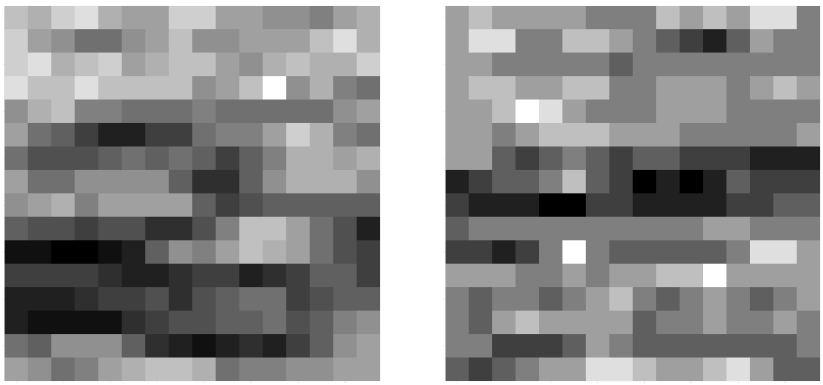
(a) Normal (b) Hipertenso

Figura 19 – Imagem R-R bidimensional replicada (fonte: Autor)

4.4 Modelagem - Modeling

Foram criados cinco modelos de dados de entrada, cada modelo utiliza uma estratégia para ser utilizados na IA: Modelo Original, Modelo em PONS, Modelo Estendido em Índices Médicos, Modelo Estendido Original, Modelo Estendido em PONS. Os modelos serão explicados um a um abaixo.

Modelo original - MO: Este modelo é composto pelo sinal de entrada original, um sinal que contém 256 posições é transformado em uma matriz de 16x16, pode-se observar este modelo na Figura 20. O objetivo deste modelo é criar formas visíveis proporcionadas pela variação dos sinais R-R, criando assim uma perspectiva na troca de variação, causada pela mudança de coloração cinza, assim mudanças na frequência, causará mudança de tonalidade dos pixels.

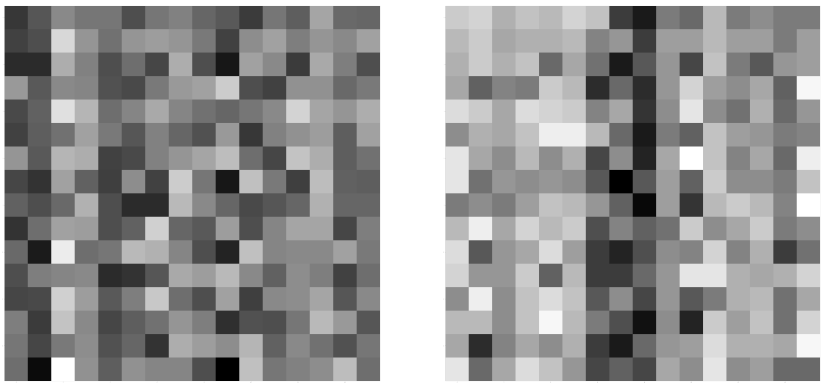


(a) Normal (b) Hipertenso

Figura 20 – Imagens da classe normal e hipertenso, modelo MO (fonte: Autor)

Modelo em PONS - MOP: Neste modelo é semelhante ao da Figura 20, após a construção de uma matriz 16x16 com os 256 sinais é utilizado a transformada PONS(seção 3.9), este procedimento busca distribuir proporcionalmente pela imagem as componentes

de frequência, assim teoricamente o sinal teria uma redundância facilitando o aprendizado de máquina, pode-se observar este modelo na Figura 21.

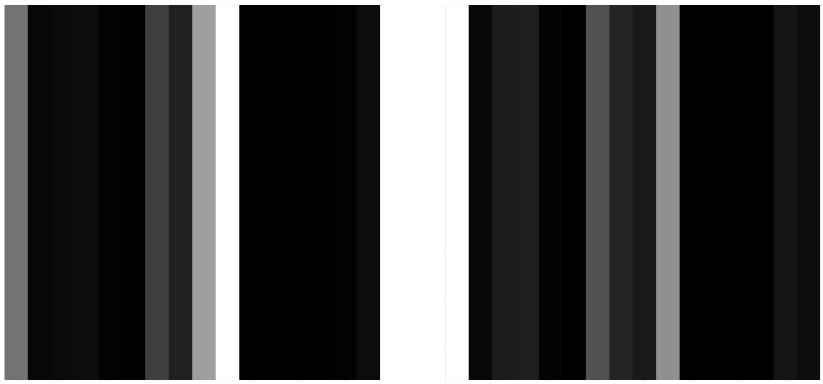


(a) Normal (b) Hipertenso

Figura 21 – Imagens da classe normal e hipertenso, modelo MOP (fonte: Autor)

Modelo Estendido em Índices Médicos - MEIM: Neste modelo é composto pelo resultado dos índices médicos, no total são 16 índices na seguinte sequência: Média, Mediana, STD, SDNN, RMSSD, pNN50, Fraction LF, Fraction HF, LF/HF, Power VLF, Power LF, Power HF,Power Total, Peak VLF, Peak LF, Peak HF.

Posteriormente são estendidos(replicados) para criar uma matriz 16x16, esse modelo busca utiliza as métricas mais utilizadas e significativas na avaliação da VFC pela medicina, uma vez que esses mesmos são observados por médicos para tentar classificar as classes, pode-se observar este modelo na Figura 22.

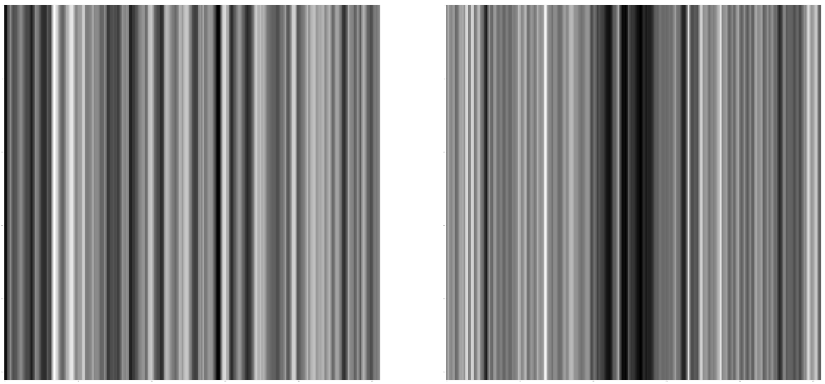


(a) Normal (b) Hipertenso

Figura 22 – Imagens da classe normal e hipertenso, modelo MEIM (fonte: Autor)

Modelo Estendido Original - MEO: Neste modelo o sinal R-R original com suas 256 posições, como no exemplo da Figura 17 é utilizado sem a quebra de linha, entretanto é replicado assim como na Figura 19a, ao invés de 16 vezes é realizada 256 réplicas na horizontal, criando uma matriz $256x256$. Assim fica bem nítido a variação de cinza, em teoria essa técnica facilitaria a localização da mudança de frequência(criando

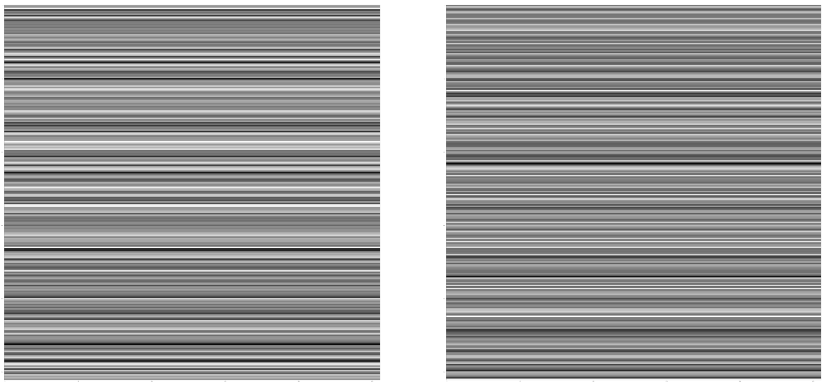
redundância), induzindo o aprendizado de máquina, pode-se observar este modelo na Figura 23.



(a) Normal (b) Hipertenso

Figura 23 – Imagens da classe normal e hipertenso, modelo MEO (fonte: Autor)

Modelo Estendido em PONS - MEP: Como no MOP, o objetivo é distribuir as componentes de frequência, entretanto neste modelo o sinal é estendido em 256 posições, assim como na Figura 23, ele utiliza o vetor original, replicado horizontalmente que posteriormente é transformado com PONS, para distribuição do sinal(algo interessante é que ele inverte as linhas para horizontal), pode-se observar este modelo na Figura 24.



(a) Normal (b) Hipertenso

Figura 24 – Imagens da classe normal e hipertenso, modelo MEP (fonte: Autor)

O modelo de IA escolhido foi CNN revisado na seção 3.4, este foi escolhido com objetivo de validar a visualização das imagens, uma vez que simula o córtex cerebral humano. As *Layers* neste teste foi: primeira *layer* 16 com filtro 3x3 ativação relu, seguido de uma *layer maxpooling* 2x2, seguido de outra *layer* 16 com filtro 3x3 e ativação relu, seguido de um *drop* de 50% e um *maxpooling* 2x2 e um *flatten*, no qual entra em uma *layer* densa de 64 neurônios, com ativação relu, na sequência a última *layer* de 2 neurônios, resultando em uma classificação, pode-se ver o modelo na Figura 25.

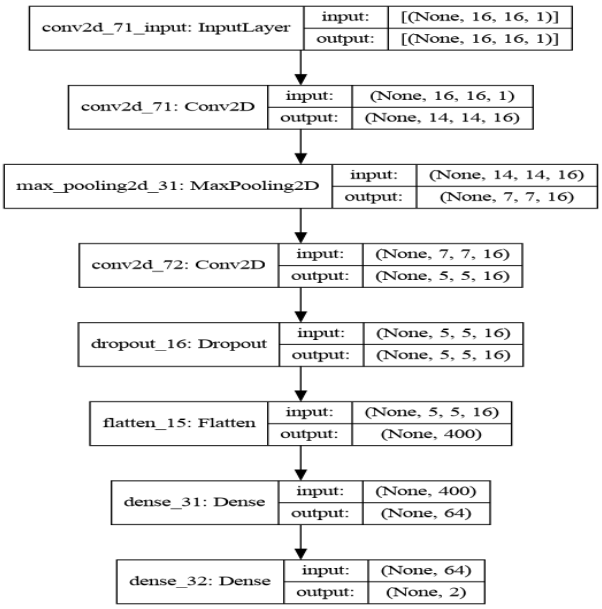


Figura 25 – Modelo de CNN (fonte: Autor)

4.5 Avaliação - Evaluation

Os modelos de dados foram treinados utilizando um notebook Dell 7460, com a biblioteca tensorflow versão 2.6.2 na linguagem Python versão 3.6. O *dataset* fornecido passou por uma normalização, pela menor classe, passando por uma seleção de sinal realizada com o algoritmo de estacionalidade e sintetização, criando os dados de treinamento, teste e validação global. O fluxo das estruturas de treinamento, pode ser visualizado na Figura 26.

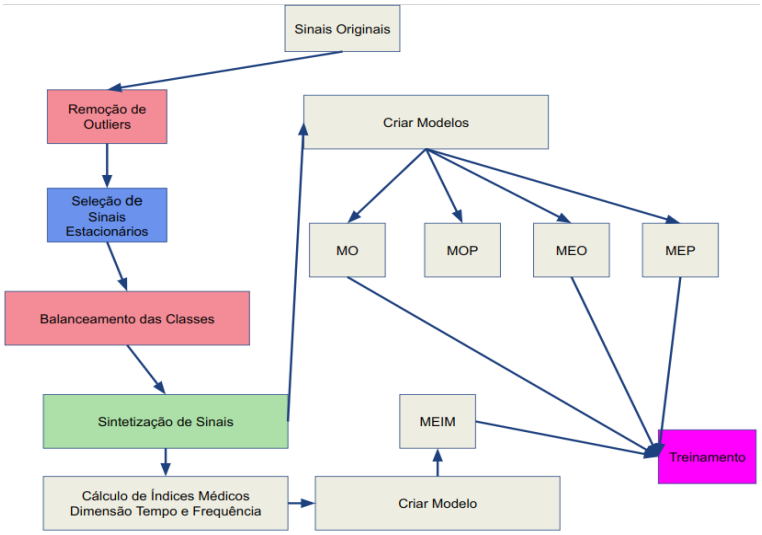


Figura 26 – Fluxo do treinamento (fonte: Autor)

Para o aprendizado de máquina, 23 sinais hipertensos e normais um total de 46 conjuntos foram utilizados. Que após sinterização resultou em 11.776 sinais hipertensos

e normais um total de 23.552 conjuntos, destes 18.841 foram para treinamento e 4.711 para validação cruzada(avaliação a cada fim de época, para ajustes de pesos). Ao final do treinamento a fim de aferir a IA, 6 sinais de ambas classes (12 conjuntos), foram dispostos para sintetizar um conjunto de validação global, que resultou em 6.144 conjuntos sendo metade de cada classe, estes nunca foram anteriormente utilizados na IA, na Figura 27, pode-se ver essa separação em alto nível de abstração.

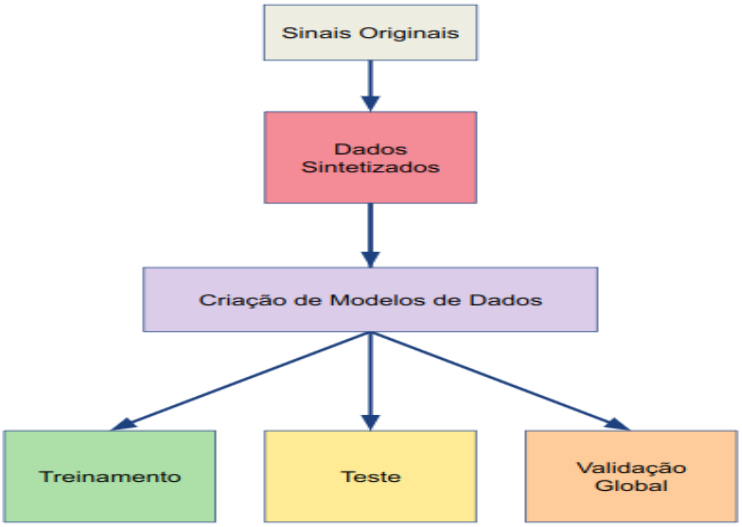


Figura 27 – Fluxo de Dados

Na Figura 28, observar-se que após o modelo treinado, o modelo é usufruídos para realização de aferimentos, com o conjunto de dados de validação global.

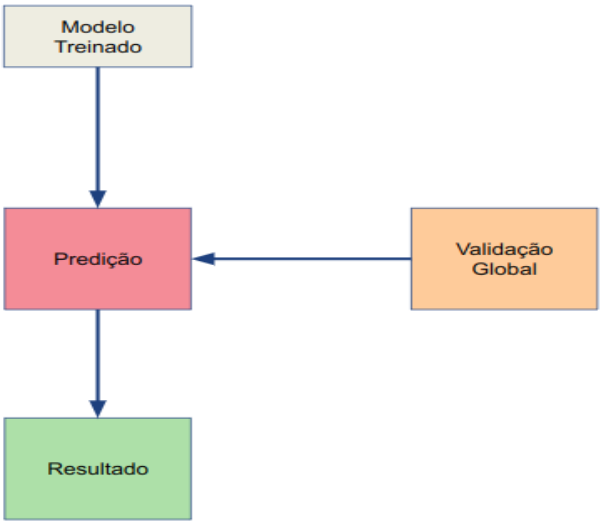


Figura 28 – Fluxo de Predição (fonte: Autor)

4.6 Aplicação Web - *Deployment*

Uma aplicação WEB foi desenvolvida para treinamento e validação. Essa ferramenta está codificada em Angular e Python, pode-se ver a página de treinamento e validação na Figura 29.

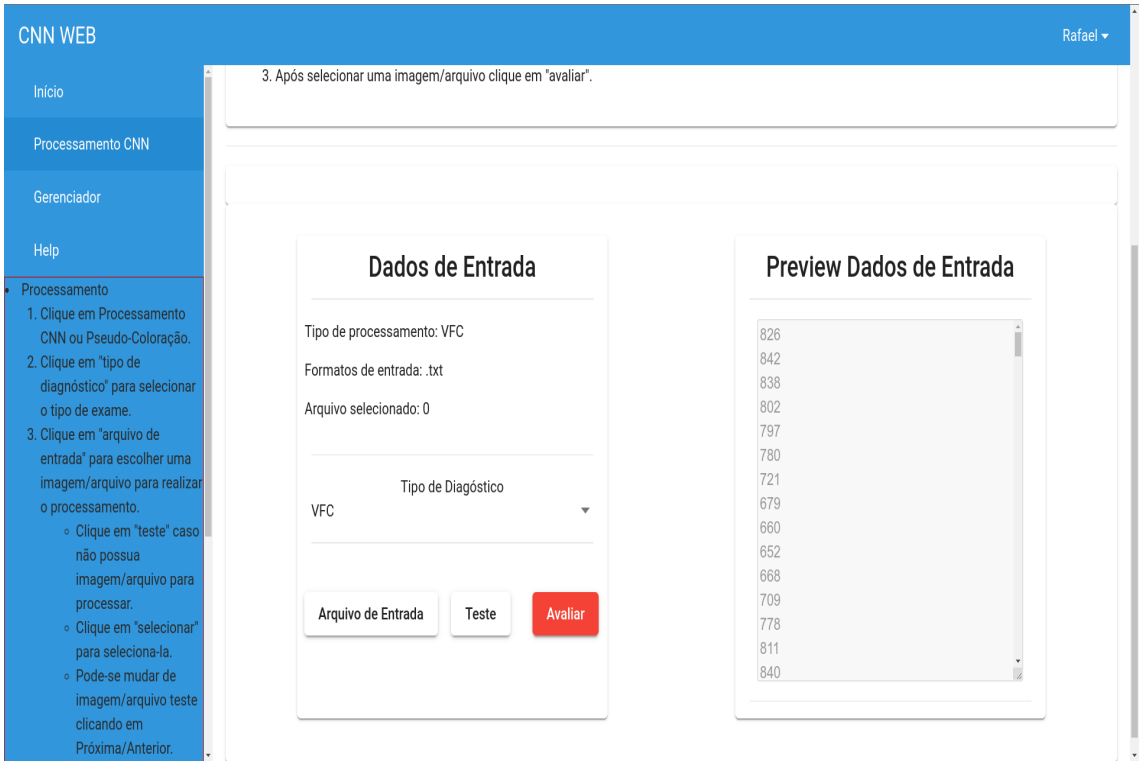


Figura 29 – Página de treinamento em desenvolvimento (fonte: Autor)

5 Resultados

O resultado do MEIM foi o mais promissor, utilizando 1000 épocas, conseguiu uma *accuracy* de 75%. A utilização dos índices médicos colaboram com os resultados, uma vez que são utilizados por médicos nas análises do VFC. Os resultados deste modelo são descritos detalhadamente na Tabela 1, no qual hipertensos é a classe 0 e normais classe 1.

Tabela 1 – Resultado MEIM

	precision	recall	f1-score	support
0	0.83	0.71	0.77	7
1	0.67	0.80	0.73	5
accuracy			0.75	12
macro avg	0.75	0.75	0.75	12
weighted	0.76	0.75	0.75	12

Na Figura 30, mostrar como o treinamento teve sua evolução durante as épocas, as validações de treinamento ficaram acima de 70%, indicando que durante as épocas a acurácia continuou acima de 70%.

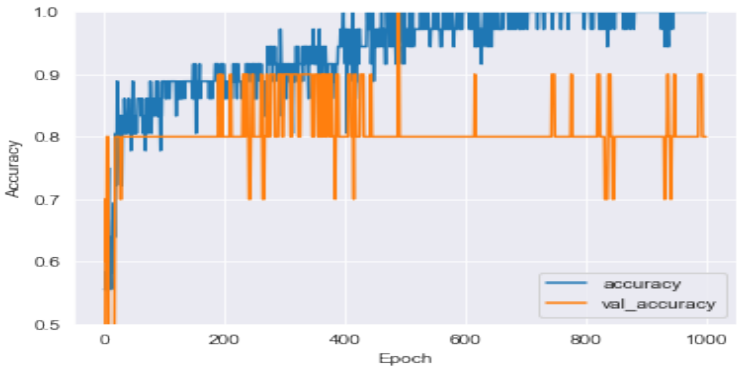


Figura 30 – Resultado do treinamento, *acurracy* por *epoch* do modelo MEIM

6 Conclusão

O treinamento da CNN, obteve resultados satisfatórios com *accuracy* de 75%, assim, esta pesquisa demonstra que a VFC, pode ser utilizada para classificar hipertensão, sem levar em conta outros fatores como idade, patologias, situação comportamental entre outros. Este resultado provavelmente esta relacionado a eficácia do medicamento utilizado pelos pacientes. Novas pesquisas visando candidatos que possuem pressão alterada (não hipertensos), deve-se ser realizada, a fim de extrair características destes, ao não utilizar medicamento de controle de pressão arterial.

Referências

ARRIBAS, S. D.; PAZ, A. M. da. Galvanômetro. *Caderno Brasileiro de Ensino de Física*, Universidade Federal de Santa Catarina (UFSC), v. 10, n. 1, p. 88–92, 1993. Citado 2 vezes nas páginas 8 e 14.

CARVALHO, J. L. Sistema para análise da variabilidade da frequência cardíaca. *Universidade de Brasília–UnB*, 2002. Citado na página 12.

CHAUHAN, R.; GHANSHALA, K. K.; JOSHI, R. Convolutional neural network (cnn) for image detection and recognition. In: *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*. [S.l.: s.n.], 2018. p. 278–282. Citado na página 17.

DELIC, H.; BYRNES, J.; OSTHEIMER, G. The prometheus orthonormal set for wideband cdma. In: *Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference (IEEE Cat. No.04CH37521)*. [S.l.: s.n.], 2004. v. 2, p. 437–440 Vol.2. Citado 2 vezes nas páginas 21 e 23.

DEVELOPER, Q. *Deep learning and Convolutional Neural Networks for Computer Vision*. Citado 2 vezes nas páginas 8 e 18.

FECHINE, J. M.; COMPUTAÇÃO, G. P. A transformada de fourier e suas aplicações. Citado na página 21.

GONZALEZ, R.; WOODS, R. *Processamento de Imagens Digitais 3ª Edição*. [S.l.]: Person. ISBN 9788581435862. Citado 2 vezes nas páginas 8 e 20.

GONçALVES, U. C. *Aplicação de rede neural convolucional para o estudo da variabilidade da frequência cardıaca*. 34 p. Dissertação (Mestrado) — Universidade Federal de Mato Grosso do Sul, CAMPO GRANDE, 2019. Citado na página 13.

HAYKIN, S. *Redes neurais: princípios e prática*. [S.l.]: Bookman Editora, 2001. Citado na página 16.

KALHARA, P.; JAYASINGHEARACHCHD, V.; DIAS, A.; RATNAYAKE, V.; JAYAWARDENA, C.; KURUWITAARACHCHI, N. Treespirit: Illegal logging detection and alerting system using audio identification over an iot network. In: . [S.l.: s.n.], 2017. p. 1–7. Citado 2 vezes nas páginas 8 e 22.

KLEINA, M.; SANTOS, M. N. d.; SANTOS, T. N. d.; MARQUES, M. A. M.; SILVA, W. d. A. Técnicas de inteligência artificial aplicadas para previsão da posição de times do campeonato brasileiro de futebol. *Journal of Physical Education*, SciELO Brasil, v. 32, 2022. Citado na página 17.

LIMA, R. R.; ANDRADE, S. F.; CARVALHO, J. L.; BAUCHSPIESS, A. Classificação neural som e mlp de indivíduos normais, chagásicos e hipertensos por meio de análise da variabilidade da frequência cardíaca. Citado 2 vezes nas páginas 12 e 13.

LOBO, L. C. Inteligência artificial e medicina. *Revista Brasileira de Educação Médica*, SciELO Brasil, v. 41, p. 185–193, 2017. Citado 3 vezes nas páginas 13, 18 e 19.

LOBO, L. C. *Inteligência artificial, o Futuro da Medicina e a Educação Médica*. [S.l.]: SciELO Brasil, 2018. 3–8 p. Citado na página 12.

LUDERMIR, T. B. Inteligência artificial e aprendizado de máquina: estado atual e tendências. *Estudos Avançados*, SciELO Brasil, v. 35, p. 85–94, 2021. Citado 3 vezes nas páginas 13, 16 e 19.

MARBÁN, Ó.; MARISCAL, G.; SEGOVIA, J. A data mining & knowledge discovery process model. In: *Data mining and knowledge discovery in real life applications*. [S.l.]: IntechOpen, 2009. Citado 2 vezes nas páginas 8 e 25.

MARÃES, L. F. *TRATAMENTO E ANÁLISE DE SINAIS DE VARIABILIDADE DA FREQUÊNCIA CARDÍACA*. 37 p. Dissertação (Mestrado) — Universidade Federal de Mato Grosso do Sul, CAMPO GRANDE, 2018. Citado na página 13.

MORSCH, D. J. A. O que são as ondas do eletrocardiograma e como interpretar? *telemedicinamorsch*, v. 24, 2018. Disponível em: <https://telemedicinamorsch.com.br/blog/ondas-do-eletrocardiograma>. Citado na página 14.

NIKOLENKO, S. I. Synthetic data for deep learning. *CoRR*, abs/1909.11512, 2019. Disponível em: <http://arxiv.org/abs/1909.11512>. Citado 2 vezes nas páginas 23 e 24.

REGIS, C.; CALDEIRA, L.; GURJAO, E. Avaliação da amostragem compressiva em sinais de ecg e imagens digitais. *Revista Principia - Divulgação Científica e Tecnológica do IFPB*, v. 1, p. 95, 06 2016. Citado 2 vezes nas páginas 8 e 15.

RIMOLDI, H. J. Diagnosing the diagnostic process. *Medical education*, Wiley Online Library, v. 22, n. 4, p. 270–278, 1988. Citado na página 18.

SENEVIRATNE, S.; HU, Y.; NGUYEN, T.; LAN, G.; KHALIFA, S.; THILAKARATHNA, K.; HASSAN, M.; SENEVIRATNE, A. A survey of wearable devices and challenges. *IEEE Communications Surveys Tutorials*, v. 19, n. 4, p. 2573–2620, 2017. Citado 2 vezes nas páginas 12 e 19.

SHORTLIFFE, E. H. The adolescence of ai in medicine: will the field come of age in the'90s? *Artificial intelligence in medicine*, Elsevier, v. 5, n. 2, p. 93–106, 1993. Citado na página 18.

TURING, A. M. Computing machinery and intelligence. In: *Parsing the turing test*. [S.l.]: Springer, 2009. p. 23–65. Citado na página 16.

VANDERLEI, L. C. M.; PASTRE, C. M.; HOSHI, R. A. A.; CARVALHO, T. D. d.; GODOY, M. F. d. NoÃ§ÃbÃ¡sicas de variabilidade da frequÃcardÃaca e sua aplicabilidade clÃnica. *Brazilian Journal of Cardiovascular Surgery*, scielo, v. 24, p. 205 – 217, 06 2009. ISSN 0102-7638. Citado 2 vezes nas páginas 15 e 27.

7 Código Fonte

7.1 Biblioteca de pré-processamento

```
import numpy as np
from matplotlib import pyplot as plt
from matplotlib.patches import Ellipse
import seaborn as sns
from scipy import signal
from scipy.ndimage import label
from scipy.integrate import trapz
import os

class CustomArray(np.ndarray):
    def __new__(cls, *args, **kwargs):
        return np.asarray(args[0]).view(cls)

    def __getitem__(self, index):
        return np.ndarray.__getitem__(self, index % len(self))

def coleta_dados(path_final):
    dirFiles= os.listdir(path_final)
    array=[]
    for file in dirFiles:
        array_original =np.loadtxt(path_final+"/"+file,dtype=np.int)
        array.append((str(file),array_original))
    return np.array(array,dtype=object)

def origpons(n):
    if n==2:
        P = np.array(([1,1],[1,-1]),np.int32)
    else:
        new_n = n//2
        Q=origpons(new_n)
        P =np.zeros((n,n),np.int32)
        for i in range(0,new_n,2):
            A = Q[i,:]
            B = Q[i+1,:]
            AB=np.concatenate((A,B))
            AmenosB=np.concatenate((A,-B))
            BA=np.concatenate((B,A))
            BmenosA=np.concatenate((-B,A))

            new_vector=np.array([AB,AmenosB,BA,BmenosA])
```

```
41              P[(new_vector.shape[0]//2*i):new_vector.shape[0]+(new_vector.
                    shape[0]//2*i),:]=new_vector
    return P
43
def multipli_matriz(matriz_pons,matriz_entrada,r):
45      multi= np.inner(matriz_pons, matriz_entrada)
    return multi
47def normalize(arr, t_min, t_max):
    norm_arr = []
49      diff = t_max - t_min
    diff_arr = max(arr) - min(arr)
51      for i in arr:
        temp = (((i - min(arr))*diff)/diff_arr) + t_min
53          norm_arr.append(temp)
    return np.array(norm_arr)
55
array_1d = np.arange(1,4)
57range_to_normalize = (0,1)

59def troca(data_frame, index, lowpass, highpass, level=4,model='median',
    debug=False):

61      values = CustomArray(data_frame)
    esquerda_valores = []
63      direita_valores = []
    count_direita = 0
65      count_esquerda = 0
    j = index
67      while (count_direita < level):
        j += 1
69          if ((values[j] > lowpass) & (values[j] < highpass)):
            direita_valores.append(values[j])
71              count_direita += 1

73      j = index
    while (count_esquerda < level):
75          j += -1
        if ((values[j] > lowpass) and (values[j] < highpass)):
77              esquerda_valores.append(values[j])
            count_esquerda += 1
79
    t = np.concatenate([direita_valores, esquerda_valores])
81      if(model=='median'):
        result = np.median(t)
83      elif(model=='mean'):
        result = np.mean(t)
85      if (debug):
```

```
            print("N : [", index, " ]", "R: ", result, " : ", direita_valores,
                esquerda_valores)
87      return result


89
    def sintetiza_RR(array,engine_value=200,value_list=10):
91      v_list=[]
        for x in range(0,value_list):
93          q3, q1 = np.percentile(array, [75, 25])
            iqr = q3 - q1
95          lowpass = q1 - (iqr * 1.5)
            highpass = q3 + (iqr * 1.5)
97          array_swap=array.copy()
            for i in range(0,engine_value):
99            value =np.random.randint(0, len(array)-1)
              array_swap[value] = troca(array_swap, value, lowpass, highpass)
101         v_list.append(array_swap)
        return v_list
103


105 def image_RR(array):
        new_array=[]
107     for x in range(0,len(array)):
            new_array.append(array)
109     return np.array(new_array)


111
    def removeoutlier(values, debug=False):
113     fator = 1.5

115     q3, q1 = np.percentile(values, [75, 25])
        iqr = q3 - q1
117     lowpass = q1 - (iqr * fator)
        highpass = q3 + (iqr * fator)
119     if (debug):
            print("lowpass :", lowpass)
121         print("highpass :", highpass)
        for i in range(0, len(values)):
123         if (debug):
                print("Consultando:[ ", i, " ]", values[i])
125         if values[i] > lowpass and values[i] < highpass:
                pass
127         else:
                if (debug):
129                 print("Removendo: ", values[i])
                values[i] = troca(values, i, lowpass, highpass)
131             if (debug):
```

```
                print("Colocando: ", values[i])
    return np.array(values)

def detect_peaks(ecg_signal, threshold=0.5, qrs_filter=None):
    '''
    Peak detection algorithm using cross corrrelation and threshold
    '''
    if qrs_filter is None:
        # create default qrs filter, which is just a part of the sine
            function
        t = np.linspace(1.5 * np.pi, 3.5 * np.pi, 15)
        qrs_filter = np.sin(t)

    # normalize data
    ecg_signal = (ecg_signal - ecg_signal.mean()) / ecg_signal.std()

    # calculate cross correlation
    similarity = np.correlate(ecg_signal, qrs_filter, mode="same")
    similarity = similarity / np.max(similarity)

    # return peaks (values in ms) using threshold
    return ecg_signal[similarity > threshold].index, similarity

def group_peaks(p, threshold=5):
    output = np.empty(0)

    peak_groups, num_groups = label(np.diff(p) < threshold)

    for i in np.unique(peak_groups)[1:]:
        peak_group = p[np.where(peak_groups == i)]
        output = np.append(output, np.median(peak_group))
    return output


def percentil_mov(signal_ECG):
    return np.percentile(signal_ECG, [98])

def get_rr(signal):
    limiar=700
    peaks = signal[signal.ECG > limiar].copy()
    return peaks


def removeoutlier_split(array, corte):
    array = np.array_split(array, corte)
    for array_ in array:
        removeoutlier(array_)
```

```
    return np.concatenate(array)
179
181 def circular_array(idx, starting_index, ending_index):
    idx = np.roll(idx, -starting_index)[:(len(idx) - starting_index +
        ending_index) % len(idx)]
183     return idx
185
    def get_circular(array,value_list=8):
187     array_temp = []
        value_list_group=int(len(array)/value_list)
189     for data_index in range(0,len(array),value_list_group):
            array_swap = []
191         for data in range(0, len(array)):
                array_swap.append(array[(data_index + data) % len(array)])
193         array_temp.append(np.array(array_swap))
        return np.array(array_temp)
195
197 def select_estavel(data_set):
        inicio=80
199     select_windows_value=8
        tamanho=256
201     data_sets_final=[]
        while inicio+tamanho <= len(data_set):
203         flag_stop=True
            select_windows=0
205         media_ant=data_set[inicio]
            while (flag_stop):
207             select_windows=select_windows+select_windows_value
                select_data_swap = data_set[inicio:inicio+select_windows].copy
                    ()
209             deslocamento = 0.1 * media_ant
                media = np.mean(select_data_swap)
211             deslocamento_direita = media_ant + deslocamento
                deslocamento_esquerda = media_ant - deslocamento
213             if(select_windows >= tamanho):
                    flag_stop=False
215                 data_sets_final.append((inicio,select_data_swap))
                if  (media > deslocamento_esquerda) and ( deslocamento_direita
                   > media) :
217                    media_ant=np.mean(select_data_swap)
                else:
219                 flag_stop=False
            inicio=select_windows+inicio
221
```

```
    return np.array(data_sets_final,dtype=object)
223
225 def plot_poincare(rr,save_rr_poincare=None):
    rr_n = rr[:-1]
227 rr_n1 = rr[1:]

229 sd1 = np.sqrt(0.5) * np.std(rr_n1 - rr_n)
    sd2 = np.sqrt(0.5) * np.std(rr_n1 + rr_n)
231
    m = np.mean(rr)
233 min_rr = np.min(rr)
    max_rr = np.max(rr)
235
    plt.figure(figsize=(10, 10))
237 plt.title("Poincare plot")

239 sns.scatterplot(x=rr_n, y=rr_n1, color="#51A6D8")

241 plt.xlabel(r'$RR_n (ms)$')
    plt.ylabel(r'$RR_{n+0} (ms)$')
243
    e1 = Ellipse((m, m), 2*sd1, 2*sd2, angle=-45, linewidth=1.2, fill=False
        , color="k")
245 plt.gca().add_patch(e1)

247 plt.arrow(m, m, (max_rr-min_rr)*0.4, (max_rr-min_rr)*0.4, color="k",
        linewidth=0.8, head_width=5, head_length=5)
    plt.arrow(m, m, (min_rr-max_rr)*0.4, (max_rr-min_rr)*0.4, color="k",
        linewidth=0.8, head_width=5, head_length=5)
249
    plt.arrow(m, m, sd2 * np.sqrt(0.5), sd2 * np.sqrt(0.5), color="green",
        linewidth=5)
251 plt.arrow(m, m, -sd1 * np.sqrt(0.5), sd1 * np.sqrt(0.5), color="red",
        linewidth=5)

253 plt.text(max_rr, max_rr, "SD2", fontsize=20, color="green")
    plt.text(m-(max_rr-min_rr)*0.4-20, max_rr, "SD1", fontsize=20, color="
        red")
255 if save_rr_poincare:
        plt.savefig(save_rr_poincare)
257
    return sd1, sd2
259

261 def timedomain(rr):
    results = {}
```

```
263     hr = 60000 / rr
        results['mean'] = np.mean(rr)
265     results['std'] = np.std(rr)
        results['median'] = np.median(hr)
267     results['max'] = np.max(hr)
        results['rmssd'] = np.sqrt(np.mean(np.square(np.diff(rr)))) #
            relevante
269     results['pnn50'] = np.sum(np.abs(np.diff(rr)) > 50) * 1
        return results
271


273 def frequency_domain(rri, fs=4):
        # Estimate the spectral density using Welch's method
275     fxx, pxx = signal.welch(x=rri, fs=fs)
        '''
277     Segement found frequencies in the bands
         - Very Low Frequency (VLF): 0-0.04Hz
279      - Low Frequency (LF): 0.04-0.15Hz
         - High Frequency (HF): 0.15-0.4Hz
281     '''
        cond_vlf = (fxx >= 0) & (fxx < 0.04)
283     cond_lf = (fxx >= 0.04) & (fxx < 0.15)
        cond_hf = (fxx >= 0.15) & (fxx < 0.4)
285     # calculate power in each band by integrating the spectral density
        vlf = trapz(pxx[cond_vlf], fxx[cond_vlf])
287     lf = trapz(pxx[cond_lf], fxx[cond_lf])
        hf = trapz(pxx[cond_hf], fxx[cond_hf])
289     # sum these up to get total power
        total_power = vlf + lf + hf
291     # find which frequency has the most power in each band
        peak_vlf = fxx[cond_vlf][np.argmax(pxx[cond_vlf])]
293     peak_lf = fxx[cond_lf][np.argmax(pxx[cond_lf])]
        peak_hf = fxx[cond_hf][np.argmax(pxx[cond_hf])]
295     # fraction of lf and hf
        lf_nu = 100 * lf / (lf + hf)
297     hf_nu = 100 * hf / (lf + hf)
        results = {}
299

        results['lf_nu'] = lf_nu
301     results['hf_nu'] = hf_nu
        results['lf_hf'] = (lf / hf) * 10
303     results['vlf'] = vlf / 4
        results['lf'] = lf / 4
305     results['hf'] = hf / 4
        results['total_f'] = total_power / 10
307     results['peak_vlf'] = peak_vlf * 1000
        results['peak_lf'] = peak_lf * 1000
```

```
309     results['peak_hf'] = peak_hf * 1000

311     return results, fxx, pxx

313 def passa_baixa(fft_x,fft_y,frequencia_corte):
        for i in range(fft_x.size):
315         if (fft_x[i] > frequencia_corte): # cut off all frequencies higher
                than 0.005
                fft_y[i] = 0.0
317     return fft_y
```

Treinamento - MEIM

```
1  import tensorflow as tf
   from tensorflow.keras import datasets, layers, models,constraints
3  from sklearn.model_selection import train_test_split
   from sklearn.metrics import classification_report
5  from imblearn.under_sampling import RandomUnderSampler
   import numpy as np
7  from matplotlib import pyplot as plt
   from libs import archanjo as arch
9
   categorias = [0,1]
11 categorias_string = ["Hipertensos_Repouso_65_75_Supino","
       Saudaveis_Repouso_20_30_Supino"]

13 path_base = "/home/rafael/Documentos/FACOM/Mestrado/Materias/Defesa/
       Pesquisa/DSP/DataSet_VFC/New_Set/data/"

15 data_frame_cat = []
   data_frame_numpy= []
17 ponse= arch.origpons(256)
   for x in categorias:
19     for class_data_dict in arch.coleta_dados(path_base + categorias_string[
           x]):
           name_user = class_data_dict[0]
21         data = class_data_dict[1]
           sem_outliers = arch.removeoutlier(data.copy(), debug=False)
23         estacionario = arch.select_estavel(sem_outliers)
           for estac_ in estacionario:
25             data_=[]
               data_frame_cat.append(x)
27             data_time=arch.timedomain(estac_[1])
               result,_,_ = arch.frequency_domain(rri=estac_[1])
29             for dat_ in data_time:
                   data_.append(np.round(data_time[dat_],2))
```

```
            for dat_ in result:
                data_.append(np.round(result[dat_],2))
            data_frame_numpy.append(np.array(data_))
data_frame_cat = np.array(data_frame_cat,dtype=np.int)
data_frame_numpy= np.array(data_frame_numpy,dtype=np.int)

rus = RandomUnderSampler(random_state=120)
train_images_balance_s_temp,train_labels_balance_s_temp = rus.fit_resample(
    data_frame_numpy,data_frame_cat)
train_images_balance_s=[]
train_labels_balance_s =train_labels_balance_s_temp

dimensao=(-1,16,16,1)
train_images_balance_s=train_images_balance_s.reshape(dimensao)
train_images_balance_s=train_images_balance_s/150
print(data_frame_numpy.shape)
print(data_frame_cat.shape)
print(train_images_balance_s.shape)
print(train_labels_balance_s.shape)


train_images_balance, teste_images_predic, train_labels_balance,
    test_labels_predic = train_test_split(train_images_balance_s,
    train_labels_balance_s, shuffle=True, random_state=10, train_size=0.8)

print(train_images_balance.shape)

print(teste_images_predic.shape)


train_images, test_images, train_labels, test_labels = train_test_split(
    train_images_balance,train_labels_balance , shuffle=True, random_state
    =120, train_size=0.8)



print(train_images.shape)

print(test_images.shape)
print(test_labels)
fig ,ax= plt.subplots(1, figsize=(56, 56))
plt.imshow(test_images[3],cmap="gray")

model = models.Sequential()

model.add(layers.Conv2D(16, (3, 3), activation='relu', input_shape=(16, 16,
     1)))
```

```
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(16, (3, 3), activation='relu'))
model.add(layers.Dropout(0.2))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Flatten())
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(2,activation='softmax'))

model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(
    from_logits=True),  metrics=['accuracy'])

history = model.fit(train_images, train_labels, epochs=1000,
                    validation_data=(test_images, test_labels))

y_pred=model.predict(teste_images_predic)
y_pred_max = np.argmax(y_pred, axis=1)
# print(y_pred)

print(y_pred_max)
print(test_labels_predic)
print(classification_report(test_labels_predic, y_pred_max))

plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')

test_loss, test_acc = model.evaluate(teste_images_predic,
   test_labels_predic, verbose=2)
print(test_acc)
```

Treinamento - MEO

```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models,constraints

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imblearn.under_sampling import RandomUnderSampler
import numpy as np
from matplotlib import pyplot as plt
from libs import archanjo as arch
```

```
categorias = [0,1]
categorias_string = ["Hipertensos_Repouso_65_75_Supino","Saudaveis_Repouso_20_30_Supino"]

path_base = "/home/rafael/Documentos/FACOM/Mestrado/Materias/Defesa/Pesquisa/DSP/DataSet_VFC/New_Set/data/"

data_frame_cat = []
data_frame_numpy= []

for x in categorias:
    for class_data_dict in arch.coleta_dados(path_base + categorias_string[x]):
        name_user = class_data_dict[0]
        data = class_data_dict[1]
        sem_outliers = arch.removeoutlier(data.copy(), debug=False)
        estacionario = arch.select_estavel(sem_outliers)

        for data_ in estacionario:
            data_frame_cat.append(x)
            array_normalizado = arch.normalize(data_[1],0,255).astype(np.uint8)
            data_frame_numpy.append(array_normalizado)

data_frame_cat = np.array(data_frame_cat,dtype=np.int)
data_frame_numpy= np.array(data_frame_numpy,dtype=np.int)

rus = RandomUnderSampler(random_state=130,sampling_strategy = "majority"

)
train_images_balance_s,train_labels_balance_s, = rus.fit_resample(data_frame_numpy,data_frame_cat)


train_images_balance, teste_images_predic, train_labels_balance, test_labels_predic = train_test_split(train_images_balance_s, train_labels_balance_s, shuffle=True, random_state=10, train_size=0.8)


print(train_labels_balance)
print(test_labels_predic)

print(train_labels_balance.shape)
print(test_labels_predic.shape)
data_x=[]
data_y=[]
value_list=64
```

```
for index_data in range(0,len(train_images_balance)):
    print(index_data)
    plot_list=arch.sintetiza_RR(train_images_balance[index_data],value_list
        =value_list,engine_value=100)
    for d_x in plot_list:
        list_ciruclar=arch.get_circular(np.array(d_x))
        for ar_cl in list_ciruclar :
            data_=[]
            for x in range(0,256):
                data_.append(ar_cl)
            data_x.append(np.array(data_))
            data_y.append(train_labels_balance[index_data])

train_labels_balance = np.array(data_y,dtype=np.int)
train_images_balance = np.array(data_x,dtype=np.int)

print(train_labels_balance)
print(train_images_balance.shape)
print(train_labels_balance.shape)

dimensao=(-1,256,256,1)
train_images_balance_2d=train_images_balance.reshape(dimensao)/255

teste_images_predic=teste_images_predic.reshape(dimensao)/255

print(teste_images_predic.shape)
print(test_labels_predic.shape)
print(train_images_balance.shape)
print(train_labels_balance.shape)
train_images, test_images, train_labels, test_labels = train_test_split(
    train_images_balance,train_labels_balance , shuffle=True, random_state
    =120, train_size=0.8)

print(train_labels)
print(test_labels)
print(train_images.shape)
print(test_images.shape)
model = models.Sequential()
model.add(layers.Conv2D(32, (5, 5), activation='relu', input_shape=(16, 16,
    1)))
model.add(layers.Conv2D(32, (5, 5), activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Conv2D(64, (5, 5), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(2,activation='softmax'))
```

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(
    from_logits=True),  metrics=['accuracy'])

history = model.fit(train_images, train_labels, epochs=5,
                    validation_data=(test_images, test_labels))
y_pred=model.predict(teste_images_predic)
y_pred_max = np.argmax(y_pred, axis=1)
print(y_pred)
print(y_pred_max)
print(test_labels_predic)
print(classification_report(test_labels_predic, y_pred_max))
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')
test_loss, test_acc = model.evaluate(teste_images_predic,
    test_labels_predic, verbose=2)
print(test_acc)
```

Treinamento - MEP

```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models,constraints
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imblearn.under_sampling import RandomUnderSampler
import numpy as np
from matplotlib import pyplot as plt
from libs import archanjo as arch

categorias = [0,1]
categorias_string = ["Hipertensos_Repouso_65_75_Supino","
    Saudaveis_Repouso_20_30_Supino"]
path_base = "/home/rafael/Documentos/FACOM/Mestrado/Materias/Defesa/
    Pesquisa/DSP/DataSet_VFC/New_Set/data/"
data_frame_cat = []
data_frame_numpy= []
ponse= arch.origpons(256)
for x in categorias:
    for class_data_dict in arch.coleta_dados(path_base + categorias_string[
        x]):
        name_user = class_data_dict[0]
        data = class_data_dict[1]
```

```
20          sem_outliers = arch.removeoutlier(data.copy(), debug=False)
            estacionario = arch.select_estavel(sem_outliers)
22          for estac_ in estacionario:
                data_=[]
24              data_frame_cat.append(x)
                data_frame_numpy.append(estac_[1])
26
data_frame_cat = np.array(data_frame_cat,dtype=np.int)
28 data_frame_numpy= np.array(data_frame_numpy,dtype=np.int)
rus = RandomUnderSampler(random_state=120)
30 train_images_balance_s_temp,train_labels_balance_s_temp = rus.fit_resample(
    data_frame_numpy,data_frame_cat)
train_images_balance_s=[]
32 train_labels_balance_s =train_labels_balance_s_temp

34 for data in train_images_balance_s_temp:
    data_=[]
36  for x in range(0,256):
        data_.append(data)
38  ponse_d=arch.multipli_matriz(ponse,np.array(data_),256)

40  train_images_balance_s.append(np.abs(np.round(ponse_d,2)))
train_images_balance_s=np.array(train_images_balance_s)
42 dimensao=(-1,256,256,1)
train_images_balance_s=train_images_balance_s.reshape(dimensao)
44 train_images_balance_s=train_images_balance_s/1500
print(data_frame_numpy.shape)
46 print(data_frame_cat.shape)
print(train_images_balance_s.shape)
48 print(train_labels_balance_s.shape)
train_images_balance, teste_images_predic, train_labels_balance,
    test_labels_predic = train_test_split(train_images_balance_s,
    train_labels_balance_s, shuffle=True, random_state=10, train_size=0.8)
50 print(train_images_balance.shape)
print(teste_images_predic.shape)
52 train_images, test_images, train_labels, test_labels = train_test_split(
    train_images_balance,train_labels_balance , shuffle=True, random_state
    =120, train_size=0.8)
print(train_images.shape)
54 print(test_images.shape)
print(test_labels)
56 fig ,ax= plt.subplots(1, figsize=(56, 56))
plt.imshow(test_images[0],cmap="gray")
58 model = models.Sequential()
model.add(layers.Conv2D(16, (3, 3), activation='relu', input_shape=(256,
    256, 1)))
60 model.add(layers.MaxPooling2D((2, 2)))
```

```
model.add(layers.Conv2D(16, (3, 3), activation='relu'))
model.add(layers.Dropout(0.2))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(2,activation='softmax'))
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(
    from_logits=True),  metrics=['accuracy'])
history = model.fit(train_images, train_labels, epochs=30,
                    validation_data=(test_images, test_labels))
y_pred=model.predict(teste_images_predic)
y_pred_max = np.argmax(y_pred, axis=1)
print(y_pred_max)
print(test_labels_predic)
print(classification_report(test_labels_predic, y_pred_max))
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')
```

Treinamento - MLP

```
import tensorflow as tf
from matplotlib.pyplot import figure
from tensorflow.keras import datasets, layers, models,constraints
import numpy as np
import seaborn as  sns
import matplotlib.pyplot as plt
from scipy import interpolate
import pandas as pd
from matplotlib import pyplot as plt
from libs import archanjo as arch
import cv2
import matplotlib.pyplot as plt
import os
import cv2
# import cv
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imblearn.under_sampling import RandomUnderSampler
categorias = [0,1]
categorias_string = ["Hipertensos_Repouso_65_75_Supino","
```

```
    Saudaveis_Repouso_20_30_Supino"]

path_base = "/home/rafael/Documentos/FACOM/Mestrado/Materias/Defesa/
    Pesquisa/DSP/DataSet_VFC/New_Set/data/"

data_frame_cat = []
data_frame_numpy= []

for x in categorias:
    for class_data_dict in arch.coleta_dados(path_base + categorias_string[
        x]):
        name_user = class_data_dict[0]
        data = class_data_dict[1]
        sem_outliers = arch.removeoutlier(data.copy(), debug=False)
        estacionario = arch.select_estavel(sem_outliers)

        for data_ in estacionario:
            data_frame_cat.append(x)
            data_frame_numpy.append(data_[1])

data_frame_cat = np.array(data_frame_cat,dtype=np.int)
data_frame_numpy= np.array(data_frame_numpy,dtype=np.int)
rus = RandomUnderSampler(random_state=130,sampling_strategy = "majority"

)
train_images_balance_s,train_labels_balance_s, = rus.fit_resample(
    data_frame_numpy,data_frame_cat)
autopct = "%.2f"

fig, axs = plt.subplots(ncols=2, figsize=(10, 5))
axs[0].set_title("Original")
pd.Series(name='Pacientes',data=data_frame_cat).value_counts().plot.pie(
    autopct=autopct, ax=axs[0])

pd.Series(name='Pacientes',data=train_labels_balance_s).value_counts().plot
    .pie( autopct=autopct,ax=axs[1])

axs[1].set_title("Under-sampling")
fig.tight_layout()

train_images_balance, teste_images_predic, train_labels_balance,
    test_labels_predic = train_test_split(train_images_balance_s,
    train_labels_balance_s, shuffle=True, random_state=123, train_size=0.7
      )
print(train_labels_balance)
print(test_labels_predic)
```

```
print(train_labels_balance.shape)
print(test_labels_predic.shape)
data_x=[]
data_y=[]
value_list=64
for index_data in range(0,len(train_images_balance)):
    print(index_data)
    plot_list=arch.sintetiza_RR(train_images_balance[index_data],value_list
        =value_list,engine_value=100)
    for d_x in plot_list:
        # print("add")
        list_ciruclar=arch.get_circular(np.array(d_x))
        for ar_cl in list_ciruclar:
            fs = 4
            x= np.linspace(0, len(ar_cl)-1, num=len(ar_cl), endpoint=True)
            f = interpolate.interp1d(x, ar_cl, kind='cubic')

            xnew = np.linspace(0, len(ar_cl)-1,num= len(ar_cl)*fs, endpoint
                =True)
            rr_interpolated = f(xnew)
            _,transform_fourier_f,transform_fourier_magnitude = arch.
                frequency_domain(rr_interpolated,fs=4)
            psd_f = interpolate.interp1d(transform_fourier_f,
                transform_fourier_magnitude)
            data_x.append(arch.normalize(psd_f(np.linspace(0.0,0.5,256))
                ,0,1))
            data_y.append(train_labels_balance[index_data])

train_images_balance = np.array(data_x)
train_labels_balance = np.array(data_y)
data_x=[]
data_y=[]
value_list=64
for index_data in range(0,len(teste_images_predic)):
    print(index_data)
    fs = 4
    rr_manual=teste_images_predic[index_data]
    x= np.linspace(0, len(rr_manual)-1, num=len(rr_manual), endpoint=True)
    f = interpolate.interp1d(x, rr_manual, kind='cubic')
    xnew = np.linspace(0, len(rr_manual)-1,num= len(rr_manual)*fs, endpoint
        =True)
    rr_interpolated = f(xnew)
    _,transform_fourier_f,transform_fourier_magnitude = arch.
        frequency_domain(rr_interpolated,fs=4)
    psd_f = interpolate.interp1d(transform_fourier_f,
        transform_fourier_magnitude)
    data_x.append(arch.normalize(psd_f(np.linspace(0.0,0.5,256)),0,1))
```

```
    data_y.append(test_labels_predic[index_data])
teste_images_predic = np.array(data_x)
test_labels_predic = np.array(data_y)
train_images, test_images, train_labels, test_labels = train_test_split(
    train_images_balance,train_labels_balance , shuffle=True, random_state
    =312, train_size=0.7)
train_images_conv1d =train_images.reshape(-1,16,16,1)
test_images_conv1d =test_images.reshape(-1,16,16,1)
teste_images_predic_conv1d=teste_images_predic.reshape(-1,16,16,1)
model = models.Sequential()
model.add(layers.Conv2D(16, (3, 3), activation='relu', input_shape=(16, 16,
     1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(16, (3, 3), activation='relu'))
model.add(layers.Dropout(0.2))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(2,activation='softmax'))

batch_size=129, verbose=1)

model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(
    from_logits=True),  metrics=['accuracy'])
#
history = model.fit(train_images_conv1d, train_labels, epochs=10,
                    validation_data=(test_images_conv1d, test_labels))
```

Treinamento - MO

```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models,constraints
import pandas as pd
from libs import archanjo as arch
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imblearn.under_sampling import RandomUnderSampler
categorias = [0,1]
categorias_string = ["Hipertensos_Repouso_65_75_Supino","
    Saudaveis_Repouso_20_30_Supino"]

path_base = "/home/rafael/Documentos/FACOM/Mestrado/Materias/Defesa/
    Pesquisa/DSP/DataSet_VFC/New_Set/data/"
```

```
14
data_frame_cat = []
16 data_frame_numpy= []

18 for x in categorias:
    for class_data_dict in arch.coleta_dados(path_base + categorias_string[
       x]):
20      name_user = class_data_dict[0]
        data = class_data_dict[1]
22      sem_outliers = arch.removeoutlier(data.copy(), debug=False)
        estacionario = arch.select_estavel(sem_outliers)
24
        for data_ in estacionario:
26          data_frame_cat.append(x)
            array_normalizado = arch.normalize(data_[1],0,255).astype(np.
               uint8)
28          data_frame_numpy.append(array_normalizado)

30 data_frame_cat = np.array(data_frame_cat,dtype=np.int)
data_frame_numpy= np.array(data_frame_numpy,dtype=np.int)
32


34 rus = RandomUnderSampler(random_state=130,sampling_strategy = "majority"

36 )
train_images_balance_s,train_labels_balance_s, = rus.fit_resample(
   data_frame_numpy,data_frame_cat)
38 autopct = "%.2f"

40 fig, axs = plt.subplots(ncols=2, figsize=(10, 5))
axs[0].set_title("Original")
42 pd.Series(name='Pacientes',data=data_frame_cat).value_counts().plot.pie(
   autopct=autopct, ax=axs[0])
pd.Series(name='Pacientes',data=train_labels_balance_s).value_counts().plot
   .pie(autopct=autopct,ax=axs[1])
44 axs[1].set_title("Under-sampling")
fig.tight_layout()
46 train_images_balance, teste_images_predic, train_labels_balance,
   test_labels_predic = train_test_split(train_images_balance_s,
   train_labels_balance_s, shuffle=True, random_state=10, train_size=0.8)
print(train_labels_balance)
48 print(test_labels_predic)
print(train_labels_balance.shape)
50 print(test_labels_predic.shape)
data_x=[]
52 data_y=[]
value_list=10
```

```
54
for index_data in range(0,len(train_images_balance)):
56      print(index_data)
        plot_list=arch.sintetiza_RR(train_images_balance[index_data],value_list
            =value_list,engine_value=100)
58      for d_x in plot_list:
            list_ciruclar=arch.get_circular(np.array(d_x))
60          for ar_cl in list_ciruclar:
                data_=[]
62              for x in range(0,256):
                    data_.append(ar_cl)
64              data_x.append(np.array(data_))
                data_y.append(train_labels_balance[index_data])
66
print("fim")
68 data_x=[]
data_y=[]
70 i=0
for ar_cl in teste_images_predic:
72      data_=[]
        for x in range(0,256):
74          data_.append(ar_cl)
        data_x.append(np.array(data_))
76      data_y.append(test_labels_predic[i])
        i=+1
78 teste_images_predic = np.array(data_x,dtype=np.int)
dimensao=(-1,256,256,1)
80 train_images_balance=train_images_balance.reshape(dimensao)/255
teste_images_predic=teste_images_predic.reshape(dimensao)/255
82 print(teste_images_predic.shape)
print(test_labels_predic.shape)
84 print(train_images_balance.shape)
print(train_labels_balance.shape
86 train_images, test_images, train_labels, test_labels = train_test_split(
    train_images_balance,train_labels_balance , shuffle=True, random_state
    =120, train_size=0.8)
model = models.Sequential()
88 model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(256,
    256, 1)))
model.add(layers.MaxPooling2D((2, 2)))
90 model.add(layers.Conv2D(32, (3, 3), activation='relu'))
model.add(layers.Dropout(0.5))
92 model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
94 model.add(layers.Dense(2,activation='softmax'))
model.compile(optimizer='adam',
96                  loss=tf.keras.losses.SparseCategoricalCrossentropy(
```

```
    from_logits=True),  metrics=['accuracy'])

history = model.fit(train_images, train_labels, epochs=5,
                    validation_data=(test_images, test_labels))
y_pred=model.predict(teste_images_predic)
y_pred_max = np.argmax(y_pred, axis=1)
print(y_pred)
print(y_pred_max)
print(test_labels_predic)
print(classification_report(test_labels_predic, y_pred_max))
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')
```

Treinamento - MOP

```

categorias = [0,1]
categorias_string = ["Hipertensos_Repouso_65_75_Supino","
    Saudaveis_Repouso_20_30_Supino"]

path_base = "/home/rafael/Documentos/FACOM/Mestrado/Materias/Defesa/
    Pesquisa/DSP/DataSet_VFC/New_Set/data/"


data_frame_cat = []
data_frame_numpy= []

for x in categorias:
    for class_data_dict in arch.coleta_dados(path_base + categorias_string[
        x]):
        name_user = class_data_dict[0]
        data = class_data_dict[1]
        sem_outliers = arch.removeoutlier(data.copy(), debug=False)
        estacionario = arch.select_estavel(sem_outliers)
        for data_ in estacionario:
            data_frame_cat.append(x)
            data_frame_numpy.append(data_[1])


data_frame_cat = np.array(data_frame_cat,dtype=np.int)
data_frame_numpy= np.array(data_frame_numpy,dtype=np.int)
rus = RandomUnderSampler(random_state=120)
```

```
train_images_balance_s_temp,train_labels_balance_s_temp = rus.fit_resample(
    data_frame_numpy,data_frame_cat)
train_labels_balance_s =train_labels_balance_s_temp

dimensao=(-1,16,16)
train_images_balance_s_temp=train_images_balance_s_temp.reshape(dimensao)

ponse=arch.origpons(16)
train_images_balance_s=[]
for data in train_images_balance_s_temp:
    ponse_d=arch.multipli_matriz(ponse,data,16)
    train_images_balance_s.append(np.abs(np.round(ponse_d,2)))
train_images_balance_s=np.array(train_images_balance_s)
dimensao=(-1,16,16,1)
train_images_balance_s=train_images_balance_s.reshape(dimensao)
train_images_balance_s=np.round(train_images_balance_s/1500,2)

train_images_balance, teste_images_predic, train_labels_balance,
    test_labels_predic = train_test_split(train_images_balance_s,
    train_labels_balance_s, shuffle=True, random_state=10, train_size=0.8)

train_images, test_images, train_labels, test_labels = train_test_split(
    train_images_balance,train_labels_balance , shuffle=True, random_state
    =120, train_size=0.8)
model = models.Sequential()

model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(16, 16,
    1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(16, (3, 3), activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(2,activation='softmax'))
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(
    from_logits=True),  metrics=['accuracy'])



history = model.fit(train_images, train_labels, epochs=1000,
                    validation_data=(test_images, test_labels))
y_pred=model.predict(teste_images_predic)
y_pred_max = np.argmax(y_pred, axis=1)
print(y_pred)
print(y_pred_max)
```

```
print(test_labels_predic)
print(classification_report(test_labels_predic, y_pred_max))
```

Código Web - Back-End

```
from flask import Blueprint, request, jsonify
from models.users import fetch
import numpy as np
from scipy import signal
from wavelets import Ricker
import os
# import cv2
import sys
import base64
import re
import codecs
import io
from datetime import datetime
from io import BytesIO
import json
import skimage
from PIL import Image
import requests
try:
    from StringIO import StringIO
except ImportError:
    from io import StringIO

from numpy import genfromtxt

blueprint = Blueprint("api", __name__)

@blueprint.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run()


def salvar_registro(name,tipo):
    data = {}
    data["name"]=name
    data["api"]=tipo
    data["data"]=str(datetime.timestamp(datetime.now()))
```

```
42      json_file= read_logs_registro()
        print('Lido: ', json_file, file=sys.stderr)
44
        save_logs_registro(json_file,data)
46
    def save_logs_registro(json_file,inf_log):
48      with open('/home/rafael/Documentos/Projetos/Meus/CNN/cnnwebv2/back-end/
            arquivo_de_log.json', 'w') as outfile:
            print('Log atual: ', json_file, file=sys.stderr)
50          json_file['logs'].append(inf_log)
            print('Log atual: ', json_file, file=sys.stderr)
52          json.dump(json_file, outfile)


54
    @blueprint.route('/logs', methods=("GET", ))
56  def read_logs_registro():
        with open('/home/rafael/Documentos/Projetos/Meus/CNN/cnnwebv2/back-end/
            arquivo_de_log.json', 'r') as outfile:
58          return json.load(outfile)


60
    def readb64(base64_string):
62      playload = re.sub(
            r"^data:image\/(?:gif|png|jpeg|bmp|webp)(?:;charset=utf-8)?;base64,
                ", "", base64_string)
64      sbuf = StringIO()
        img = base64.b64decode(playload)
66      img = Image.open(io.BytesIO(img)).convert('L')

68      img = np.array(img,dtype=np.uint8)

70      print('LOG: ',img.shape, file=sys.stderr)

72      return img

74  def rr_to_bpm(frequencies):
        for i in range(len(frequencies)):
76          if frequencies[i] != 0:
                frequencies[i] = (frequencies[i]**(-1))*1000*60
78      return frequencies


80
    def moving_median(array):
82      initiation_window = 0
        end_window = janela = 9  # janela definida como 9
84      while initiation_window < (array.size-janela):
            window = array[initiation_window: end_window]
```

```
86          median_window = np.median(window)
            if array[initiation_window] > median_window + 200:  # threshold de
                200
88              array[initiation_window] = int(median_window)
            elif array[initiation_window] < median_window - 200:  # threshold
                de 200
90              array[initiation_window] = int(median_window)
            initiation_window = initiation_window + 1
92          end_window = end_window + 1
        return array
94


96  def image(array):
        axis_x = np.linspace(0, len(array)-1, len(array))
98      axis_y = array
        return axis_x.tolist(), axis_y.tolist()
100


102 def recebe_arquivo(text):
        array = []
104     text = re.split("\r\n|\r|\n", text)
        for line in text:
106         if len(line) != 0:
                array.append(int(line))
108     array = np.asarray(array, dtype=np.intc)
        return array
110


112 def Filtro(array):
        if(len(array) >= 348):
114         filtered = moving_median(array)
            for i in range(348, len(filtered)):
116             filtered = np.delete(filtered, 348)
        return filtered
118


120 def Wavelet(array):
        frequenciesWavalet = []
122     for line in array:
            frequenciesWavalet.append(int(line))
124     bpm_values = rr_to_bpm(frequenciesWavalet)
        bpm_np = np.array(bpm_values)
126     widths = np.arange(1, 31)
        MatrWavelet = signal.cwt(bpm_np, signal.ricker, widths)
128     return MatrWavelet.tolist()


130
```

```
def tratamentoMatrizWavelet(Matrix):
    xtt = []
    xtt.append(Matrix)
    xg = np.array(xtt)
    xg = xg.astype('float32')/371.041
    return xg.reshape(-1, 30, 348, 1)


def VFC(data):
    frequencies = recebe_arquivo(data['conteudoArquivo'])
    Plot_Original_x_axis, Plot_Original_y_axis = image(frequencies)

    filtered = Filtro(frequencies)

    Plot_Filtrada_x_axis, Plot_Filtrada_y_axis = image(filtered)

    Matr_Wavelet = Wavelet(filtered)

    xg = tratamentoMatrizWavelet(Matr_Wavelet)

    payload = {
        "instances": xg.tolist()

    }

    r = requests.post(
        'http://localhost:8501/v1/models/vfc:predict', json=payload)
    pred = json.loads(r.content.decode('utf-8'))
    # print('LOG: '+'Request:',pred, file=sys.stderr)
    # print('LOG: '+'Request:',     model.predict(xg), file=sys.stderr)

    msg = {
        "Saudavel": "%.2f" % (pred["predictions"][0][2]*100),
        "Hipertensao": "%.2f" % (pred["predictions"][0][1]*100),
        "Arritmia": "%.2f" % (pred["predictions"][0][0]*100),
        "Plot_Original_x_axis": Plot_Original_x_axis,
        "Plot_Original_y_axis": Plot_Original_y_axis,
        "Plot_Filtrada_x_axis": Plot_Filtrada_x_axis,
        "Plot_Filtrada_y_axis": Plot_Filtrada_y_axis,
        "Plot_Wavelet": Matr_Wavelet,
        "Erro": False
    }
    return msg


def readb64(base64_string):
     playload = re.sub(
```

```
178         r"^data:image\/(?:gif|png|jpeg|bmp|webp)(?:;charset=utf-8)?;base64
                ,", "", base64_string)
        sbuf = StringIO()
180     img = base64.b64decode(playload)
        img = Image.open(io.BytesIO(img)).convert('L')
182
        img = np.array(img,dtype=np.uint8)
184
        print('LOG: ',img.shape, file=sys.stderr)
186
        return img
188



190


192 def equalizar_imagem(img):
        img = cv2.equalizeHist(img)
194     return img


196
 def encode(image):
198
    # convert image to bytes
200     with BytesIO() as output_bytes:
            PIL_image = Image.fromarray(skimage.img_as_ubyte(image))
202         PIL_image.save(output_bytes, 'JPEG') # Note JPG is not a vaild
                type here
            bytes_data = output_bytes.getvalue()
204
    # encode bytes to base64 string
206     base64_str = ""+str(base64.b64encode(bytes_data
            ), 'utf-8')
   # return base64_str
208
def decodifica_imagen_base(image_base_64):
210     img_new = readb64(image_base_64)

212     img = np.resize(img_new, (256, 256))

214     imagem_equalizada = equalizar_imagem(img)

216     imagem_equalizada_predicao = np.reshape(imagem_equalizada, (-1, 256,
            256, 1))

218     axis_x = np.linspace(0, 255, 256)

220     normalize_histograma = np.histogram(imagem_equalizada.ravel(), bins=
```

```
        axis_x)
    image_histograma = np.histogram(img.ravel(), bins=axis_x)

    # print('LOG: ',normalize_histograma, file=sys.stderr)
    # print('LOG: ',normalize_histograma, file=sys.stderr)

@blueprint.route('/api', methods=("POST", ))
def api():
    data = request.get_json(force=True)
    salvar_registro(data['name'],data['tipo'])

    print('LOG: ', data['tipo'], file=sys.stderr)

    # if(data['tipo'] == "VFC"):
    return jsonify(VFC(data))
from flask import Flask, Blueprint, request, render_template
from config.index import Config
from app import api
from lib.db import db
from .exceptions import ExceptionHandler
from app.errors import blueprint
from lib import cache
from flask_cors import CORS


def create_app(config_object=Config):
    """ Factory function to start application  """
    app = Flask(__name__)
    CORS(app)
    app.url_map.strict_slashes = False
    app.config.from_object(config_object)
    db.init_app(app)
    cache.init_app(app)
    register_blueprints(app)
    register_error_handler(app)
    return app


def register_blueprints(app):
    app.register_blueprint(api.routes.blueprint)
    app.register_blueprint(blueprint)



def register_error_handler(app):
    """ Register function for handling errors  """
    def errorhandler(error):
```

```
        response = error.to_json()
268     response.status_code = error.status_code
        print(response.status_code)
270     return response

272 app.errorhandler(ExceptionHandler)(errorhandler)
```

Exportação Modelo

```
import tensorflow as tf
2
tf.keras.backend.set_learning_phase(0)  # Ignore dropout at inference
4 model = tf.keras.models.load_model('/home/rafael/Documentos/DSP/RR/
    Statistica/CNN-HRV/best_model.h5')
export_path = './tensorflow_models/vfc-rv/1'
6
with tf.keras.backend.get_session() as sess:
8     tf.saved_model.simple_save(
        sess,
10      export_path,
        inputs={'sensor': model.input},
12      outputs={t.name: t for t in model.outputs})
```

Exportação Modelo

```
import tensorflow as tf
2
tf.keras.backend.set_learning_phase(0)  # Ignore dropout at inference
4 model = tf.keras.models.load_model('/home/rafael/Documentos/DSP/RR/
    Statistica/CNN-HRV/best_model.h5')
export_path = './tensorflow_models/vfc-rv/1'
6
with tf.keras.backend.get_session() as sess:
8     tf.saved_model.simple_save(
        sess,
10      export_path,
        inputs={'sensor': model.input},
12      outputs={t.name: t for t in model.outputs})
```

Página Web - Processamento

```
<app-page-header [heading]="'Processamento de exames via convolutional
    neural network (CNN) [EM DESENVOLVIMENTO]'"
2   [icon]="'fa-desktop'"></app-page-header>
```

```
4
<ng-template #customLoadingTemplate>
6   <div class="custom-class">
      <h3>
8       Sua requisi  o foi encaminhada aguarde.....!
      </h3>
10    <button>
        Click me!
12    </button>
    </div>
14 </ng-template>
<mat-card>
16
    <h4>Como utilizar o m dulo Processamento CNN</h4>
18
    <div>
20    <ol>
        <li>Clique em "tipo de diagn stico" para selecionar o tipo de exame
            .</li>
22      <li>Clique em "arquivo de entrada" para escolher uma imagem/arquivo
            para realizar o processamento.</li>
        <ul>
24        <li>Clique em "teste" caso n o possua imagem/arquivo para
              processar.</li>
          <li>Clique em "selecionar" para seleciona-la.</li>
26        <li>Pode-se mudar de imagem/arquivo teste clicando em Pr xima/
              Anterior.</li>
        </ul>
28      <li>Ap s selecionar uma imagem/arquivo clique em "avaliar".</li>
      </ol>
30  </div>
</mat-card>
32 <hr>


34


36 <div class="loading-container">
    <div class="loading">
38
      <ngx-loading [show]="loading_value"
40      [config]="{primaryColour: primaryColour, secondaryColour:
            secondaryColour, tertiaryColour: primaryColour,
            backdropBorderRadius: '3px'}"></ngx-loading>

42    <mat-card>
        <mat-card-actions>
```

```
44          <div class="exemple-container">
              <div class="row"  *ngIf="uploadForm.controls['diagnostico'].value
                  .tipo=='Tracking EPI'">
46              <div class="col-sm">
                  <div align="center">
48                <button  mat-raised-button color="warn"
                    (click)="demonstracao=!demonstracao">Visualizar Video de
                        Demonstra  o </button>
50                </div>
                </div>
52            </div>
            </div>
54          </mat-card-actions>
        </mat-card>
56      <mat-card>
          <mat-card *ngIf="demonstracao" >
58          <div align="center">
             <H2>V deo de Demonstra  o do Tracking de Equipamento Individual
                  de Seguran a - EPI</H2>
60          </div>
            <mat-card-content class="demostracao">
62            <iframe src="https://drive.google.com/file/d/153
                  H6yJQmlf6ryFNbyxy8MKo6tO1NFfqH/preview" width="100%" height="
                  100%"></iframe>
            </mat-card-content>
64
            <mat-card-actions>
66            <div align="center">
              <button  mat-raised-button color="primary"
68            (click)="demonstracao=!demonstracao">Voltar Para o Processamento
                  </button>
            </div>
70          </mat-card-actions>
          </mat-card>
72


74        <form [formGroup]="uploadForm" (ngSubmit)="submit()" *ngIf="!
              demonstracao">
            <div class="row">
76            <div class="col-sm">
                <mat-card class='matCard'>
78                <div align="center">
                    <mat-card-title>Dados de Entrada</mat-card-title>
80                </div>
                  <hr>
82                <mat-card-content>
                    <div class="exemple-container">
```

```
84                    <div class="row">
                        <p> Tipo de processamento: 
86
                          <div *ngIf="uploadForm.controls['diagnostico'].valid"
                              >
88                          {{uploadForm.controls['diagnostico'].value.tipo}}</
                              div>
                    </div>
90
                    <div class="row">
92
                      <p>Formatos de entrada: 
94                      <div *ngIf="uploadForm.controls['diagnostico'].valid"
                            >
                          <p>{{uploadForm.controls['diagnostico'].value.shape
                              }}
96                      </div>
                    </div>
98
                    <div class="row">
100                   <p>Arquivo selecionado: 
                        <div *ngIf="file ">
102                       {{file.name}}
                        </div>
104                   </div>
                  </div>
106
                  <hr>
108
                  <div class="exemmple-container">
110                   <div class="row">
                        <div class="col-12">
112                       <div align="center">
                            <mat-label>Tipo de Diag stico</mat-label>
114                       </div>
                          <mat-select formControlName="diagnostico" placeholder
                              ="Selecione">
116                         <mat-option (click)='checkFile()' *ngFor="let d of
                                diagnosticos" [value]="d">
                              {{d.tipo}}
118                         </mat-option>
                          </mat-select>
120                       <hr>
                        </div>
122                   </div>
                    </div>
124               </mat-card-content>
```

```
126             </mat-card>
            </div>
128
130
132
            <div class="col-sm " *ngIf="mostrar_vfc">
134           <mat-card class='matCard'>
                <div align="center">
136               <mat-card-title>Arquivo VFC {{arquivo_vfc_atual_index}}</
                      mat-card-title>

138             </div>
                <hr>
140             <mat-card-content>

142               <textarea name="text" disabled class="imagePreview_VFC"
                      matInput [value]="arquivo_vfc_atual"></textarea>
                </mat-card-content>
144             <hr>
                <mat-card-footer>
146
                  <div class="container">
148                 <div class="row">
                      <div class="col-sm">
150                     <button [disabled]="!uploadForm.controls['diagnostico
                            '].valid" type="button" mat-raised-button
                          (click)="anterior_vfc()">Anterior</button>
152                   </div>
                      <div class="col-sm">
154                     <button [disabled]="!uploadForm.controls['diagnostico
                            '].valid" type="button" mat-raised-button
                          (click)="proximo_vfc()">Pr ximo</button>
156                   </div>
                      <div class="col-sm">
158                     <button [disabled]="!uploadForm.controls['diagnostico
                            '].valid" type="button" mat-raised-button
                          (click)="select_vfc(arquivo_vfc_atual)">Selecionar
                              </button>
160                   </div>
                    </div>
162               </div>
                </mat-card-footer>
164           </mat-card>
```

```
166
            </div>
168
170
172
174          <div class="col-sm " *ngIf="uploadForm.controls['valido'].value "
                 >
               <mat-card class='matCard'>
176              <div align="center">
                   <mat-card-title>Preview Dados de Entrada</mat-card-title>
178              </div>
                 <hr>
180
                 <mat-card-content>
182                <div class="exemple-container">
                     <div aling="center" class="imagePreview" *ngIf="
                         uploadForm.controls['diagnostico'].value.tipo=='VFC'">
184                    <textarea disabled class="imagePreview" matInput
                         [value]="uploadForm.controls['dados_input'].value"></
                             textarea>
186                  </div>
                   </div>
188              </mat-card-content>
                 <hr>
190              <mat-card-footer>
                 </mat-card-footer>
192            </mat-card>
             </div>
194
         </div>
196    </form>
       <div class="row">
198      <br>
       </div>
200    <div
         *ngIf="uploadForm.controls['dados_input'].value !=''&&uploadForm.
             controls['diagnostico'].value.tipo=='VFC' && show_vfc ">
202      <div class="row">
           <div class="col-sm">
204          <mat-card class='matCardGrafico'>
               <plotly-plot [useResizeHandler]="true" [style]="{position: '
                   relative', width: '100%', height: '100%'}"
206              [data]="graph1.data" [layout]="graph1.layout"></plotly-plot
                   >
```

```
            </mat-card>
          </div>
          <div class="col-sm">
            <mat-card class="matCardGrafico">
              <plotly-plot [useResizeHandler]="true" [style]="{position: '
                  relative', width: '100%', height: '100%'}"
                [data]="graph2.data" [layout]="graph2.layout"></plotly-plot
                  >
            </mat-card>
          </div>


          <div class="col-sm">
            <mat-card class='matCardGrafico'>
              <plotly-plot [useResizeHandler]="true" [style]="{position: '
                  relative', width: '100%', height: '100%'}"
                [data]="graph3.data" [layout]="graph3.layout"></plotly-plot
                  >
            </mat-card>
          </div>

        </div>
      </div>
    </mat-card>
  </div>
</div>
```

Página Web - Processamento TS

```
import { DomSanitizer } from '@angular/platform-browser';
import { ngxLoadingAnimationTypes,NgxLoadingComponent } from 'ngx-loading';
import { HttpClient } from '@angular/common/http';
import { Observable, Observer } from 'rxjs';
import { ToastrService } from 'ngx-toastr';
import { DataECG } from '../../../shared/modelos/ECG';
import { CNNWebApiService } from '../../../shared/services/cnnapi/cnnapi';
import { Component, OnInit,ViewChild, TemplateRef } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from "@angular/forms";
import { DataCovid19 } from 'src/app/shared/modelos/DataCovid19';
import { map, flatMap } from 'rxjs/operators';
import { NgbCarousel } from '@ng-bootstrap/ng-bootstrap';

interface Diagnostico {
  tipo: string;
  shape: String;
}
```

```
const PrimaryWhite = '#ffffff';
const SecondaryGrey = '#ccc';
const PrimaryRed = '#dd0031';
const SecondaryBlue = '#006ddd';
@Component({
  selector: 'app-processamento',
  templateUrl: './processamento.component.html',
  styleUrls: ['./processamento.component.scss']
})
export class ProcessamentoComponent implements OnInit {
  @ViewChild('ngxLoading', { static: false })
  ngxLoadingComponent!: NgxLoadingComponent;

  @ViewChild('customLoadingTemplate', { static: false })
    customLoadingTemplate!: TemplateRef<any>;
  public quokkaAsyncData!: Observable<string>;
  public quokkaData!: string;
  public loadingTemplate!: TemplateRef<any>;
  public primaryColour = PrimaryWhite;
  public secondaryColour = SecondaryGrey;
  public ngxLoadingAnimationTypes = ngxLoadingAnimationTypes;
  public config = { animationType: ngxLoadingAnimationTypes.none,
    primaryColour: this.primaryColour, secondaryColour: this.
    secondaryColour, tertiaryColour: this.primaryColour,
    backdropBorderRadius: '3px' };
  public graph1 = {
    data: [
      { x: [], y: [], type: 'scatter', mode: 'lines+points', marker: {
        color: 'red' }, name:"Sem filtro" },
      { x: [], y: [], type: 'scatter', mode: 'lines+points', marker: {
        color: 'blue' }, name:"Com filtro" }
    ],
    layout: { width: 640, height: 550, title: 'Original - RR intervals (ms)
      ' }
  };
  public graph2 = {
    data: [
      { x: [], y: [], type: 'scatter', mode: 'lines+points', marker: {
        color: 'red' }, }
    ],
    layout: { width: 640, height: 550, title: 'Filtrado - RR intervals (ms)
      ' }
  };
  public graph3 = {
    data: [
      { z: [], type: 'surface', showscale: false }
    ],
```

```
56      layout: { width: 640, height: 550, title: 'Wavelet Surface RR' }
    };
58
    public graph_histograma_covid_antes = {
60
      data: [
62      {
            x:[],
64          y:[],
          type: 'bar',
66      }
      ],
68    layout: { width: 640, height: 550, title: 'Histograma Original' }
    };
70


72  diagnosticos: Diagnostico[] = [
      { tipo: 'VFC', shape: '.txt' }
74
    ];
76  demonstracao=false;
    file!: File
78  show_resultdo = false;
    loading_value = false;
80  show_vfc = false
    dataECG!: DataECG;
82
    paused = false;
84  mostrar_vfc=false;

86  uploadForm: FormGroup;
    dados_vfc!: String;
88
    @ViewChild('carousel', {static : true}) carousel!: NgbCarousel;
90
    constructor(private sanitizer: DomSanitizer,
92    // private toastr: ToastrService,
       public fb: FormBuilder, private cnnwebapiservice: CNNWebApiService) {
94    this.uploadForm = this.fb.group({
        'diagnostico': ['',
96        Validators.required
        ],
98      'dados_input': ['', [
          Validators.required
100     ]],
        'valido': [false, [
102       Validators.requiredTrue
```

```
      ]],
104   });

106 }
  arquivo_vfc_atual=''
108 arquivo_vfc_atual_index=0
  arquivos_vfc_original=[
110   ['https://raw.githubusercontent.com/rafaelgov95/repotest/master/VFC
        /0-1.txt','0-1.txt'],
      ['https://raw.githubusercontent.com/rafaelgov95/repotest/master/VFC
        /1-1.txt','1-1.txt'],
112   ['https://raw.githubusercontent.com/rafaelgov95/repotest/master/VFC
        /2-1.txt','2-1.txt']
  ]
114 arquivos_vfc:any= []
  ngOnInit() {
116   // this.loadingTemplate = this.customLoadingTemplate;

118  }

120 ngOnDestroy() {
    this
122 }


124
  checkFile() {
126   console.log("Check Filse")
    this.mostrar_vfc=false
128   if (this.uploadForm.value.diagnostico.tipo == "VFC"){
    for (let index = 0; index < this.arquivos_vfc_original.length; index++)
        {
130     const element = this.arquivos_vfc_original[index][0];
          this.downloadDataAsTxt(element,index)
132       console.log(index)
      }
134   }
    if (this.file) {
136    if ((this.uploadForm.value.diagnostico.tipo == "VFC") && (this.file.
        type == "text/plain")) {
        // this.FormshowSuccess()
138     this.uploadForm.controls['valido'].setValue(true)
      } else if (this.uploadForm.value.diagnostico.tipo == "VFC") {
140     // this.FormatError("VFC")
        this.uploadForm.controls['valido'].setValue(false)
142   }
    } else {
144     this.uploadForm.controls['valido'].setValue(false)
```

```
      this.uploadForm.controls['dados_input'].reset()
146   this.show_resultdo = false
      this.show_vfc = false
148  }
}
150

showPreview(event:any) {
152  const file_local = (event.target as HTMLInputElement).files![0]
  console.log(file_local)
154  if (file_local) {
    this.file = file_local
156    this.uploadForm.controls['dados_input'].reset()
    console.log("Change File")
158    this.show_resultdo = false
    this.show_vfc = false
160

    const reader = new FileReader();
162

    if (file_local.type == "text/plain") {
164      reader.readAsText(file_local)
      reader.onload = () => {
166        const valor: string = reader.result as string
        this.uploadForm.controls['dados_input'].setValue(valor)
168        // this.uploadForm.controls['dados_input'].setValue(JSON.
            stringify(valor))
      }
170  } else {
      // this.FormatFileInvalid(this.file.type)
172  }

174  }
  console.log(this.uploadForm.controls['dados_input'])
176

}
178 uploadFormPreview=true
submit() {
180

  if (this.uploadForm.valid) {
182    this.loading_value = true;
    let user = JSON.parse(localStorage.getItem("currentUser") || '{}')[0];
184   if (this.uploadForm.value.diagnostico.tipo == "VFC") {
      this.cnnwebapiservice.Diagnostico(this.uploadForm.controls['
          dados_input'].value, "VFC",user.nome).subscribe(
186       data => {
          this.loading_value = false;
188       this.dataECG = data
          this.graph1.data[0].x = this.dataECG.Plot_Original_x_axis
```

```
190             this.graph1.data[0].y = this.dataECG.Plot_Original_y_axis

192             this.graph1.data[1].x = this.dataECG.Plot_Filtrada_x_axis
                this.graph1.data[1].y = this.dataECG.Plot_Filtrada_y_axis
194
                this.graph2.data[0].x = this.dataECG.Plot_Filtrada_x_axis
196             this.graph2.data[0].y = this.dataECG.Plot_Filtrada_y_axis

198             this.graph3.data[0].z = this.dataECG.Plot_Wavelet

200 //              this.showSuccess()
                this.show_resultdo = true
202             this.show_vfc = true;
              },
204           error => {
//              this.FormatAPI();
206             this.loading_value = false;
                this.show_vfc = false;
208           });
              this.mostrar_vfc=!this.mostrar_vfc
210       }


212
      }
214
  }
216 // FormshowSuccess() {
  //   this.toastr.success('Carregados com Sucesso!', 'Dados de Entrada');
218 // }
  // showSuccess() {
220 //   this.toastr.success('Sucesso!', 'Request');
  // }
222 // FormatError(tipo) {
  //   this.toastr.error('Formato inv lido para o tipo ' + tipo, 'Dado de
      Entrada');
224 // }

226 // FormatAPI() {
  //   this.toastr.error('Erro na requisi  o ', 'API CNNWEB');
228 // }

230 // FormatFileInvalid(tipo) {
  //   this.toastr.error('Tipo ' + tipo + ' file inv lido ', 'Dado de
      Entrada');
232 // }

234 mostrar_teste(){
```

```
  if(this.uploadForm.controls['diagnostico'].value.tipo=='VFC'){
236
      this.arquivo_vfc_atual = this.arquivos_vfc[0]
238   this.mostrar_vfc =!this.mostrar_vfc

240   }

242 }


244


246 proximo_vfc(){
    console.log((this.arquivo_vfc_atual_index) )
248
    if((this.arquivo_vfc_atual_index+1) >= this.arquivos_vfc.length){
250   this.arquivo_vfc_atual = this.arquivos_vfc[0]
      this.arquivo_vfc_atual_index = 0
252
    }else{
254   this.arquivo_vfc_atual_index = this.arquivo_vfc_atual_index+1

256   this.arquivo_vfc_atual = this.arquivos_vfc[this.
          arquivo_vfc_atual_index]
    }
258
  }
260 anterior_vfc(){
    console.log(this.arquivos_vfc)
262 console.log((this.arquivo_vfc_atual_index) )
    if((this.arquivo_vfc_atual_index-1) >= 0){
264   this.arquivo_vfc_atual_index = this.arquivo_vfc_atual_index-1

266   this.arquivo_vfc_atual = this.arquivos_vfc[(this.
          arquivo_vfc_atual_index)]

268 }else{
      this.arquivo_vfc_atual_index = 0
270
      this.arquivo_vfc_atual = this.arquivos_vfc[0]
272 }
  }
274
  select_vfc(arquivo_vfc:any){
276 this.file = new File([''],this.arquivo_vfc_atual_index.toString())
    this.uploadForm.controls['valido'].setValue(true)
278 this.uploadForm.controls['dados_input'].setValue(this.arquivo_vfc_atual
        )
```

```
    this.mostrar_vfc=!this.mostrar_vfc

  }

   downloadDataAsTxt(url: string, index:number) {
    console.log(url)
    this.cnnwebapiservice.downloadDataAsTxt(url).subscribe(
      data => this.arquivos_vfc[index]=data,
      error => console.log(error)
    );

  }

}
```