

PROJETO DA DISCIPLINA IN1102

APRENDIZADO DE MÁQUINA (2024.1)

Bruno Ramos, Jorge Mariz, Rodrigo Araujo e Tertius Ferraz, UFPE
e-mail: bsr, jlvm, raa11, tsgf [@cin.ufpe.br]

Abstract—As redes neurais são modelos de aprendizado de máquina que se baseiam no teorema da aproximação universal e simulam processos fisiológicos do cérebro humano, sendo capazes de realizar tarefas de aprendizado supervisionado ou não supervisionado. A partir de uma arquitetura composta por camadas de neurônios interconectados com pesos e funções de ativação associados, devido à sua flexibilidade e capacidade de generalização de relações não lineares, este tornou-se um dos modelos de aprendizado mais populares. *Multi-Layer Perceptrons (MLPs)* são um tipo flexível de rede neural totalmente conectada que emprega uma estrutura *feedforward* com retropropagação. Uma alternativa promissora é a *Kolmogorov-Arnold Network (KAN)*, baseada no teorema de representação de Kolmogorov-Arnold. Embora ambas possuam arquitetura totalmente conectada, as *MLPs* têm funções de ativação fixas nos neurônios, enquanto as *KANs* têm funções de ativação que podem ser aprendidas nas arestas (pesos). Como resultado, as *KANs* não têm nenhuma matriz de peso linear, já que cada parâmetro de peso é substituído por uma função univariada parametrizada como *spline*, de modo que os nós das *KANs* apenas somam os sinais recebidos sem aplicar nenhuma não linearidade. Este estudo, portanto, apresenta a evolução das *KANs* e suas principais características, comparando sua capacidade de classificação em sete *datasets* com o desempenho de outros sete modelos: *k-Nearest Neighbors (k-NN)*, *Learning Vector Quantization (LVQ)*, *Support Vector Machine (SVM)*, *Decision Trees (DT)*, *Random Forest (RF)*, *MLP* e um classificador *stacking* destes. As *KANs* mostraram-se superiores aos demais modelos de classificação somente em um dos *dataset* testados, embora tenha apresentado desempenho competitivo na maioria dos *dataset*. Entretanto, seus hiperparâmetros não estão ainda suficientemente ajustados, um maior esforço neste âmbito deve ampliar seu potencial em estudos subsequentes.

Index Terms—Aprendizado de máquina, Redes neurais, Teorema de representação, *Kolmogorov-Arnold Network*.

I. INTRODUÇÃO

UMA rede neural artificial (*Artificial Neural Networks, ANN*) é um programa ou modelo de aprendizado de máquina que toma decisões a partir da metáfora do cérebro humano, usando processos que imitam a maneira como os neurônios biológicos trabalham juntos para identificar fenômenos, pesar opções e chegar a conclusões. Cada rede neural consiste em camadas de nós, ou neurônios artificiais, contendo uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Cada nó se conecta a outros e tem seu próprio peso e limite (*threshold*) associados. Se a saída de qualquer nó individual estiver acima do valor limite especificado, esse nó será ativado, enviando dados para a próxima camada da rede; caso contrário, nenhum dado será transmitido para a próxima camada da rede [?]. As redes

neurais dependem de dados de treinamento para aprender e melhorar sua precisão ao longo do tempo, mas uma vez ajustados, elas se tornam ferramentas poderosas que permitem classificar e agrupar dados em alta velocidade. As tarefas de reconhecimento de fala ou de imagem podem levar minutos contra as horas demandadas pela identificação manual por especialistas humanos, onde um dos exemplos mais conhecidos de rede neural é o algoritmo de busca do Google [?].

As redes neurais podem ser classificadas em diferentes tipos, sendo utilizadas para diferentes finalidades. O *perceptron* é a rede neural mais antiga, criada por Rosenblatt [?], em que a ideia de McCulloch e Pitt [?] de comparar neurônios com um *threshold* binário com lógica booleana foi levada adiante ao serem introduzidos pesos na equação. Ivakhnenko e Lapa [?] propuseram a primeira rede neural *feedforward* de aprendizagem profunda (*Deep Learning*), ainda sem usar gradiente descendente estocástico, mas Amari [?] empregou uma rede de aprendizagem profunda usando gradiente descendente estocástico para classificar classes de padrões não linearmente separáveis. *Feedforward Neural Networks* ou *Multi-Layer Perceptrons (MLPs)* [?] são redes neurais compostas por uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída, e embora sejam chamadas de *MLPs*, elas são compostas por neurônios sigmóides ao invés de *perceptrons*, já que a maioria dos problemas do mundo real não são lineares [?].

Embora não tenham sido os desenvolvedores da ideia da retropropagação (*backpropagation*), LeCun *et al.* [?] empregaram uma rede neural para reconhecer dígitos de códigos postais escritos à mão fornecidos pelo Serviço Postal dos EUA, ilustrando como o uso de restrições na retropropagação e sua integração na arquitetura de uma rede neural pode ser usado para treinar algoritmos. As redes neurais convolucionais (*Convolutional Neural Networks, CNNs*), por sua vez, são semelhantes às redes *feedforward*, mas geralmente são utilizadas para reconhecimento de imagens, padrões e/ou visão computacional, baseando-se em princípios da álgebra linear, particularmente multiplicação de matrizes, para identificar padrões dentro de uma imagem. Já as redes neurais recorrentes (*Recurrent Neural Networks, RNNs*) são identificadas por seus ciclos de *feedback*, sendo empregadas principalmente ao usar dados de séries temporais para fazer previsões sobre resultados futuros, como previsões do mercado de ações ou previsões de vendas [?].

Em linhas gerais, dentre as vantagens apresentadas pelas *ANNs*, podem-se citar sua capacidade de fazer previsões mesmo quando a informação é incompleta, sua tolerância

a falhas em células e sua capacidade de treinamento em paralelo. Por outro lado, é possível citar como limitações a necessidade de ajuste da arquitetura da rede, a dificuldade na interpretação dos resultados e a necessidade da inserção de variáveis numéricas para treinar a rede [?].

Uma alternativa às *MLPs*, que se baseiam no teorema da aproximação universal [?], são as *Kolmogorov-Arnold Networks (KANs)*, propostas recentemente por Liu *et al.* [?] a partir de uma inspiração no teorema de representação de Kolmogorov-Arnold [?], [?]. Embora ambas possuam arquitetura totalmente conectada, enquanto as *MLPs* têm funções de ativação fixas em nós (neurônios), as *KANs* têm funções de ativação que podem ser aprendidas nas arestas (pesos). Como resultado, as *KANs* não têm nenhuma matriz de peso linear, já que cada parâmetro de peso é substituído por uma função univariada parametrizada como *spline*, tal que os nós das *KANs* simplesmente somam os sinais recebidos sem aplicar nenhuma não linearidade. Devido a esse potencial inovador ainda pouco explorado pela comunidade científica, este estudo se propõe a aprofundar o entendimento sobre as *KANs* e experimentar seu desempenho em comparação com outros modelos de aprendizagem.

Este estudo, portanto, visa implementar uma *KAN* conforme a descrição de Liu *et al.* [?] e comparar seu desempenho com outros 7 classificadores, os modelos de aprendizado *k-Nearest Neighbors (k-NN)*, *Learning Vector Quantization (LVQ)*, *Support Vector Machine (SVM)*, *Decision Trees (DT)*, *Random Forest (RF)*, *MLP* e um classificador *stacking* composto a partir do desempenho destes seis classificadores individuais. Os desempenhos de todos estes classificadores serão validados em 7 *datasets*: os bancos de dados *Iris*, *Wine Quality*, *Adult*, *Brest Cancer Winsconsin* e *Heart Disease*, obtidos no *UCI Machine Learning Repository*, além dos *datasets Rain in Australia* e *Spotify Tracks*. Após análises dos classificadores e seus desempenhos frente aos *datasets*, foram propostas modificações nos parâmetros dos modelos a fim de otimizar o desempenho dos algoritmos, especialmente o *KAN*.

Uma vez que sejam verificados os desempenhos de cada modelo de classificação em cada *dataset*, serão empregados testes estatísticos não paramétricos para verificar quais modelos são superiores em cada circunstância. Conforme preconizado por Demsar [?], será empregado o teste de Friedman [?] e, uma vez que seja refutada a hipótese nula, será aplicado o teste *post-hoc* de Nemenyi [?], capaz de criar um ranking entre os modelos.

II. REVISÃO BIBLIOGRÁFICA

O algoritmo *k-NN* é um classificador de aprendizagem supervisionado não paramétrico, que usa proximidade para fazer classificações ou previsões sobre o agrupamento de uma instância individual. É um dos algoritmos de classificação e regressão mais populares e simples usados no aprendizado de máquina atualmente. Para problemas de classificação, um rótulo de classe é atribuído com base no voto majoritário, ou seja, é usado o rótulo que é representado com mais frequência em torno de uma determinada instância considerando a métrica de distância escolhida para avaliar este dado e seus vizinhos [?].

Quantizadores vetoriais de aprendizagem (*LVQ*) são modelos esparsos para aprendizagem de classificação baseada em protótipos, sendo um tipo de rede neural que se concentra na divisão do espaço dos dados em regiões correspondentes a diferentes classes, contando com um esquema de atração e repulsão para adaptação dos protótipos. O *LVQ* atribui pesos a cada um dos nós da rede que são ajustados durante a fase de treinamento para que o modelo possa classificar os dados de forma eficaz, sendo este ajuste baseado na medida de distância escolhida, assim como o *k-NN*. Assim, através da aprendizagem competitiva, o protótipo mais próximo da amostra de treinamento é aquele que será atualizado, aproximando-se ou afastando-se conforme favorece os resultados da classificação [?], [?].

Uma máquina de vetores de suporte (*SVM*) é um algoritmo de aprendizado supervisionado que classifica dados ao definir hiperplanos que maximizem as distâncias entre indivíduos nas fronteiras de cada classe em um espaço *n*-dimensional. O número de características nos dados de entrada determina se o hiperplano é uma linha em um espaço 2-D ou um plano em um espaço *n*-dimensional, e como vários hiperplanos podem ser encontrados para diferenciar classes, maximizar a margem entre os pontos permite que o algoritmo encontre o melhor limite de decisão entre as classes. As linhas adjacentes ao hiperplano ideal são conhecidas como vetores de suporte [?].

Uma árvore de decisão (*DT*) é um algoritmo de aprendizagem supervisionado não paramétrico com estrutura hierárquica em árvore, podendo ser empregado tanto em tarefas de classificação como de regressão. A estrutura das *DTs* consiste em um nó raiz, que não possui nenhuma ramificação de entrada, ramificações de saída do nó raiz, nós internos (nós de decisão) e nós folha (nós terminais), onde estes últimos representam os resultados possíveis dentro do conjunto de dados [?].

A floresta aleatória (*RF*) é um método de aprendizagem em conjunto (*ensemble*) que combina as decisões de múltiplas *DTs* para chegar a uma única decisão, sendo a classe selecionada pela maioria das árvores o *output* de problemas de classificação. Este algoritmo combina a ideia de *bagging* (ou agregação *bootstrap*) e seleção aleatória de características para construir uma coleção de árvores de decisão com variância controlada [?].

Conforme indicado anteriormente, um dos tipos mais populares de redes neurais é a *MLP*, sendo compostas por uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída, dotadas de neurônios *sigmóides* e empregando técnicas como *feedforward* e *backpropagation* [?]. Em linhas gerais, dentre as vantagens apresentadas pelas *MLPs* (ou mesmo de outras redes neurais), podem-se citar sua capacidade de fazer previsões mesmo quando a informação é incompleta, sua tolerância a falhas em células e sua capacidade de treinamento em paralelo. Por outro lado, é possível citar como limitações a necessidade de ajuste da arquitetura da rede, a dificuldade na interpretação dos resultados e a necessidade da inserção de variáveis numéricas para treinar a rede [?].

A abordagem por empilhamento (*stacking*) é um método de aprendizagem em conjunto (*ensemble*) que utiliza diferentes modelos de aprendizado de máquina, um após o outro, onde

se adicionam as previsões de cada modelo para criar um modelo generalizado final. A arquitetura desta abordagem envolve dois ou mais modelos básicos, geralmente chamados de modelos de nível 0, e um metamodelo que combina as previsões dos modelos básicos, chamado de modelo de nível 1. O empilhamento de modelos deve sempre ser empregado com validação cruzada para reduzir o sobreajuste dos modelos aos dados de treinamento, sendo empregado também um conjunto de treinamento e um de validação (*holdout*) em cada nível. Ao contrário do *bagging*, os modelos empregados no *stacking* normalmente são diferentes e se ajustam ao mesmo conjunto de dados ao invés de amostras do conjunto de dados de treinamento. Ao contrário do *boosting*, em vez de uma sequência de modelos que corrigem as previsões dos modelos anteriores, um único modelo é usado no *stacking* para aprender como combinar melhor as previsões dos modelos contribuintes. Em essência, espera-se que as previsões de cada modelo empregado no *stacking* se tornem uma parcela de um novo modelo preditivo, de modo que cada um destes consiga prever uma fatia dos dados de treinamento para o modelo generalizado final [?], [?].

Servindo como fundamentação às KANs [?], Vladimir Arnold [?] e Andrey Kolmogorov [?] estabeleceram que se f é uma função contínua multivariada em um domínio limitado, então f pode ser escrita como uma composição finita de funções contínuas 1D e a operação binária de adição. Mais especificamente, para um $f \in [0, 1]^n \rightarrow R$, tem-se a Equação ??:

$$f(x) = f(x_1, \dots, x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^n \Psi_{q,p}(x_p) \right) \quad (1)$$

Liu *et al.* [?] propuseram as KANs, generalizando a rede para larguras e profundidades arbitrárias. A despeito da interpretação matemática, as KANs são basicamente combinações de *splines* e *MLPs*, aproveitando os pontos fortes de ambas e evitando seus respectivos pontos fracos. *Splines* são precisas para funções de baixa dimensão, fáceis de ajustar localmente e capazes de alternar entre diferentes resoluções, mas apresentam sérios problemas quanto à maldição de dimensionalidade devido à sua incapacidade de explorar estruturas composicionais. Por outro lado, as *MLPs*, sofrem menos com a maldição de dimensionalidade graças ao seu aprendizado de recursos, mas são menos precisas que as *splines* em dimensões baixas, devido à sua incapacidade de otimizar funções univariadas [?].

Para aprender uma função com precisão, um modelo não deve apenas aprender a estrutura composicional (graus de liberdade externos, ou Φ_q), mas também aproximar bem as funções univariadas (graus de liberdade internos, ou $\Psi_{q,p}$). KANs são modelos que abordam ambos, pois possuem *MLPs* na parte externa e *splines* na parte interna. Como resultado, as KANs podem não apenas aprender recursos (graças à sua semelhança externa com *MLPs*), mas também otimizar esses recursos aprendidos com grande precisão (graças à sua semelhança interna com as *splines*). Portanto, Liu *et al.* [?] demonstraram que através de mudanças simples na arquitetura,

as KANs superaram as *MLPs* em termos de precisão e interpretabilidade. A Figura ?? apresenta uma comparação entre as arquiteturas de uma *MLP* e uma *KAN*.

Como todas as funções a serem aprendidas são funções univariadas, pode-se parametrizar cada função 1D como uma curva *B-spline*, com coeficientes que podem ser aprendidos de funções de base *B-spline* locais (Figura 2b). Desta maneira, tem-se um protótipo de *KAN* cuja arquitetura é especificada exatamente pela Equação ?? e ilustrada na Figura 1b (com dimensão de entrada $n = 2$), compondo uma rede neural de duas camadas, cuja intermediária possui largura $2n + 1$, e cujas funções de ativação são inseridas nas arestas ao invés de nós, onde a soma simples é realizada nos nós. A Figura ?? apresenta um esquema de como as funções de ativação fluem pela rede e como são parametrizadas como *B-splines*.

Em termos de precisão, KANs muito menores podem alcançar precisão comparável ou melhor do que *MLPs* muito maiores no ajuste de dados e na resolução de Equações Diferenciais Parciais (*Partial Differential Equations, PDEs*), reduzindo o número parâmetros e possibilitando melhor generalização e interpretabilidade. Entretanto, geralmente o usuário vai desconhecer a arquitetura ótima de uma *KAN* antes de resolver o problema. Desta maneira, Liu *et al.* [?] propuseram uma abordagem que inicia pela proposição de uma *KAN* grande, seguida de seu treino por meio de uma regularização esparsa e de um procedimento de poda, de modo que KANs podadas são mais interpretáveis do que as não podadas. Esta abordagem é subdividida em quatro etapas: esparsificação, visualização, poda e simbolificação.

III. METODOLOGIA

A. DESCRIÇÃO DOS DATASETS E PRÉ-PROCESSAMENTO DOS DADOS

O banco de dados *Iris* contém um conjunto de 150 amostras com informações sobre quatro atributos, sendo *sepal length*, *sepal width*, *petal length*, *petal width* e a variável de saída, a coluna *species*, contendo os tipos de pétalas *Setosa*, *Versicolor* e *Virginica*. As variáveis de entrada são numéricas, o banco de dados já é balanceado e não há dados faltantes, de modo que não é necessário empregar nenhum tratamento adicional aos dados.

O banco de dados *Wine Quality* contém um conjunto de 4.898 amostras com informações sobre onze atributos, sendo *fixed acidity*, *volatile acidity*, *citric acid*, *residual sugar*, *chlorides*, *free sulfur dioxide*, *total sulfur dioxide*, *density*, *pH*, *sulphates*, *color* e *alcohol* as variáveis de entrada e *quality* a variável de saída, contendo valores entre 0 e 10. Todas as variáveis são numéricas, a variável alvo é desbalanceada e não há dados faltantes, de modo que foi necessário realizar técnicas de oversampling.

O banco de dados *Adult* contém um conjunto de 48.842 amostras com informações sobre quatorze atributos, sendo *age*, *workclass*, *fnlwgt*, *education*, *education-num*, *marital-status*, *occupation*, *relationship*, *race*, *sex*, *capital-gain*, *capital-loss*, *hours-per-week* e *native-country* as variáveis de entrada e *income* a variável de saída, cujo objetivo é prever se a renda de um indivíduo excede US\$ 50 mil/ano. Este *dataset*

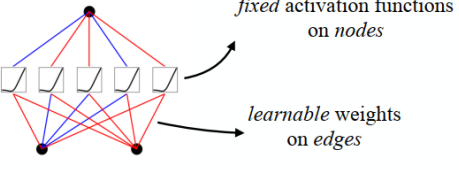
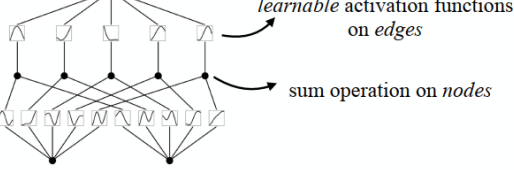
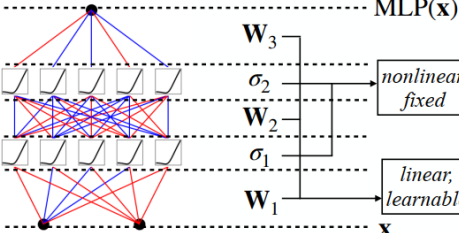
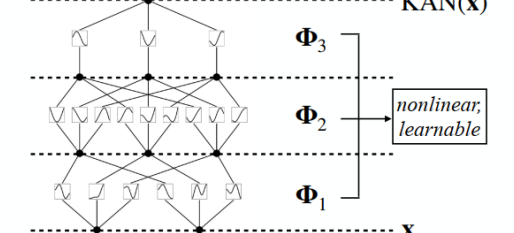
Model	Multi-Layer Perceptron (MLP)	Kolmogorov-Arnold Network (KAN)
Theorem	Universal Approximation Theorem	Kolmogorov-Arnold Representation Theorem
Formula (Shallow)	$f(\mathbf{x}) \approx \sum_{i=1}^{N(\epsilon)} a_i \sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i)$	$f(\mathbf{x}) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^n \phi_{q,p}(x_p) \right)$
Model (Shallow)	(a) 	(b) 
Formula (Deep)	$\text{MLP}(\mathbf{x}) = (\mathbf{W}_3 \circ \sigma_2 \circ \mathbf{W}_2 \circ \sigma_1 \circ \mathbf{W}_1)(\mathbf{x})$	$\text{KAN}(\mathbf{x}) = (\Phi_3 \circ \Phi_2 \circ \Phi_1)(\mathbf{x})$
Model (Deep)	(c) 	(d) 

Fig. 1: Comparação entre as arquiteturas de uma *Multi-Layer Perceptron* (MLP) e uma *Kolmogorov-Arnold Network* (KAN) (LIU *et al.* [?])

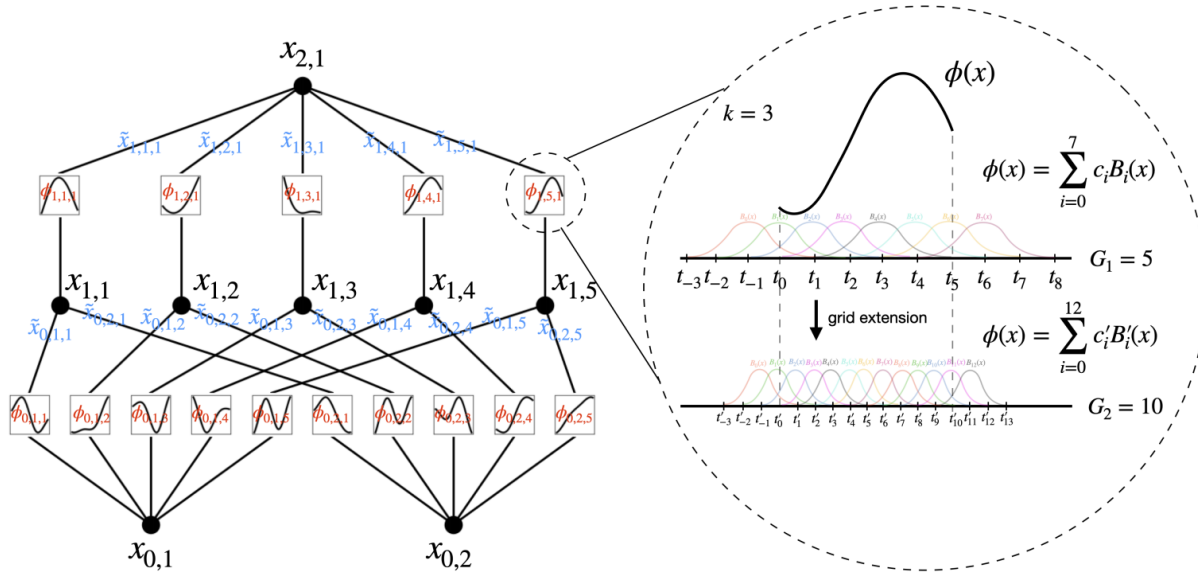


Fig. 2: Funções de ativação que fluem pela rede (a) e uma função de ativação parametrizada como *B-spline*, que permite alternar entre *grids* maiores e menores (b) (LIU *et al.* [?]).

contém variáveis numéricas e categóricas, não apresenta dados faltantes, de modo que foi aplicado *one hot encoder* nas variáveis categóricas, possibilitando tratar todo o conjunto de dados como numérico.

O banco de dados *Brest Cancer Winsconsin* contém um conjunto de 569 amostras com informações sobre dez atributos, sendo *radius*, *texture*, *perimeter*, *area*, *smoothness*, *compact-*

ness, *concavity*, *concave points*, *symmetry* e *fractal dimension* esses atributos. Como foram computadas em cada imagem a média, o erro padrão e o pior/menor (média dos três maiores valores) para cada variável, este *dataset* na verdade apresenta trinta variáveis de entrada, sendo *ID number* e *diagnosis* as demais variáveis dos dados, podendo esta última representar um câncer benigno ou maligno. As variáveis de entrada são

numéricas, o banco de dados já é balanceado e não há dados faltantes, de modo que não é necessário empregar nenhum tratamento adicional aos dados.

O banco de dados *Heart Disease*, por sua vez, contém um conjunto de 303 amostras com informações sobre treze atributos, sendo *age*, *sex*, *cp*, *trestbps*, *chol*, *fbs*, *restecg*, *thalach*, *exang*, *oldpeak*, *slope*, *ca* e *thal* as variáveis de entrada, enquanto *num* é a variável de saída, que indica o diagnóstico de doença cardíaca. Este *dataset* contém variáveis numéricas e categóricas, além de haver dados faltantes, de modo foi necessário realizar um *simpleimputer*. A variável alvo também está desbalanceada.

O banco de dados *Rain in Australia* possui 145.000 amostras e informações associadas a 22 colunas de entrada. A variável de resposta seria *RainTomorrow*, indicando se haveria ou não chuva no dia seguinte. Este *dataset* contém variáveis numéricas e categóricas, além de haver dados faltantes, para contornar esses problemas foi utilizado o *IterativeImputer* do *sklearn*, numa tentativa de lidar com os dados faltantes levando em consideração o grande grau de correlação entre as variáveis. Para lidar com as variáveis categóricas, os valores faltantes foram preenchidos randomicamente de acordo com a distribuição de probabilidade.

O banco de dados *Spotify Tracks* contém 114.000 amostras e informações associadas a 19 variáveis de entrada. A variável resposta seria a coluna *track_genre*, composta por 115 categorias. Dentre as 19 colunas de entrada, todas aquelas que possuíam variáveis categóricas foram removidas (i.e.: *track_id*, *artists*, *album_name*, *track_name*, *key*, *liveness*, *mode*, *explicit*, *time_signature*). Destarte, as colunas restantes para possibilitar o treinamento dos modelos foram *popularity*, *duration_ms*, *danceability*, *energy*, *loudness*, *speechiness*, *acousticness*, *instrumentalness*, *valence* e tempo. Não foi necessário eliminar nenhuma das 114.000 amostras, pois não haviam dados faltantes no conjunto de dados.

B. TREINAMENTO E VALIDAÇÃO DOS MODELOS

Considerando um determinado valor de *seed*, para assegurar que os modelos de aprendizado fossem executados nas mesmas partições, os dados foram divididos em conjuntos de treinamento (80%) e teste (20%). Cada um destes conjuntos é composto por suas respectivas variáveis de entrada (X) e saída (y), sendo empregados em seguida no *pipeline* de treinamento de cada modelo de aprendizado. Para minimizar o efeito da diferença de unidades entre as colunas, os dados do conjunto X de treinamento foram padronizados (*scaling*) e transformados, transformação também aplicada nos conjuntos X de validação e teste, evitando assim um possível sobreajustamento.

Foi efetuada uma comparação quanto ao desempenho dos algoritmos *k-NN*, *LVQ*, *SVM*, *DT*, *RF*, *MLP* e uma versão *stacking* destes, além da avaliação do desempenho de uma rede neural *KAN*. Para possibilitar uma otimização dos hiperparâmetros destes modelos de aprendizado, foi empregada uma estratégia *grid search* com validação cruzada em 10 *folds*, considerando a recuperação das métricas *accuracy*, *precision*, *recall*, *F1* e *Auc-ROC*. Desta forma, seria possível identificar para cada algoritmo qual configuração de hiperparâmetros

resultaria em um menor erro. Neste âmbito, a Tabela ?? apresenta os conjuntos de hiperparâmetros testados para cada modelo de classificação.

TABLE I: Conjuntos de hiperparâmetros testados para cada modelo de classificação

<i>k-NN</i>	n neighbors = range(1, 50); metric = Euclidean, Manhattan;
<i>LVQ</i> :	prototype n per class = range(1, 4); distance type = Euclidean, Manhattan e Chebyshev; solver params = (max runs = 5; step size = 0.1, 0.5 e 1.0);
<i>SVM</i> :	C = 0.1, 1; gamma = 0.1, 1; kernel = RBF, Sigmoid;
<i>DT</i> :	criterion = Gini e entropy; max depth = range(1, 10); min samples split = range(2, 10); min samples leaf = range(1, 10);
<i>RF</i> :	n estimators = 10, 20, 30 e 50; criterion = Gini e entropy; max depth = range(1, 6); min samples split = range(2, 6); min samples leaf = range(1, 6);
<i>MLP</i> :	hidden layer sizes = (50,), (100,), (50, 50), (100, 100); activation = Tanh e ReLU; solver = SGD e Adam; alpha = 0.0001 e 0.05; learning rate = constant e adaptive;
<i>Stacking</i> :	Nível 1: estimators = <i>k-NN</i> , <i>LVQ</i> , <i>DT</i> , <i>ANN</i> e <i>RF</i> com conjuntos de hiperparâmetros otimizados por meio do <i>grid search</i> e da validação cruzada; Nível 2: final estimator = LogisticRegression (max iter = 2000), cv = 10;
<i>KAN</i> :	Input dimension = Associado ao dataset; Layer = 1; Neurons = 20; Grid = 10; k = 4

O número máximo de iterações de cada modelo de aprendizado foi determinado com 1.500 iterações, com exceção do nível 2 do modelo *stacking*, onde foram consideradas 2.000 iterações em um modelo de regressão logística. Uma vez que as configurações ideais resultaram em um melhor modelo para o conjunto de treinamento, seria verificado ainda o desempenho deste modelo no conjunto de validação. Já em relação ao ajuste dos hiperparâmetros do modelo *KAN*, não foi possível realizar experimentos suficientes para otimizá-los. Ao passo que o *Input_dimension* e o *Output_dimension* estão associados aos *datasets* sendo experimentados, inserir um número de neurônios maior que 20 inviabilizava a execução do algoritmo. Não foi testado também o efeito que um maior número de camadas exerceria na capacidade do algoritmo. O parâmetro *k*, por fim, é a ordem polinomial do polinômio que se está tentando aproximar. De antemão, antes mesmo de apresentar os resultados, já sugerimos que é necessário que mais testes sejam efetuados para que seja possível explorar o potencial do modelo.

C. FLUXOGRAMA DE EXECUÇÃO

A Figura ?? apresenta o fluxograma executado no treinamento e teste de cada modelo de aprendizagem aplicado a cada *dataset*.

Algorithm 1: Procedimento de Treinamento, Teste e Comparação de Modelos de Aprendizagem

Aplicação: Aprendizagem Supervisionada;
Input: $Algorithm = [k-NN, LVQ, DT, RF, MLP, Stacking, KAN]$;
 $Datasets = [Iris, Wine\ quality, Adult, Breast\ cancer\ Wisconsin, Heart\ disease, Rain\ in\ Australia, Spotify\ tracks]$;
 $Scores = [Accuracy, Precision, Recall, F1, Roc-AUC]$;
 Sejam $i \in I$ os diferentes $Datasets$ e $j \in J$ os diferentes $Algorithms$;
for $i \in I$ **do**
 Pré-processamento dos dados:
 Tratamento individualizado para corrigir desbalanceamento, dados faltantes e variáveis categóricas em cada $dataset_i$ (quando necessário), seguido da Subdivisão das colunas em variáveis de entrada (X_i) e saída (y_i);
 Treinamento e Teste dos Modelos:
 Separação dos dados em conjuntos de treino (80%) e teste (20%);
 Padronização (*scaling*) dos valores de $X_{itreino}$ (*fit*), seguido da transformação dos valores de X_{iteste} e X_{iteste} ;
 for $j \in J$ **do**
 $Params_j$ = Dicionário contendo listas de valores a serem combinados para cada hiperparâmetro de j ;
 $GridSearch_{ij} = GridSearch(j, Params_j, cv = 10, Scores)$;
 $BestModel_{ij} = GridSearch_{ij}(X_{itreino}, y_{itreino})$;
 $BestModelTest_{ij} = BestModel_{ij}(X_{iteste}, y_{iteste})$;
 end
 Comparação entre Modelos:
 $Friedman_i(GridSearch_{ij} \forall j \in J)$;
 $Nimenyi_i(GridSearch_{ij} \forall j \in J)$;
 $BestSolution_i = \underset{Nimenyi_i Rank}{Argmax} (GridSearch_{ij} \forall j \in J)$;
end
Output: $BestSolution_i \forall i \in I$

Fig. 3: Fluxograma da execução de treinamento e teste

D. FERRAMENTAS COMPUTACIONAIS

Os experimentos foram conduzidos em *Python*, sendo empregadas as bibliotecas *Numpy*, *Pandas* e *Scikit-Learning* para efetuar o tratamento dos dados (*train test split*, *GridSearchCV*, etc). Quanto à implementação dos modelos de aprendizado, foram usadas também bibliotecas do *Scikit-Learning* (*KNeighborsClassifier*, *GLVQ*, *DecisionTreeClassifier*, *MLPClassifier*, *RandomForestClassifier* e *StackingClassifier*). As métricas empregadas para avaliar os modelos também foram implementadas a partir de bibliotecas do *Scikit-Learning* (*make_scorer*, *recall_score*, *f1_score*, *accuracy_score*, *precision_score* e *roc_auc_score*).

Foram empregados 3 computadores: o primeiro com processador Intel i5 de 7ª geração, 12Gb de memória RAM e sistema operacional Windows (64bits); o segundo com processador AMD Ryzen 5 3600 6-Core, 16Gb da memória RAM e sistema operacional Windows (64bits); enquanto o terceiro contém processador i7 de 11 geração com 16 gb de memória ram e sistema operacional Windows 11 (64bits). Esta divisão se fez necessária, uma vez que não haveria tempo hábil para executar todos os experimentos em uma única máquina.

IV. RESULTADOS E DISCUSSÃO

Nesta seção são apresentados os resultados dos experimentos efetuados em cada *dataset*, considerando o desempenho

dos oito classificadores nos conjuntos de validação cruzada e teste em cada *dataset*.

A. IRIS DATASET

Após a aplicação do *grid search* com validação cruzada em 10 *folds* no conjunto de treinamento do *Iris Dataset*, a Tabela ?? apresenta os melhores hiperparâmetros verificados para cada modelo de classificação, ao passo que a Tabela ?? apresenta as métricas obtidas nestes dados. A Tabela ??, por sua vez, apresenta as métricas obtidas no conjunto de testes do *Iris Dataset*.

TABLE II: Melhores hiperparâmetros nos conjuntos de validação cruzada do Iris Dataset

<i>k-NN</i> :	$n_neighbors = 10$; $metric = Euclidean$;
<i>LVQ</i> :	$prototype_n_per_class = 3$; $distance_type = Euclidean$; $solver_params = (max_runs = 5; step_size = 1.0)$;
<i>SVM</i> :	$C = 1$; $gamma = 0.1$; $kernel = Sigmoid$;
<i>DT</i> :	$criterion = Gini$; $max_depth = 3$; $min_samples_split = 2$; $min_samples_leaf = 3$;
<i>RF</i> :	$n_estimators = 10$; $criterion = Gini$; $max_depth = 2$; $min_samples_split = 2$; $min_samples_leaf = 3$;
<i>MLP</i> :	$hidden_layer_sizes = (50, 50)$; $activation = Tanh$; $solver = Adam$; $alpha = 0.0001$; $learning_rate = constant$;

Tanto nos conjuntos de treinamento quanto de teste, todos os classificadores apresentaram resultados satisfatórios. Em relação à validação cruzada, os piores resultados foram obtidos

TABLE III: Desempenho na Validação cruzada - Iris Dataset

<i>Iris Dataset - Validação Cruzada</i>					
<i>Classificador</i>	<i>Accuracy</i>	<i>Recall</i>	<i>F1</i>	<i>Precision</i>	<i>Auc_ROC</i>
<i>k-NN</i>	0.97	0.97	0.96	0.97	0.98
<i>LVQ</i>	0.93	0.93	0.92	0.94	0.99
<i>SVM</i>	0.97	0.96	0.97	0.97	1.00
<i>DT</i>	0.95	0.95	0.95	0.93	0.97
<i>RF</i>	0.93	0.93	0.92	0.95	0.99
<i>MLP</i>	0.96	0.96	0.96	0.97	1.00
<i>Stacking</i>	0.96	0.96	0.96	0.97	1.00
<i>KAN</i>	0.94	0.95	0.94	0.94	

* As células hachuradas contém os melhores desempenhos.

TABLE IV: Desempenho no Teste - Iris Dataset

<i>Iris Dataset - Teste</i>					
<i>Classificador</i>	<i>Accuracy</i>	<i>Recall</i>	<i>F1</i>	<i>Precision</i>	<i>Auc_ROC</i>
<i>k-NN</i>	0.97	0.97	0.97	0.97	0.99
<i>LVQ</i>	1.00	1.00	1.00	1.00	1.00
<i>SVM</i>	1.00	1.00	1.00	1.00	1.00
<i>DT</i>	0.93	0.95	0.93	0.93	1.00
<i>RF</i>	0.97	0.97	0.97	0.96	1.00
<i>MLP</i>	1.00	1.00	1.00	1.00	1.00
<i>Stacking</i>	0.97	0.97	0.97	0.96	1.00
<i>KAN</i>	0.97	0.97	0.97	0.96	

* As células hachuradas contém os melhores desempenhos.

pelos *LVQ* e *RF*, ao passo que os melhores resultados foram obtidos pelos *k-NN* e *SVM*. Já no conjunto de teste, o pior desempenho foi o do classificador *DT*, ao passo que os classificadores *LVQ*, *SVM* e *MLP* atingiram os valores máximos para todas as métricas averiguadas. O classificador *KAN* apresentou um desempenho mediano no *Iris dataset*.

Em relação ao teste de Friedman, a única métrica em que foi possível rejeitar a hipótese nula foi a *Auc_ROC*, indicando que havia uma diferença estatística entre os classificadores durante a validação cruzada. Não foi possível calcular esta métrica para o *KAN*, de modo que o teste de Nemenyi foi aplicado nos demais resultados. Visto que muitos classificadores apresentaram desempenhos semelhantes, onde muitos *folds* atingem a nota máxima, a implementação usada do *Auc_ROC* não foi capaz de criar um ranking adequado. Ela atribui melhor posição ao classificador verificado primeiro, mesmo que este tenha uma nota semelhante aos demais, fato verificado em diferentes testes, de modo que o teste não foi conclusivo para o *Iris dataset*.

B. WINE QUALITY DATASET

Após a aplicação do *grid search* com validação cruzada em 10 *folds* no conjunto de treinamento do *Wine Quality Dataset* exibe-se na Tabela ?? os melhores hiperparâmetros verificados para cada modelo de classificação, ao passo que a Tabela ?? apresenta as métricas obtidas nestes dados. A Tabela ??, por sua vez, apresenta as métricas obtidas no conjunto de testes do *Wine Quality Dataset*.

De toda a base de dados somente a coluna *color* foi excluída da análise num primeiro momento, ao passo que as outras 11 colunas são valores reais. Na etapa de treino os classificadores que apresentaram melhor comportamento foram *k-NN*, *MLP* e *stacking*. O fato dos dados estarem próximos uns dos outros e a maneira como os algoritmos executam as aproximações

TABLE V: Melhores hiperparâmetros nos conjuntos de validação cruzada do Wine Quality Dataset

<i>k-NN</i> :	n_neighbors = 1; metric = Manhattan;
<i>LVQ</i> :	prototype_n_per_class = 3; distance_type = Euclidean; solver_params = (max_runs = 5; step_size = 0.5);
<i>SVM</i> :	C = 1; gamma = 1; kernel = rbf;
<i>DT</i> :	criterion = entropy; max_depth = 9; min_samples_split = 3; min_samples_leaf = 1;
<i>RF</i> :	n_estimators = 50; criterion = entropy; max_depth = 5; min_samples_split = 2; min_samples_leaf = 4;
<i>MLP</i> :	hidden_layer_sizes = (100, 100); activation = tanh; solver = adam; alpha = 0.0001; learning_rate = adaptive;

TABLE VI: Desempenho na Validação cruzada - Wine Quality Dataset

<i>Wine Quality Dataset - Validação cruzada</i>					
<i>Classificador</i>	<i>Accuracy</i>	<i>Recall</i>	<i>F1</i>	<i>Precision</i>	<i>Auc_ROC</i>
<i>k-NN</i>	0.906	0.906	0.903	0.902	0.945
<i>LVQ</i>	0.454	0.454	0.425	0.432	0.795
<i>SVM</i>	0.893	0.894	0.892	0.891	0.987
<i>DT</i>	0.715	0.716	0.712	0.712	0.929
<i>RF</i>	0.636	0.636	0.616	0.618	0.901
<i>MLP</i>	0.904	0.904	0.900	0.900	0.976
<i>Stacking</i>	0.922	0.923	0.922	0.922	0.990
<i>KAN</i>	0.143	0.141	0.035	0.02	-

* As células hachuradas contém os melhores desempenhos.

TABLE VII: Desempenho no Teste - Wine Quality Dataset

<i>Wine Quality Dataset - Teste</i>					
<i>Classificador</i>	<i>Accuracy</i>	<i>Recall</i>	<i>F1</i>	<i>Precision</i>	<i>Auc_ROC</i>
<i>k-NN</i>	0.916	0.915	0.912	0.913	0.951
<i>LVQ</i>	0.468	0.468	0.439	0.437	0.795
<i>SVM</i>	0.908	0.907	0.905	0.904	0.989
<i>DT</i>	0.714	0.711	0.710	0.711	0.931
<i>RF</i>	0.633	0.631	0.612	0.616	0.901
<i>MLP</i>	0.914	0.913	0.910	0.910	0.979
<i>Stacking</i>	0.936	0.935	0.935	0.934	0.992
<i>KAN</i>	0.14	0.02	0.14	0.035	-

* As células hachuradas contém os melhores desempenhos.

beneficiaram *k-NN* e *MLP*. Em contrapartida, no treinamento os piores desempenhos foram do *KAN* e *LVQ*.

Na etapa de teste, conforme visto na tabela ??, o melhor classificador foi *Stacking*. De modo geral, o desempenho no teste foi bom apenas para alguns algoritmos, como *k-NN*, *SVM*, *MLP* e *Stacking*. Já para *LVQ*, *DT*, *RF* e *KAN* o desempenho não foi bom. Um fator que pode ter influenciado na queda de performance, é o desbalanceamento do banco de dados, que apesar de ter sido contornado com *RandomOverSampler* não foi favorecido pelos algoritmos.

Para o balanceamento não foi viável técnicas de *under sampling*, pois algumas classificações de qualidade tinham poucos dados, especificamente na casa de unidades, enquanto outras estavam na casa de milhar (i.e.: a nota 9 aparece em 5 ocorrências, enquanto a nota 6 aparece 2836 vezes). Deste modo, técnicas de *over sampling* foram as alternativas possíveis. Entre *Synthetic Minority Over-sampling Technique (SMOTE)* e *RandomOverSampler*, esta apresentou melhores resultados e comportamento no balanceamento.

O classificador *KAN* passou por mais testes em que foram alterados o número de neurônios na camada oculta, de 10 passou para 19 neurônios. Contudo, obteve-se os mesmos resultados da tabela ??.

Do teste de Friedman foi verificados que os valores de p para *Accuracy*, *Recall*, *F1 Score*, *Precision* e *Roc-Auc* foram menores que 0,05 e proximos de 0, logo há evidências estatísticas de que existe ao menos uma diferença significativa entre os grupos testados. Por fim, foi aplicado o teste de *Nemenyi* em que os menores rankings médios, portanto, melhor avaliados foram: *Stacking*, *SVM* e *MLP*.

C. ADULT DATASET

Após a aplicação do *grid search* com validação cruzada em 10 *folds* no conjunto de treinamento do *Adult Dataset*, a Tabela ?? apresenta os melhores hiperparâmetros verificados para cada modelo de classificação, ao passo que a Tabela ?? apresenta as métricas obtidas nestes dados. A Tabela ??, por sua vez, apresenta as métricas obtidas no conjunto de testes do *Adult Dataset*.

TABLE VIII: Melhores hiperparâmetros nos conjuntos de validação cruzada do Adult Dataset

<i>k-NN</i> :	n_neighbors = 17; metric = Euclidean;
<i>LVQ</i> :	prototype_n_per_class = 3; distance_type = Euclidean; solver_params = (max_runs = 5; step_size = 0.1);
<i>SVM</i> :	C = 1; gamma = 0.1; kernel = RBF;
<i>DT</i> :	criterion = Gini; max_depth = 9; min_samples_split = 8; min_samples_leaf = 3;
<i>RF</i> :	n_estimators = 30; criterion = entropy; max_depth = 5; min_samples_split = 2; min_samples_leaf = 4;
<i>MLP</i> :	hidden_layer_sizes = (50,); activation = Tanh; solver = Adam; alpha = 0.05; learning_rate = constant;

TABLE IX: Desempenho na Validação cruzada - Adult Dataset

Adult Dataset - Validação Cruzada					
Classificador	Accuracy	Recall	F1	Precision	Auc_ROC
<i>k-NN</i>	0.84	0.76	0.77	0.79	0.89
<i>LVQ</i>	0.76	0.51	0.45	0.88	0.85
<i>SVM</i>	0.86	0.77	0.79	0.82	0.90
<i>DT</i>	0.86	0.78	0.79	0.81	0.90
<i>RF</i>	0.83	0.68	0.71	0.82	0.90
<i>MLP</i>	0.86	0.78	0.79	0.81	0.91
<i>Stacking</i>	0.86	0.78	0.80	0.82	0.92
<i>KAN</i>	0.84	0.74	0.76	0.79	

* As células hachuradas contém os melhores desempenhos.

TABLE X: Desempenho no Teste - Adult Dataset

Adult Dataset - Teste					
Classificador	Accuracy	Recall	F1	Precision	Auc_ROC
<i>k-NN</i>	0.84	0.77	0.78	0.79	0.89
<i>LVQ</i>	0.76	0.51	0.45	0.88	0.85
<i>SVM</i>	0.86	0.77	0.80	0.82	0.90
<i>DT</i>	0.86	0.78	0.80	0.81	0.91
<i>RF</i>	0.84	0.71	0.74	0.81	0.90
<i>MLP</i>	0.87	0.79	0.81	0.83	0.92
<i>Stacking</i>	0.87	0.79	0.81	0.83	0.92
<i>KAN</i>	0.86	0.78	0.80	0.82	

* As células hachuradas contém os melhores desempenhos.

Em relação à validação cruzada, os piores resultados foram obtidos pelo *LVQ*, ao passo que os melhores resultados foram obtidos pelo classificador *Stacking*. Já no conjunto de teste, o pior desempenho apresentado foi novamente o do *LVQ*, ao passo que os classificadores *MLP* e *Stacking* apresentaram as melhores métricas. O classificador *KAN* apresentou um

desempenho próximo aos dos melhores classificadores no *Adult dataset*.

Em relação ao teste de Friedman, em todas as métricas foi possível rejeitar a hipótese nula, indicando que havia uma diferença estatística entre os classificadores durante a validação cruzada. Portanto, o teste de Nemenyi foi aplicado a esses conjuntos de dados. O classificador *MLP* esteve entre os dois melhores em todos os testes, ao passo que o *KNN* esteve quatro vezes nas melhores colocações, seguido pelo *LVQ*, que foi o primeiro colocado na métrica *Auc_ROC*. Em geral, o desempenho do *KAN* foi mediano a baixo nestes testes, não sendo possível calcular seu desempenho para a métrica *Auc_ROC*.

D. BREAST CANCER WINSCONSIN DATASET

Após a aplicação do *grid search* com validação cruzada em 10 *folds* no conjunto de treinamento do *Breast Cancer Winsconsin Dataset*, a Tabela ?? apresenta os melhores hiperparâmetros verificados para cada modelo de classificação, ao passo que a Tabela ?? apresenta as métricas obtidas nestes dados. A Tabela ??, por sua vez, apresenta as métricas obtidas no conjunto de testes do *Breast Cancer Winsconsin Dataset*.

TABLE XI: Melhores hiperparâmetros nos conjuntos de validação cruzada do Breast Cancer Winsconsin Dataset

<i>k-NN</i> :	n_neighbors = 3; metric = Manhattan;
<i>LVQ</i> :	prototype_n_per_class = 3; distance_type = Euclidean; solver_params = (max_runs = 5; step_size = 0.5);
<i>SVM</i> :	C = 1; gamma = 0.1; kernel = RBF;
<i>DT</i> :	criterion = entropy; max_depth = 6; min_samples_split = 2; min_samples_leaf = 1;
<i>RF</i> :	n_estimators = 10; criterion = Gini; max_depth = 5; min_samples_split = 3; min_samples_leaf = 1;
<i>MLP</i> :	hidden_layer_sizes = (50, 50); activation = ReLu; solver = Adam; alpha = 0.0001; learning_rate = adaptive;

TABLE XII: Desempenho na Validação cruzada - Breast Cancer Winsconsin Dataset

Breast Cancer Winsconsin Dataset - Validação cruzada					
Classificador	Accuracy	Recall	F1	Precision	Auc_ROC
<i>k-NN</i>	0.97	0.97	0.97	0.97	0.98
<i>LVQ</i>	0.94	0.92	0.93	0.95	0.99
<i>SVM</i>	0.96	0.96	0.95	0.96	0.99
<i>DT</i>	0.93	0.93	0.93	0.93	0.93
<i>RF</i>	0.95	0.95	0.95	0.95	0.98
<i>MLP</i>	0.98	0.98	0.98	0.98	0.99
<i>Stacking</i>	0.98	0.98	0.98	0.99	0.99
<i>KAN</i>	0.96	0.93	0.93	0.94	

* As células hachuradas contém os melhores desempenhos.

Tanto nos conjuntos de treinamento quanto de teste, todos os classificadores apresentaram resultados satisfatórios. Em relação à validação cruzada, os piores resultados foram obtidos pelos classificadores *LVQ*, *DT* e *KAN*, ao passo que os melhores resultados foram obtidos pelos classificadores *Stacking* e *MLP*, respectivamente. Já no conjunto de teste, os piores desempenhos apresentados foram os do *LVQ*, *DT* e *SVM* ao passo que os classificadores *Stacking* e *MLP* novamente apresentaram as melhores métricas, respectivamente. O classificador *KAN* apresentou um desempenho próximo aos

TABLE XIII: Desempenho no Teste - Breast Cancer Winsconsin Dataset

<i>Breast Cancer Winsconsin Dataset - Teste</i>					
<i>Classificador</i>	<i>Accuracy</i>	<i>Recall</i>	<i>F1</i>	<i>Precision</i>	<i>Auc_ROC</i>
<i>k-NN</i>	0.97	0.95	0.96	0.97	0.97
<i>LVQ</i>	0.92	0.89	0.91	0.94	0.99
<i>SVM</i>	0.93	0.93	0.93	0.92	0.99
<i>DT</i>	0.94	0.92	0.93	0.96	0.95
<i>RF</i>	0.96	0.96	0.95	0.96	0.98
<i>MLP</i>	0.98	0.98	0.97	0.98	0.99
<i>Stacking</i>	0.98	0.98	0.98	0.99	0.99
<i>KAN</i>	0.97	0.96	0.97	0.98	

* As células hachuradas contém os melhores desempenhos.

dos melhores classificadores no conjunto de testes do *Breast Cancer Winsconsin dataset*.

Em relação ao teste de Friedman, em todas as métricas foi possível rejeitar a hipótese nula, indicando que havia uma diferença estatística entre os classificadores durante a validação cruzada. Portanto, o teste de Nemenyi foi aplicado a esses conjuntos de dados. Com exceção da métrica *precision*, os classificadores *Stacking* e *MLP* foram o primeiro e o segundo colocados em todas as métricas, respectivamente. Em relação à métrica *precision*, os melhores resultados foram obtidos pelo *LVQ*, *RF* e *Stacking*, respectivamente. Em geral, o desempenho do *KAN* foi baixo nestes testes, não sendo possível calcular seu desempenho para a métrica *Auc_ROC*.

E. HEART DISEASE DATASET

Após a aplicação do *grid search* com validação cruzada em 10 *folds* no conjunto de treinamento do *Heart Disease Dataset*, a Tabela ?? apresenta os melhores hiperparâmetros verificados para cada modelo de classificação, ao passo que a Tabela ?? apresenta as métricas obtidas nestes dados. A Tabela ??, por sua vez, apresenta as métricas obtidas no conjunto de testes do *Heart Disease Dataset*.

TABLE XIV: Melhores hiperparâmetros nos conjuntos de validação cruzada do Heart Disease Dataset

<i>k-NN</i> :	n_neighbors = 4; metric = Euclidean;
<i>LVQ</i> :	prototype_n_per_class = 3; distance_type = Euclidean; solver_params = (max_runs = 5; step_size = 0.1);
<i>SVM</i> :	C = 1; gamma = 1; kernel = Sigmoid;
<i>DT</i> :	criterion = entropy; max_depth = 8; min_samples_split = 2; min_samples_leaf = 2;
<i>RF</i> :	n_estimators = 10; criterion = entropy; max_depth = 5; min_samples_split = 5; min_samples_leaf = 1;
<i>MLP</i> :	hidden_layer_sizes = (100, 100); activation = ReLu; solver = SGD; alpha = 0.05; learning_rate = constant;

Em relação à validação cruzada, os melhores resultados foram obtidos pelos classificadores *k-NN*, tendo o *KAN* vencido os demais em relação à métrica *Recall*. Já no conjunto de teste, os piores desempenhos apresentados foram os do *k-NN*, sugerindo que o modelo teve *overfitting* nos dados de treinamento, ao passo que os classificadores *Stacking* e *KAN* apresentaram as melhores métricas.

Em relação ao teste de Friedman, somente em relação à métrica *Recall* foi possível rejeitar a hipótese nula, indicando que havia uma diferença estatística entre os classificadores durante a validação cruzada. Portanto, o teste de Nemenyi foi

TABLE XV: Desempenho na Validação cruzada - Heart Disease Dataset

<i>Heart Disease Dataset - Validação cruzada</i>					
<i>Classificador</i>	<i>Accuracy</i>	<i>Recall</i>	<i>F1</i>	<i>Precision</i>	<i>Auc_ROC</i>
<i>k-NN</i>	0.62	0.37	0.36	0.36	
<i>LVQ</i>	0.58	0.33	0.29	0.29	
<i>SVM</i>	0.53	0.31	0.30	0.31	
<i>DT</i>	0.55	0.33	0.31	0.31	
<i>RF</i>	0.58	0.31	0.28	0.28	
<i>MLP</i>	0.56	0.34	0.30	0.27	
<i>Stacking</i>	0.60	0.31	0.29	0.31	
<i>KAN</i>	0.60	0.39	0.35	0.35	

* As células hachuradas contém os melhores desempenhos.

TABLE XVI: Desempenho no Teste - Heart Disease Dataset

<i>Heart Disease Dataset - Teste</i>					
<i>Classificador</i>	<i>Accuracy</i>	<i>Recall</i>	<i>F1</i>	<i>Precision</i>	<i>Auc_ROC</i>
<i>k-NN</i>	0.54	0.27	0.23	0.21	0.64
<i>LVQ</i>	0.53	0.29	0.29	0.30	0.76
<i>SVM</i>	0.51	0.32	0.33	0.42	0.65
<i>DT</i>	0.43	0.28	0.27	0.29	0.62
<i>RF</i>	0.56	0.29	0.27	0.30	0.78
<i>MLP</i>	0.54	0.34	0.32	0.32	0.77
<i>Stacking</i>	0.57	0.33	0.31	0.29	0.79
<i>KAN</i>	0.54	0.37	0.36	0.36	

* As células hachuradas contém os melhores desempenhos.

aplicado, indicando que os melhores classificadores aplicados ao *Heart Disease dataset* eram o *KAN* e o *MLP*, respectivamente. Novamente, não foi possível calcular o desempenho do *KAN* para a métrica *Auc_ROC*.

TABLE XVII: Desempenho na Validação cruzada - Heart Disease Dataset

<i>Heart Disease Dataset - Validação cruzada</i>					
<i>Classificador</i>	<i>Accuracy</i>	<i>Recall</i>	<i>F1</i>	<i>Precision</i>	<i>Auc_ROC</i>
<i>k-NN</i>	0.62	0.37	0.36	0.36	
<i>LVQ</i>	0.58	0.33	0.29	0.29	
<i>SVM</i>	0.53	0.31	0.29	0.31	
<i>DT</i>	0.55	0.33	0.31	0.31	
<i>RF</i>	0.58	0.27	0.25	0.24	
<i>MLP</i>	0.56	0.34	0.30	0.27	
<i>Stacking</i>	0.62	0.38	0.36	0.38	
<i>KAN</i>	0.60	0.39	0.35	0.35	

* As células hachuradas contém os melhores desempenhos.

TABLE XVIII: Desempenho no Teste - Heart Disease Dataset

<i>Heart Disease Dataset - Teste</i>					
<i>Classificador</i>	<i>Accuracy</i>	<i>Recall</i>	<i>F1</i>	<i>Precision</i>	<i>Auc_ROC</i>
<i>k-NN</i>	0.54	0.27	0.23	0.21	0.64
<i>LVQ</i>	0.53	0.29	0.29	0.30	0.76
<i>SVM</i>	0.49	0.30	0.29	0.29	0.79
<i>DT</i>	0.43	0.28	0.27	0.29	0.62
<i>RF</i>	0.53	0.35	0.35	0.39	0.76
<i>MLP</i>	0.54	0.34	0.32	0.32	0.77
<i>Stacking</i>	0.53	0.30	0.29	0.28	0.78
<i>KAN</i>	0.56	0.39	0.38	0.38	

* As células hachuradas contém os melhores desempenhos.

Na tabela ?? este dataset não obteve bons resultados nas métricas, em parte isto se deve a pouca quantidade de dados, particularmente a classe 4 por exemplo há apenas 8 pontos de dados. Foi tentado realização da técnica *SMOTE* e *RandomOverSampler*, os resultados na validação cruzada e em treino até melhoraram, passando dos 0.9 para todas as métricas, mas

as métricas no conjunto de teste tiveram desempenho ainda pior. Não era possível realizar um *UnderSampler*, pois como a classe minoritária tem poucos dados, faltaria informação para os modelos. Também ao realizar os testes estatísticos, não houve nenhuma métrica em que o p valor fosse abaixo de 0,05, logo não havia evidências suficiente para rejeitar a hipótese nula.

Apesar de não ter um bom resultado na validação cruzada, Na tabela ??, o KAN apresenta as melhores métricas para o conjunto de teste, tendo sido utilizado 2 camadas com 10 neurônios cada, grade = 3 e k = 2. Outra tentativa de tentar melhorar as métricas foi a atribuição de balanceamento de pesos para as classes nos algoritmos *RF*, *SVM* e *Stacking*, os quais apresentaram uma leve melhora em algumas métricas e piora em outras, mas nada que destoasse muito dos resultados anteriores.

F. RAIN IN AUSTRALIA DATASET

Após a aplicação do *grid search* com validação cruzada em 10 *folds* no conjunto de treinamento do *Rain in Australia Dataset*, a Tabela ?? apresenta os melhores hiperparâmetros verificados para cada modelo de classificação, ao passo que a Tabela ?? apresenta as métricas obtidas nestes dados. A Tabela ??, por sua vez, apresenta as métricas obtidas no conjunto de testes do *Rain in Australia Dataset*.

TABLE XIX: Melhores hiperparâmetros nos conjuntos de validação cruzada do Rain in Australia Quality Dataset

<i>k-NN</i> :	n_neighbors = 10; metric = euclidean;
<i>LVQ</i> :	prototype_n_per_class = 1; distance_type = euclidean; solver_params = (max_runs = 5; step_size = 0.1);
<i>SVM</i> :	C = 0.1; gamma = 0.1; kernel = sigmoid;
<i>DT</i> :	criterion = gini; max_depth = 9; min_samples_split = 4; min_samples_leaf = 8;
<i>RF</i> :	n_estimators = 50; criterion = gini; max_depth = 5; min_samples_split = 3; min_samples_leaf = 2;
<i>MLP</i> :	hidden_layer_sizes = (20,); activation = relu; solver = adam; alpha = 0.0001; learning_rate = adaptive;

TABLE XX: Desempenho na Validação cruzada - Rain in Australia Dataset

Rain in Australia Dataset - Validação cruzada					
Classificador	Accuracy	Recall	F1	Precision	Auc_ROC
<i>k-NN</i>	0,86	0,86	0,86	0,86	0,93
<i>LVQ</i>	0,75	0,75	0,75	0,75	0,83
<i>SVM</i>	0,5	0,5	0,33	0,25	0,5
<i>DT</i>	0,84	0,84	0,84	0,84	0,91
<i>RF</i>	0,81	0,81	0,81	0,81	0,9
<i>MLP</i>	0,88	0,88	0,87	0,91	0,9
<i>Stacking</i>	0,9	0,9	0,89	0,91	0,97
<i>KAN</i>	0,89	0,85	0,85	0,88	

* As células hachuradas contém os melhores desempenhos.

Em relação à validação cruzada, os melhores resultados foram obtidos pelo *Stacking*, enquanto o pior resultado foi o apresentado pelo *SVM*. Já no conjunto de teste, o melhor desempenho foram dos classificadores *DT* e *KAN*, tendo o *SVM* mais uma vez apresentado o pior desempenho.

Com relação aos testes estatísticos, em todas as métricas o teste de Friedman deu p < 0,05 descartando completamente a hipótese nula. Após aplicar o teste de Nemenyi foi observado

TABLE XXI: Desempenho no Teste - Rain in Australia Dataset

Rain in Australia Dataset - Teste					
Classificador	Accuracy	Recall	F1	Precision	Auc_ROC
<i>k-NN</i>	0,78	0,76	0,72	0,71	0,76
<i>LVQ</i>	0,75	0,75	0,75	0,75	0,75
<i>SVM</i>	0,22	0,5	0,18	0,11	0,5
<i>DT</i>	0,87	0,87	0,87	0,87	0,87
<i>RF</i>	0,8	0,78	0,74	0,72	0,78
<i>MLP</i>	0,85	0,71	0,74	0,82	0,71
<i>Stacking</i>	0,84	0,78	0,78	0,77	0,78
<i>KAN</i>	0,89	0,85	0,85	0,88	

* As células hachuradas contém os melhores desempenhos.

que o *SVM* juntamente com o *KNN* ocuparam a ultima posição em todas as métricas, para a *accuracy*, *recall*, *Auc_ROC* e *f1*, o modelo *Stacking* liderou os ranks, demonstrando robusteza em tal modelo, para a precisão o líder foi o *KAN*. Os modelos que mais se saíram bem com esse dataset, foram o *LVQ*, *KAN* e o *RF*.

Devido a complexidade dos dados e o seu volume, o custo computacional acabou sendo um empecilho considerável, levando em conta a demora de treinamento do *SVM* e do *MLP* bem como a validação cruzada de ambos, impedindo assim qualquer experimento para melhorar tais resultados.

G. SPOTIFY TRACKS DATASET

Após a aplicação do *grid search* com validação cruzada em 10 *folds* no conjunto de treinamento do *Spotify Tracks Dataset*, a Tabela ?? apresenta os melhores hiperparâmetros verificados para cada modelo de classificação, ao passo que a Tabela ?? apresenta as métricas obtidas nestes dados. A Tabela ??, por sua vez, apresenta as métricas obtidas no conjunto de testes do *Spotify Tracks Dataset*.

TABLE XXII: Melhores hiperparâmetros nos conjuntos de validação cruzada do Spotify Tracks Dataset

<i>k-NN</i> :	n_neighbors = 20; metric = Manhattan;
<i>LVQ</i> :	prototype_n_per_class = 2; distance_type = Euclidean; solver_params = (max_runs = 5; step_size = 0.1);
<i>SVM</i> :	C = 1; gamma = 0.1; kernel = RBF;
<i>DT</i> :	criterion = entropy; max_depth = 9; min_samples_split = 9; min_samples_leaf = 6;
<i>RF</i> :	n_estimators = 50; criterion = entropy; max_depth = 5; min_samples_split = 2; min_samples_leaf = 5;
<i>MLP</i> :	hidden_layer_sizes = (50, 50); activation = Tanh; solver = Adam; alpha = 0.05; learning_rate = constant;

TABLE XXIII: Desempenho na Validação cruzada - Spotify Tracks Dataset

Spotify Tracks Dataset - Validação cruzada					
Classificador	Accuracy	Recall	F1	Precision	Auc_ROC
<i>k-NN</i>	0,21	0,21	0,19	0,20	0,79
<i>LVQ</i>	0,17	0,17	0,15	0,14	0,84
<i>SVM</i>	0,22	0,22	0,19	0,21	0,09
<i>DT</i>	0,20	0,20	0,18	0,19	0,80
<i>RF</i>	0,19	0,19	0,13	0,15	0,89
<i>MLP</i>	0,26	0,26	0,25	0,25	0,92
<i>Stacking</i>	0,28	0,28	0,26	0,26	0,92
<i>KAN</i>	0,01	0,01	0,01	0,01	

* As células hachuradas contém os melhores desempenhos.

TABLE XXIV: Desempenho no Teste - Spotify Tracks Dataset

Classificador	Spotify Tracks Dataset - Teste				
	Accuracy	Recall	F1	Precision	Auc_ROC
k-NN	0,20	0,20	0,19	0,20	0,80
LVQ	0,16	0,16	0,14	0,13	0,84
SVM	0,22	0,22	0,15	0,21	0,90
DT	0,19	0,19	0,18	0,19	0,81
RF	0,19	0,19	0,13	0,14	0,89
MLP	0,27	0,27	0,25	0,26	0,92
Stacking	0,28	0,28	0,26	0,26	0,93
KAN	0,01	0,01	0,01	0,01	

* As células hachuradas contém os melhores desempenhos.

Conforme verificado ao longo do semestre, o *Spotify Dataset* mostra-se novamente um banco de dados desafiador, onde nenhum classificador atinge acurácias acima de 30%. Tanto em relação à validação cruzada como aos dados de testes, os melhores resultados foram obtidos pelos classificadores *MLP* e *Stacking*, respectivamente. O classificador *KAN* apresentou um desempenho muito baixo, claramente os hiperparâmetros empregados não foram capazes de contribuir com o aprendizado do modelo.

Em relação ao teste de Friedman, para todas as métricas foi possível rejeitar a hipótese nula, indicando que havia uma diferença estatística entre os classificadores durante a validação cruzada. Portanto, o teste de Nemenyi foi aplicado, e em todas as métricas os melhores resultados de classificação dos dados do *Spotify Tracks dataset* foram obtidos pelo *Stacking* e pelo *MLP*, respectivamente. Novamente, não foi possível calcular o desempenho do *KAN* para a métrica *Auc_ROC*.

Na tentativa de melhoria das métricas desse *dataset*, foram avaliadas diversas combinações de *features*, como também diversas combinações de hiper-parâmetros. Como as classes já estão balanceadas, chegamos a conclusão que a qualidade dos dados não é tão boa. o *KAN* tem desempenho insignificativo, não tendo sido possível testar camadas de neurônios muito grandes, pois acusava falta de memória no computador.

V. CONCLUSÃO

Este estudo teve como objetivo implementar e avaliar o desempenhos de *Kolmogorov-Arnold Networks (KANs)* quando aplicadas a tarefas de classificação em sete *datasets* diferentes, subdivididos em dados de treino e teste. Conforme indicado na metodologia, foram testados ainda outros sete classificadores nestes *datasets*, seguido de testes estatísticos não paramétricos para comparar e ranquear seus desempenhos.

Nos *datasets Iris* e *Adult*, as *KANs* apresentaram desempenhos medianos a baixos quando comparado com os demais modelos de classificação. Nos *datasets e Breast Cancer Winsconsin*, as *KANs* apresentaram desempenhos medianos a altos em relação aos demais modelos. Já no *dataset Heart Disease*, a *KAN* apresentou o melhor desempenho dentre todos os candidatos. Não foi possível apresentar o desempenho das *KANs* para o *Rain in Australia dataset*. Por último, nos *datasets Wine Quality* e *Spotify Tracks* (neste último todos os classificadores encontraram dificuldades), a *KAN* empregada foi incapaz de prever os resultados, sugerindo que seus hiperparâmetros devem ser melhor ajustados para que seu desempenhos seja, no mínimo, aceitável.

TABLE XXV: *p-value* das métricas para cada *dataset*.

<i>p-value</i>	Accuracy	Recall	F1 Score	Precision	Roc-Auc*
<i>Iris</i>	0,065	0,065	0,065	0,065	0,0068
<i>Wine Quality</i>	2,62e-12	2,62e-12	2,24e-12	2,62e-12	4,5e-11
<i>Adult</i>	2,95e-11	5,83e-11	1,44e-11	1,32e-8	4,86e-10
<i>Breast Cancer</i>	3,61e-5	7,59e-5	5,21e-5	6,82e-5	1,24e-6
<i>Heart Disease</i>	0,004	0,64	0,64	0,73	-
<i>Rainfall in Australia</i>	9,6e-5	9,6e-5	9,6e-5	2,4e-6	1,77e-7
<i>Spotify Tracks</i>	2,33e-12	2,97e-12	1,95e-12	5,51e-12	4,5e-11

* Testes não realizados no *KAN*.

Nos testes de Friedman, poucas foram as hipóteses aceitas, como visto na tabela ?? células em cinza. Em que na maioria das ocorrências a hipótese nula foi rejeitada, evidenciando que estatisticamente há diferença significativa entre as amostras. Além disso, na mesma tabela, os resultados de *Auc-Roc* não foram avaliados para o *KAN*, devido a dificuldades técnicas de sua implementação.

Infelizmente, os experimentos esbarraram na limitação computacional, pois a implementação da *KANs* conforme Liu *et al.* [?] preconizaram não permitiu empregar mais do que 20 neurônios na única camada, mesmo em *datasets* medianos. Consequentemente, não foi ainda possível efetuar um ajuste adequado em seus hiperparâmetros. O classificador *KAN*, não correspondeu às expectativas de desempenho ao lidar com *datasets* grandes. A documentação da ferramenta peca ao não demonstrar como adaptar, lidar ou configurar para usos específicos.

REFERENCES

- [1] IBM. (2024a) What is a neural network? Acesso em: 26/04/2024. [Online]. Available: <https://www.ibm.com/topics/neural-networks>
- [2] F. ROSENBLATT, "The perceptron: a probabilistic model for information storage and organization in the brain," *American Psychological Association*, vol. 65, no. 6, pp. 386–408, 1958. [Online]. Available: <http://dx.doi.org/10.1037/h0042519>
- [3] W. S. MCCULLOCH and W. PITTS, "A logical calculus of the ideas immanent in nervous activity," *Springer Science and Business Media LLC*, vol. 5, no. 4, pp. 115–133, 1943. [Online]. Available: <http://dx.doi.org/10.1007/bf02478259>
- [4] A. G. IVAKHNENKO and V. G. LAPA, "Cybernetics and forecasting techniques," *New York: American Elsevier Pub. Co.*, 1967.
- [5] S. AMARI, "A theory of adaptive pattern classifiers," *IEEE Transactions On Electronic Computers*, vol. 16, no. 3, pp. 299–307, 1967. [Online]. Available: <http://dx.doi.org/10.1109/pgec.1967.264666>
- [6] D. E. RUMELHART, G. E. HINTON, and R. J. WILLIAMS, "Learning internal representations by error propagation," *Readings In Cognitive Science. Elsevier*, vol. 65, no. 6, pp. 399–421, 1988. [Online]. Available: <http://dx.doi.org/10.1016/b978-1-4832-1446-7.50035-2>
- [7] Y. LECUN, B. BOSER, J. S. DENKER, D. HENDERSON, R. E. HOWARD, W. HUBBARD, and L. D. JACKEL, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–531, 1989. [Online]. Available: <http://dx.doi.org/10.1162/neco.1989.1.4.541>
- [8] D. A. KHALILOV, N. A. K. JUMABOYEVA, and T. M. K. KURBONOVA, "Advantages and applications of neural networks," *Academic Research in Educational Sciences*, vol. 2, no. 2, pp. 1153–1159, 2001.
- [9] K. HORNIK, M. STINCHCOMBE, and H. WHITE, "Multilayer feedforward networks are universal approximators," *Elsevier BV*, vol. 2, no. 5, pp. 359–366, 1989. [Online]. Available: [http://dx.doi.org/10.1016/0893-6080\(89\)90020-8](http://dx.doi.org/10.1016/0893-6080(89)90020-8)

- [10] Z. LIU, Y. WANG, S. VAIDYA, F. RUEHLE, J. HALVERSON, M.SOLJACIĆ, T.Y.HOU, and M. TEGMARK, “Kan: Kolmogorov-arnold networks,” *ArXiv*, 2024. [Online]. Available: <https://doi.org/10.48550/arXiv.2404.19756>
- [11] A. N. KOLMOGOROV, “On the representation of continuous functions of several variables as superpositions of continuous functions of one variable and addition,” 1956, acesso em: 17/05/2024. [Online]. Available: <https://cs.uwaterloo.ca/~y328yu/classics/Kolmogorov57.pdf>
- [12] V. ARNOLD, “On the representation of functions of several variables as a superposition of functions of a smaller number of variables,” *Springer Berlin Heidelberg*, pp. 25–46, 2009. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-01742-1_5
- [13] J. DEMSAR, “Statistical comparisons of classifiers over multiple data sets,” *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.
- [14] M. FRIEDMAN, “A comparison of alternative tests of significance for the problem of m rankings,” *The Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86–92, 1940. [Online]. Available: <http://dx.doi.org/10.1214/aoms/1177731944>
- [15] P. B. NEMENYI, “Distribution-free multiple comparisons,” Ph.D. dissertation, Princeton University, 1963.
- [16] IBM. (2024b) What is the k-nearest neighbors (knn) algorithm? Acesso em: 26/04/2024. [Online]. Available: <https://www.ibm.com/topics/knn#:~:text=Evelyn%20Fix%20and%20Joseph%20Hodges,%E2%80%9C9CNearest%20Neighbor%20Pattern%20Classification.%E2%80%9D>
- [17] W. K. WONG, X. H. ZENG, and K. F. AU, “Selecting the location of apparel manufacturing plants using neural networks,” *Optimizing Decision Making in the Apparel Supply Chain Using Artificial Intelligence (AI)*, pp. 41–54, 2013. [Online]. Available: <http://dx.doi.org/10.1533/9780857097842.41>
- [18] A. MEYER-BAESE and V. SCHMID, “Foundations of neural networks,” *Pattern Recognition and Signal Analysis in Medical Imaging*, pp. 197–243, 2014. [Online]. Available: <http://dx.doi.org/10.1016/b978-0-12-409545-8.00007-8>
- [19] IBM. (2023) What are support vector machines (svms)? Acesso em: 26/04/2024. [Online]. Available: <https://www.ibm.com/topics/support-vector-machine>
- [20] —. (2024c) What is a decision tree? Acesso em: 15/04/2024. [Online]. Available: <https://www.ibm.com/topics/decision-trees>
- [21] —. (2024d) What is a random forest? Acesso em: 29/04/2024. [Online]. Available: <https://www.ibm.com/topics/random-forest>
- [22] —. (2020) Stack machine learning models: Get better results. Acesso em: 29/04/2024. [Online]. Available: <https://developer.ibm.com/articles/stack-machine-learning-models-get-better-results/>
- [23] J. BROWNLEE. (2021) Stacking ensemble machine learning with python. Acesso em: 26/04/2024. [Online]. Available: <https://machinelearningmastery.com/stacking-ensemble-machine-learning-with-python/>