Algoritmos de Aprendizado de Máquina para Detecção de Parkinson

Salmaze P. H.[1], **Joioso A. V. B.**[1], **Gagini P. A. M.**[1]

[1]Instituto De Ciências Matemáticas e de Computação – Universidade de São Paulo (USP)
São Carlos – SP – Brazil

Resumo. *Este trabalho investigou a aplicação de diferentes algoritmos de aprendizado de máquina para diagnóstico de Parkinson através de análise de imagens de espirais e senoides desenhadas à mão. Foram comparados métodos clássicos (Random Forest, SVM e XGBoost), em configurações de classificação binária e multiclasse, com e sem redução de dimensionalidade via PCA. Esses métodos foram avaliados em relação ao modelo de aprendizado profundo YOLOv11n-cls, com o objetivo de verificar se o maior custo computacional dos modelos mais complexos é compensado por ganhos significativos de desempenho.*

1. Motivação

O Mal de Parkinson é uma das doenças mais cruéis que assola nossa sociedade atualmente, causando graves impactos na qualidade de vida dos portadores que, em estágios mais avançados, dependem dos cuidados de outros. Apesar do alto número de pesquisas na área, não existe uma cura definida ainda; os tratamentos utilizados hodiernamente se baseiam em retardar o avanço da doença e, assim, manter a qualidade de vida do paciente pelo maior tempo possível. Nesse contexto, o diagnóstico precoce da doença se torna uma grande vantagem no tratamento.

A detecção humana de Parkinson precoce é extremamente difícil; para auxiliar em tal diagnóstico, diferentes técnicas foram utilizadas e testadas no decorrer dos anos. Técnicas com especial relevância são as baseadas em algoritmos de Aprendizado de Máquina (AM), pela capacidade dos modelos de treinar em dados existentes e anotados e detectar padrões e nuanças jamais perceptíveis por humanos. Uma grande variedade de algoritmos e modelos podem ser empregados em tal tarefa, indo desde modelos capazes de rodar em um simples celular até modelos que precisam de centenas de GPUs. Neste trabalho, buscamos comparar diferentes abordagens, algoritmos clássicos e de menor custo computacional baseados em análise de dados tabulares a um modelo mais robusto, com custo consideravelmente maior e baseado em dados visuais, imagens, com o objetivo de medir as diferenças de desempenho e analisar se o elevado custo computacional se justifica para essa aplicação.

2. Introdução

A Inteligência Artificial (IA) tem revolucionado o diagnóstico médico, especialmente no contexto de detecção precoce de Parkinson. Algoritmos de AM demonstram capacidade excepcional de identificar padrões sutis em dados médicos que frequentemente escapam à percepção humana, oferecendo precisão diagnóstica superior aos métodos convencionais. Enquanto o diagnóstico tradicional baseado em avaliação clínica apresenta taxa de erro de

aproximadamente 25% e depende da manifestação de sintomas motores — quando mais da metade dos neurônios dopaminérgicos já foi comprometida —, sistemas baseados em IA podem alcançar acurácia de até 96% e detectar sinais da doença até 7 anos antes dos primeiros sintomas se manifestarem.

O campo oferece um amplo espectro de soluções algorítmicas, desde modelos clássicos computacionalmente eficientes, como Random Forest, SVM e XGBoost, adequados desde para dispositivos móveis, até arquiteturas de aprendizado profundo mais robustas como redes neurais convolucionais, que oferecem capacidade superior de processamento de dados complexos. Apesar dos avanços promissores, existe uma lacuna significativa na literatura quanto à análise comparativa custo-benefício entre diferentes abordagens algorítmicas, onde modelos de aprendizado profundo demonstram desempenho superior mas apresentam custo computacional elevado. Este trabalho busca preencher essa lacuna através da comparação sistemática entre algoritmos clássicos de baixo custo computacional e modelos de aprendizado profundo mais robustos, especificamente na análise de espirais e ondas desenhadas à mão para diagnóstico de Parkinson, avaliando se o investimento computacional adicional se justifica pelo ganho real na performance diagnóstica.

3. Fundamentação Teórica

Nesta seção abordaremos os fundamentos teóricos de cada um dos algoritmos de AM utilizados no trabalho, apresentando seu funcionamento, principais vantagens e desvantagens.

3.1. Algoritmos Clássicos

3.1.1. Random Forest

O algoritmo Random Forest surgiu como uma maneira de evoluir as já estabelecidas árvores de decisão combinando múltiplas dessas árvores para gerar resultados mais precisos e robustos. A primeira etapa do algoritmo é a seleção de amostras, onde uma técnica chamada *bootstrap aggregating* (*bagging*) é usada para extrair diversas amostras aleatórias do conjunto original de dados com reposição. Em seguida, cada uma dessas amostras é utilizada para criar uma árvore de decisão diferente. Por fim, para cada árvore, em cada nó, o algoritmo seleciona aleatoriamente um subconjunto de *features* para determinar a melhor divisão. Isso introduz diversidade entre as árvores e reduz a correlação entre elas. Com o treinamento finalizado, para a tarefa de classificação, cada árvore contribui com uma predição, e então a classe final é determinada pela maioria das árvores.

Entre os principais pontos fortes do Random Forest temos: menor risco de *overfitting*, por combinar múltiplas árvores de decisão, já que é um problema comum em árvores de decisão individuais; facilidade de uso, por possuir poucos parâmetros para ajuste; e a possibilidade de análise da importância de cada *feature*, facilitando a interpretação e seleção de variáveis. Quanto aos pontos negativos, temos: custo computacional relativamente elevado pela necessidade de construir múltiplas árvores e menor interpretabilidade do modelo final em comparação com árvores de decisão isoladas.

3.1.2. Support Vector Machine

O algoritmo Support Vector Machine (SVM) surgiu com o objetivo de encontrar um hiperplano ótimo que melhor separe os dados em diferentes classes. A primeira etapa do algoritmo é a identificação do hiperplano ideal, onde o SVM busca maximizar a margem entre as classes, definida como a distância entre o hiperplano e os pontos de dados mais próximos de cada classe. Em seguida, o algoritmo identifica os "vetores de suporte", que são os pontos de dados mais próximos do hiperplano e críticos para definir sua posição e orientação. Por fim, para dados que não são linearmente separáveis, o SVM utiliza o "truque do kernel"(*kernel trick*), que transforma o espaço de entrada em um espaço de dimensão mais alta, onde os dados se tornam linearmente separáveis. Com o treinamento finalizado, para a tarefa de classificação, o modelo determina a qual lado do hiperplano um novo ponto de dados pertence.

Entre os principais pontos fortes do SVM, temos: eficácia em espaços de alta dimensão, funcionando bem quando o número de características é muito maior que o número de amostras; robustez contra *overfitting*, pois a maximização da margem atua como um parâmetro de regularização; versatilidade através de diferentes funções de kernel, permitindo adaptar o algoritmo a diversos tipos de problemas. Quanto aos pontos, negativos temos: complexidade computacional elevada, especialmente para grandes conjuntos de dados; necessidade de seleção cuidadosa de hiperparâmetros como o tipo de *kernel* e seus respectivos parâmetros; e menor interpretabilidade comparada a algoritmos mais simples, especialmente quando utiliza *kernels* não-lineares.

3.1.3. Extreme Gradient Boosting

O algoritmo XGBoost (Extreme Gradient Boosting) surgiu como uma implementação otimizada e avançada do *gradient boosting*, combinando múltiplas árvores de decisão para criar modelos preditivos de alta performance. A primeira etapa do algoritmo é a construção sequencial de árvores, onde cada nova árvore é treinada para corrigir os erros das árvores anteriores através do método de *gradient boosting*, que otimiza a função de perda por meio do cálculo do gradiente. Em seguida, o algoritmo utiliza um processo iterativo onde inicialmente constrói um modelo simples, faz previsões nos dados de treinamento, calcula os erros dessas previsões e constrói outra árvore para modelar e corrigir esses erros. Por fim, diferente de implementações tradicionais de *boosting*, o XGBoost permite o processamento paralelo durante a construção das árvores, seguindo uma estratégia *level-wise* que avalia a qualidade de *splits* em todos os possíveis pontos de divisão. Com o treinamento finalizado, as previsões de todas as árvores são somadas para gerar as previsões finais, resultando em um modelo mais robusto e preciso.

Entre os principais pontos fortes do XGBoost, temos: alta precisão e performance, especialmente com dados estruturados, produzindo resultados excepcionais em uma variedade de problemas de modelagem preditiva; eficiência computacional, sendo otimizado para velocidade e desempenho com suporte à paralelização; regularização integrada, incluindo técnicas L1 (*lasso*) e L2 (*ridge*) para prevenir *overfitting*; e tratamento inteligente de valores ausentes, onde o algoritmo aprende automaticamente a melhor direção para enviá-los em cada divisão da árvore. Quanto aos pontos negativos, temos: complexidade

no ajuste de parâmetros, exigindo configuração cuidadosa de diversos hiperparâmetros para obter desempenho ótimo; risco de *overfitting* quando não devidamente regulado, especialmente com conjuntos de dados pequenos; e alto consumo de memória, pois o algoritmo pode requerer carregar o conjunto de dados inteiro na memória múltiplas vezes durante o treinamento.

3.2. Aprendizado Profundo

Mudando levemente de abordagem, temos os algoritmos de aprendizado profundo, algoritmos que nada mais são do que redes neurais artificiais com múltiplas camadas que são usadas para processar e analisar dados complexos em grande quantidade. Aprendizado profundo geralmente está associado a um custo computacional extremamente elevado, com o treinamento de tais redes podendo demorar dias no melhor dos hardwares, dependendo da complexidade da tarefa. Estes algoritmos são particularmente eficazes em tarefas complexas como processamento de linguagem natural, reconhecimento de fala e no nosso caso reconhecimento de imagens.

O modelo escolhido para esse trabalho foi a YOLO (You Only Look Once), o modelo usa uma rede neural convolucional para processar a imagem inteira em uma única passagem. Para isso, a imagem é dividida em células onde cada célula é responsável por prever múltiplas caixas delimitadoras (*bounding boxes*) com suas respectivas pontuações de confiança. Para cada caixa, o algoritmo também prevê a probabilidade de classes específicas, combinando esses valores em uma pontuação final que indica a probabilidade da caixa conter determinado objeto. Após as previsões, é aplicado um processo de supressão não-máxima para eliminar caixas redundantes.

Dentro dos modelos YOLO, existem dois focos principais, sendo eles: detecção de objetos, onde o modelo procura por classes por toda a imagem; e classificação da imagem como um todo, foco esse que será o nosso. Mais especificamente utilizamos a Yolo11n-cls disponível na biblioteca Ultralytics, os modelos YOLO presentes na Ultralytics são o atual estado da arte, utilizamos a versão 'n' por ser a mais leve entre os modelos da família, e por consequência a que seria mais adequada para o hardware disponível.

Tabela 1. Comparação de modelos YOLO11 - Métricas principais

Modelo	tamanho (pixéis)	acc top1	acc top5	params (M)
YOLO11n-cls	224	70.0	89.4	1.6
YOLO11s-cls	224	75.4	92.7	5.5
YOLO11m-cls	224	77.3	93.9	10.4
YOLO11l-cls	224	78.3	94.3	12.9
YOLO11x-cls	224	79.5	94.9	28.4

Fonte: Adaptado de "YOLOv11: An Overview of the Key Architectural Enhancements".

Tabela 2. Comparação de modelos YOLO11 - Desempenho e complexidade computacional

Modelo	Velocidade CPU ONNX (ms)	Velocidade T4 TensorRT10 (ms)	FLOPs (B) em 224
YOLO11n-cls	5.0	1.1	0.5
YOLO11s-cls	7.9	1.3	1.6
YOLO11m-cls	17.2	2.0	5.0
YOLO11l-cls	23.2	2.8	6.2
YOLO11x-cls	41.4	3.8	13.7

Fonte: Adaptado de "YOLOv11: An Overview of the Key Architectural Enhancements".

4. Metodologia

4.1. Dataset

O Parkinson Yolo Dataset, disponível em `https://www.kaggle.com/datasets/cornelioac/parkinson-yolo-dataset/data`, é composto por imagens de espirais e ondas desenhadas à mão, coletadas tanto de indivíduos saudáveis quanto de pacientes diagnosticados com a doença de Parkinson. O objetivo do conjunto de dados é fornecer uma base visual para o desenvolvimento e avaliação de modelos de inteligência artificial capazes de distinguir padrões gráficos associados à presença do Mal. As imagens foram preparadas para aplicações em visão computacional, especialmente em tarefas de detecção e classificação utilizando algoritmos como o YOLO. A diversidade de desenhos permite que os modelos aprendam a identificar características motoras específicas, como tremores ou alterações de coordenação, que são comuns em pacientes com Parkinson. Esse *dataset* é relevante por oferecer uma abordagem não invasiva e acessível para auxiliar no diagnóstico precoce da doença.

4.2. Para a Yolo

Apesar de o *dataset* afirmar estar pronto para uso com a YOLO, foi necessário realizar uma nova formatação. O motivo é que o *dataset* estava estruturado para treinamento da YOLO em tarefas de detecção de objetos, e não para classificação. Cada imagem possuía apenas uma *bounding box* ocupando toda a tela, com um único rótulo associado a essa caixa. Para o contexto de classificação, essa abordagem não é adequada, pois o modelo precisa apenas do rótulo da imagem, sem a necessidade de caixas delimitadoras. Para corrigir isso, reorganizamos a hierarquia de pastas para seguir o formato utilizado pelo Ultralytics para treinamento de classificação. Essa reorganização foi realizada após a aplicação do K-Folds, facilitando assim o uso do algoritmo de validação cruzada.

O treinamento dos modelos YOLO para classificação foi realizado utilizando validação cruzada com 5 folds, garantindo uma avaliação robusta do desempenho. Para cada *fold*, o cache da GPU era limpo antes do início do treinamento, otimizando o uso de memória. O YOLO11n-cls foi carregado e treinado por 64 épocas, com *batch size* de 16 e imagens redimensionadas para 512x512 *pixels*. Algumas técnicas de aumento de

dados, como *mosaic* e *copy-paste*, foram aplicadas para melhorar a generalização. Após o treinamento de cada *fold*, o modelo era avaliado e as principais métricas, como acurácia top-1 e a matriz de confusão, eram salvas para análise posterior. No total, o treinamento de todos os *folds* levou cerca de 30 minutos no hardware disponível, um notebook equipado com processador i5 de 13ª geração e GPU RTX 3050 Mobile. Esse procedimento sistemático permitiu uma comparação justa entre os *folds* e forneceu uma visão detalhada do desempenho do modelo em diferentes subconjuntos do conjunto de dados.

A validação foi conduzida de maneira criteriosa, utilizando as matrizes de confusão salvas durante o treinamento de cada *fold*. Para cada *fold*, foram calculadas as principais métricas de avaliação, como precisão, *recall* e *F1-score*, considerando as médias macro, micro e ponderada (*weighted*), além da acurácia. Essas métricas foram extraídas a partir dos elementos da matriz de confusão, permitindo uma análise detalhada do desempenho do modelo em cada classe e no conjunto geral. Ao final, os resultados de todos os *folds* foram agregados, possibilitando o cálculo das médias e desvios padrão para cada métrica. Esse processo garantiu uma avaliação abrangente e confiável, destacando tanto o desempenho global quanto a consistência entre os diferentes *folds* da validação cruzada.

4.3. Para os Métodos Clássicos

Inicialmente, foi realizado um pré-processamento dos dados extraídos das imagens espiraladas e senoides, transformando-os em vetores numéricos de atributos para posterior utilização em modelos clássicos de aprendizado de máquina. As *features* foram cuidadosamente escolhidas para capturar diferentes dimensões da imagem: geometria global (como área, centroide e compacidade); estatísticas de intensidade (desvio padrão e entropia); e textura local (descritores HOG e LBP). Cada uma dessas características contribui com informações distintas: enquanto a área e a compacidade refletem aspectos morfológicos; a regularidade do traçado e a posição do centroide fornecem dados sobre a distribuição espacial. Por outro lado, o desvio padrão e a entropia representam a variabilidade e a complexidade visual, enquanto HOG e LBP permitem capturar padrões sutis de textura e orientação — todos fatores que podem ser afetados por alterações motoras em pacientes com Parkinson. Após a extração dos atributos, a redução de dimensionalidade via PCA foi avaliada como forma de preservar as componentes mais relevantes dos dados, minimizando redundância e ruído.

Para a tarefa de classificação, foram utilizados os três algoritmos supervisionados apresentados na seção de Fundamentação Teórica: Random Forest, SVM e XGBoost. Esses modelos foram escolhidos por sua complementaridade: enquanto o Random Forest é robusto a dados ruidosos e fornece interpretabilidade; o SVM é eficaz em espaços de alta dimensionalidade; e o XGBoost oferece ótimo desempenho com regularização e capacidade de modelar interações complexas entre atributos. Todos os modelos foram treinados tanto na versão original dos dados quanto na versão com PCA, para verificar os ganhos em desempenho e tempo computacional decorrentes da redução de dimensionalidade.

A validação dos modelos foi realizada utilizando validação cruzada estratificada com cinco dobras (*5-fold stratified cross-validation*), garantindo a manutenção da proporção entre classes em cada subdivisão. Para a escolha dos melhores hiperparâmetros, foi utilizada a biblioteca Optuna, que automatiza o processo de otimização com base em busca bayesiana. A métrica principal de avaliação foi a acurácia, comple-

mentada por precisão, *recall* e *F1-score*, a fim de captar diferentes aspectos do desempenho classificatório, especialmente em contextos de classes desbalanceadas.

5. Resultados

Nesta seção, vamos apresentar os resultados apresentados pelas diferentes estratégias e algoritmos, bem como uma discussão e comparação.

5.1. Apresentação dos resultados

5.1.1. Métodos Clássicos

Tabela 3. Comparação de Algoritmos de Classificação Clássicos

Tipo de Classificação	Redução de Dimensionalidade	Acurácia Média	F1-score Médio
Random Forest Multiclasse	Não	0.8033 ± 0.0373	0.7946 ± 0.0420
	Sim	0.7752 ± 0.0406	0.7596 ± 0.0474
Random Forest Binário	Não	0.8072 ± 0.0131	0.8051 ± 0.0142
	Sim	0.7951 ± 0.0266	0.7935 ± 0.0269
SVM Multiclasse	Não	0.7966 ± 0.0211	0.7944 ± 0.0207
	Sim	0.7925 ± 0.0128	0.7863 ± 0.0144
SVM Binário	Não	0.8166 ± 0.0190	0.8146 ± 0.0194
	Sim	0.8274 ± 0.0200	0.8262 ± 0.0203
XGBoost Multiclasse	Não	0.8046 ± 0.0367	0.7980 ± 0.0404
	Sim	0.7872 ± 0.0390	0.7786 ± 0.0427
XGBoost Binário	Não	0.7925 ± 0.0243	0.7914 ± 0.0240
	Sim	0.7912 ± 0.0201	0.7898 ± 0.0206

Fonte: Elaborado pelos autores.

Tabela 4. Resumo do impacto da redução de dimensionalidade em cada modelo e tarefa

Modelo	Tipo de classificação	Delta Acurácia	Delta F1-score
Random Forest	Multiclasse	-0,0281	-0,0350
Random Forest	Binário	-0,0121	-0,0116
SVM	Multiclasse	-0,0041	-0,0081
SVM	Binário	+0,0108	+0,0116
XGBoost	Multiclasse	-0,0174	-0,0194
XGBoost	Binário	-0,0013	-0,0016

Fonte: Elaborado pelos autores.

Nota: Valores positivos indicam melhora no desempenho com a redução de dimensionalidade; valores negativos indicam piora.

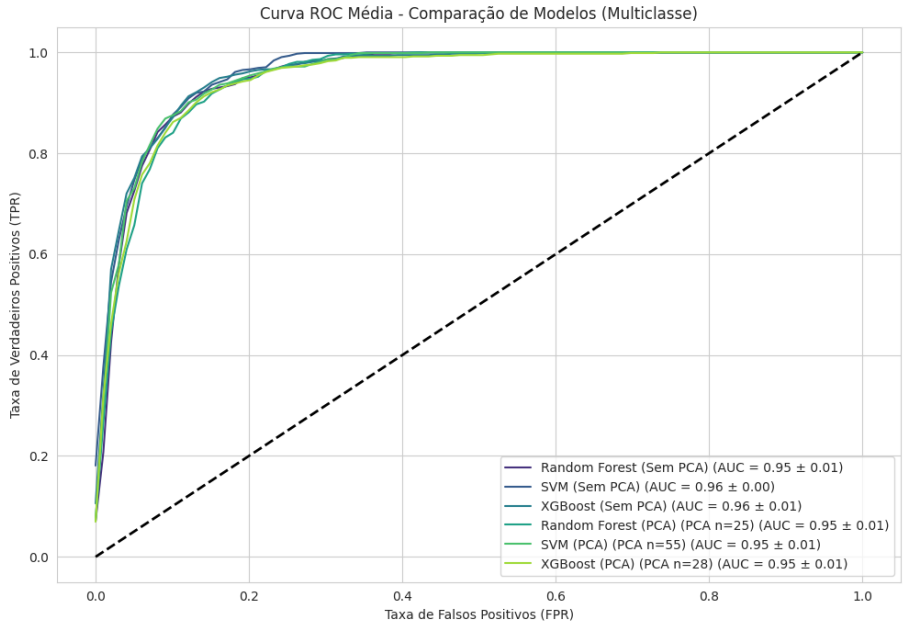

Figura 1. Curva ROC Média multiclasse

5.1.2. Yolo

Tabela 5. Métricas de Desempenho do Modelo

Métrica	Média	Desvio Padrão
Acurácia	0.882	0.036
Macro Precision	0.850	0.035
Weighted Precision	0.895	0.040
Micro Precision	0.882	0.036
Macro Recall	0.879	0.043
Weighted Recall	0.882	0.036
Micro Recall	0.882	0.036
Macro F1-score	0.858	0.033
Weighted F1-score	0.884	0.037
Micro F1-score	0.882	0.036

Fonte: Elaborado pelos autores.

5.2. Discusão e Comparação

Nos modelos clássicos, a análise comparativa entre classificação binária e multiclasse revela diferenças significativas no desempenho dos modelos avaliados. A classificação binária demonstrou superioridade consistente em termos de acurácia e F1-score em praticamente todos os algoritmos testados, resultado esperado dado que problemas binários envolvem decisões mais simples comparados aos multiclasse. O SVM foi o modelo que mais se beneficiou da binarização, apresentando melhorias substanciais na acurácia (de 0.7966 para 0.8274) e F1-score (de 0.7944 para 0.8262) quando combinado com PCA.

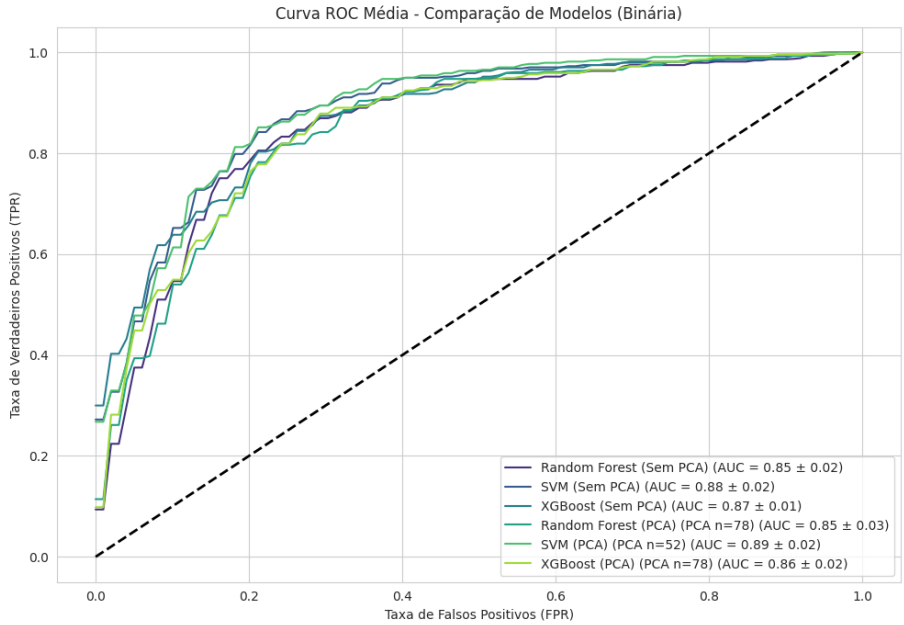

Figura 2. Curva ROC Média Binária

Quanto ao impacto da redução de dimensionalidade, o PCA mostrou efeitos variados dependendo do algoritmo: enquanto prejudicou o desempenho de modelos baseados em árvores como Random Forest e XGBoost - possivelmente por eliminar variáveis importantes ou criar representações menos interpretáveis - beneficiou significativamente o SVM na classificação binária, evidenciando que este algoritmo se favorece de dados mais compactos e com menor ruído. O SVM binário com PCA emergiu como a configuração de melhor desempenho geral (acurácia 0.8274, F1-score 0.8262, AUC 0.89), enquanto para tarefas multiclasse, o XGBoost sem PCA apresentou os melhores resultados.

O modelo YOLO apresentou uma acurácia média de 0.882 entre os folds, com o melhor fold chegando a 0.929, enquanto o máximo registrado por um algoritmo clássico foi de 0.827. Nas outras métricas, o modelo também desempenhou muito bem, com uma média de 0.6 pontos a mais que os modelos clássicos. Tudo isso com um tempo baixo de treinamento, cerca de 30 minutos, em um hardware relativamente simples, considerando o padrão moderno para treinamento de modelos de IA. Com isso podemos afirmar uma grande diferença positiva de desempenho por parte da YOLO para classificação, justificando assim seu custo computacional elevado.

6. Conclusão

Em suma, os resultados apresentados revelam que o modelo YOLO11 apresentou performance superior quando comparado aos algoritmos tradicionais de ML na identificação de Parkinson através de análise de imagens. Com desempenho médio de 88,2% de acurácia, com valores máximos atingindo 92,9%, o algoritmo de aprendizado profundo apresenta clara vantagem quando comparado com os melhores resultados dos algoritmos tradicionais, que não passaram de 82,% de precisão.

Considerando a área médica, particularmente no contexto da doença de Parkinson, onde a detecção precoce representa fator importantíssimo para o tratamento, torna-se

evidente que os custos computacionais mais elevados, necessários para a implementação de redes neurais profundas, são justificáveis. A diferença observada de aproximadamente seis pontos percentuais entre as abordagens representa, em termos práticos, uma melhora substancial no número de casos corretamente identificados, fator que pode influenciar decisivamente o estabelecimento de protocolos terapêuticos apropriados e intervenções clínicas oportunas.

Cabe ressaltar que os resultados obtidos através da implementação do YOLO11n-cls, representando a versão mais compacta desta família de algoritmos, indicam possibilidades ainda mais promissoras mediante a disponibilidade de recursos computacionais mais robustos. A utilização de variantes mais sofisticadas, como YOLO11m-cls, YOLO11l-cls ou YOLO11x-cls, poderia elevar substancialmente a precisão diagnóstica. Adicionalmente, o emprego de GPUs mais avançadas poderia provavelmente aprimorar a acurácia e diminuir consideravelmente os períodos de processamento, viabilizando a implementação clínica em escala ampliada.