Explorando Predição da Caracterização Elétrica com Machine Learning

Gabriel Lima Jacinto
gabriellimajacinto@gmail.com
Universidade Federal de Santa Catarina
Florianopolis, Santa Catarina, Brazil

Lucas Yuki Imamura
Universidade Federal de Santa Catarina
Florianopolis, Santa Catarina, Brazil
lucasyuki@yahoo.com

Mateus Grellert
Universidade Federal de Santa Catarina
Florianopolis, Santa Catarina, Brazil
mateus.grellert@gmail.com

Cristina Meinhardt
Universidade Federal de Santa Catarina
Florianopolis, Santa Catarina, Brazil
cristina.meinhardt@ufsc.br

ABSTRACT

With the advancement of integrated circuit manufacturing technology, more and more aspects must be considered during the electrical characterization of circuits in order to solve challenges such as process variability effect. This increases the characterization time due to traditional techniques based on exhaustive electrical simulations. The adoption of machine learning techniques already helps digital design at many levels of abstraction. Thus, the main objective of this research is to evaluate machine learning regression algorithms as an alternative to exhaustive electrical simulation in the cell characterization project. In this step, multiple linear regression, support vector regression, decision trees and random forest algorithms were considered. This work presents the results of NAND2 and NOT gates using *bulk* CMOS technology. Specifically, the energy values and the propagation times of this circuit will be predicted separately. A comparative analysis, together with the inference time, is made for each dependent variable between the models, in order to understand which is the best regression model for the task. The algorithm with the lowest cost function and shortest inference time proved to be the decision tree for all predicted variables in both gates.

KEYWORDS

Nanotecnologia, Caracterização Elétrica, Machine Learning, Variabilidade

1 INTRODUÇÃO

O constante avanço no processo de fabricação de transistores permite o aumento da capacidade de integração de funcionalidades em um mesmo projeto, com melhores resultados de desempenho devido aos transistores serem mais rápidos, consumirem menos energia e ocuparem menor área. Entretanto, esta evolução também insere novos desafios, principalmente ao projetarmos circuitos integrados em tecnologias nanométricas. Um dos maiores desafios nas tecnologias em nodos nanométricos é a variabilidade do processo. A variabilidade do processo afeta o comportamento normal esperado das células, alterando o atraso e o consumo de energia observados em condições padrões. Juntamente com os efeitos de variabilidade de processo, soma-se o aumento da complexidade das regras de projeto. Consequentemente, as ferramentas de projeto de circuitos integrados precisam resolver problemas cada vez mais difíceis. Esse problema vem crescendo exponencialmente, e essas dificuldades refletem diretamente no custo de desenvolvimento de projetos de dispositivos eletrônicos [1].

Assim, a caracterização da célula passa a incluir vários pontos, além dos tradicionais pontos que observam as características elétricas considerando os dispositivos mais rápidos, mais lentos e em condições nominais. Para explorar uma grande variedade de pontos de caracterização possíveis em uma tecnologia alvo, tradicionalmente, os projetistas adotam simulações elétricas exaustivas para todos pontos, observando o impacto nas métricas de atraso e consumo de energia. Assim, aumentar o número de simulações elétricas para cobrir esses pontos pode se tornar uma tarefa de demanda de grande tempo.

Técnicas de aprendizado de máquina podem ser boas alternativas na tentativa de reduzir o tempo de caracterização e os custos para o desenvolvimento de dispositivos eletrônicos. A revisão bibliográfica mostra que ainda há muito espaço a ser explorado quanto ao uso de aprendizado de máquina para ferramentas na área da microeletrônica. Seu uso atual está fortemente ligado à previsão de qualidade de soluções candidatas dadas por ferramentas existentes, ou tentando guiá-las para uma solução de melhor qualidade, por meio de previsões métricas [1].

Com a disponibilidade crescente de grandes quantidades de dados de manufatura e de melhoria de processo, técnicas de aprendizado de máquina podem ser adotadas para analisar, classificar e prever a qualidade durante a etapa de manufatura que realiza a gravura dos metais no circuito [2] ou investigar os efeitos da variabilidade do processo em células de memória *flash* do tipo 3DNAND [3]. Nas etapas de posicionamento e roteamento de circuitos, encontramos trabalhos que usam aprendizado de máquina para prever alterações na caracterização de atrasos dos circuitos baseada em caminho de análise de tempo adotando grafos [4] e preditores de tempo de pré-roteamento [5].

Além disso, o trabalho [6] explora aprendizado de máquina na caracterização elétrica de dispositivos. No entanto, o trabalho aborda caracterização especificamente para dispositivos magnéticos. Assim, salienta-se que há espaço para explorar também algoritmos de aprendizado de máquina para caracterização elétrica de células lógicas em tecnologias de *bulk* CMOS (*Complementary Metal Oxide Semiconductor*), SOI (*Silicon On Insulator*) ou FinFET (*Fin Field-Effect*).

A caracterização de confiabilidade de células combinacionais e sequenciais é tradicionalmente baseada em simulações elétricas exaustivas, elevando o tempo e custo de projeto. Observando a crescente adoção das técnicas de aprendizado de máquina na área de microeletrônica, este trabalho analisa a aplicação destas técnicas como alternativa para substituir os tradicionais métodos de simulação de efeitos de falhas de radiação e de variabilidade de processo.

Desta forma, este trabalho tem por objetivo explorar técnicas de aprendizado de máquina para prever o comportamento elétrico de circuitos. Em trabalhos anteriores, considerou-se como um estudo de caso o circuito inversor CMOS, com o alvo de predição a energia e o tempo de propagação [7]. Objetivando-se expandir e testar a técnica desenvolvida, este trabalho explora mais uma porta CMOS: a porta "não e" (NAND2). Levando em consideração os melhores modelos dos experimentos feitos com a porta inversora, procura-se responder a questão se os quatro métodos de regressão apresentam o mesmo comportamento de predição para outra função lógica. Assim, são avaliados os métodos de Árvores de Decisão (DT), Floresta Aleatória (RF), Regressão Linear Múltipla (MLR) e Regressão de Vetores de Suporte (SVR). As variáveis de entrada consideradas foram a tensão, a capacitância, a temperatura e a variabilidade do processo. A metodologia de predição apresentada neste artigo será estendida para considerar outros circuitos e tecnologias em futuros trabalhos.

Na Seção 2, discute-se sobre o referencial teórico básico dos algoritmos de aprendizado de máquina que foram utilizados, juntamente com a explicação dos cálculos matemáticos que são feitos nas predições dos modelos. Na Seção 3, explicam-se os métodos utilizados para se extrair os dados que treinaram os algoritmos, assim como as técnicas aplicadas em suas modelagens para evitar problemas de *overfit* e melhorar a performance dos modelos tomando como base a função de custo escolhida. Logo em seguida, na Seção 4, os resultados dos treinamentos são discutidos tanto por meio da análise dos erros da predições quanto pela representação gráfica desses erros. Apresentam-se também os resultados comparativos de tempo para averiguar se o método desenvolvido é realmente mais rápido do que as tradicionais simulações exaustivas. Por fim, na Seção 5, sintetiza-se todo o processo realizado e os resultados obtidos em uma conclusão que indica qual foi o melhor algoritmo e o pior algoritmo para o objetivo posto nesse trabalho.

2 ALGORITMOS DE APRENDIZAGEM SUPERVISIONADA - REGRESSÃO

Existem vários algoritmos de aprendizado de máquina aplicáveis para diferentes tarefas, que vão desde a previsão de valores no mercado de ações até a navegação de um robô em Marte. Uma vez que, neste trabalho, objetiva-se a previsão da energia e do atraso de circuitos - que são todos valores contínuos - a abordagem será usar uma técnica de aprendizagem supervisionada conhecida como regressão, que calcula a correlação entre as variáveis de entrada [8]. Existem muitos algoritmos de regressão, portanto, delimitou-se este estudo a quatro técnicas clássicas.

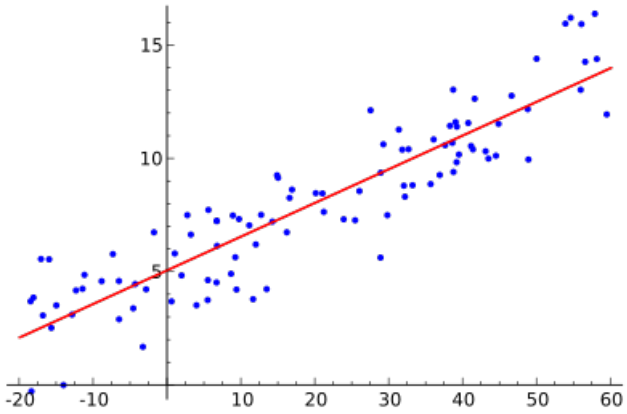

Figura 1: Visualização de Regressão Linear Simples

2.1 Regressão linear múltipla

A regressão linear é uma técnica estatística amplamente usada quando se deseja prever o comportamento de um valor alvo. Como neste trabalho as variáveis de saída são previstas com base nos valores de muitas variáveis de entrada, usou-se a regressão linear múltipla (MLR) em vez da regressão linear simples (Figura 1). O valor predito $\hat{y}$ é descrito na Eq. 1, onde θ_0 é o termo de interceptação e $\theta_1 ... \theta_n$ são os pesos de cada valor das variáveis de entrada [8]. O que se deseja é encontrar os valores de cada θ_n que minimizem a função de custo do modelo [8], que neste trabalho será a raiz do erro quadrático médio (RSME).

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n \quad (1)$$

2.2 Regressão de vetores de suporte

Embora *Support Vector Machines* (SVM) sejam normalmente aplicados a problemas de classificação, trata-se de um algoritmo versátil que também pode realizar regressões, denominado *Support Vector Regression* (SVR) [8]. A ideia é que, dado um valor ϵ, o SVR calcula uma margem em torno do hiperplano da função de previsão, onde tenta-se ajustar o máximo de observações possível entre a área definida pelas margens e reduzir os valores que são fora dessa mesma área (Fig. 2).

A função de decisão representada pela Eq. 3 é calculada de forma semelhante à regressão linear, entretanto, aqui se colocou a equação em uma forma vetorizada [8], que é a prática mais comum quando se representa algoritmos de aprendizado de máquina. Portanto temos que o valor predito é representado por $\hat{y}$, em que w^T é a matriz transposta de pesos para cada variável de entrada x (os valores $\theta_1 ... \theta_n$ da Eq. 1) e b é o valor de interceptação. O SVR tem muitos *kernels* (linear, RBF, polinomial) que são funções capazes de realizar o produto escalar $\phi(a)^T \phi(b)$ apenas usando os valores de a e b, sem necessariamente calcular a transformação ϕ [8]. Neste trabalho foi usado o *kernel* RBF representado pela Eq. 2.

$$K(a, b) = e^{-\gamma ||a-b||^2} \quad (2)$$

$$\hat{y} = w^T x + b \quad (3)$$

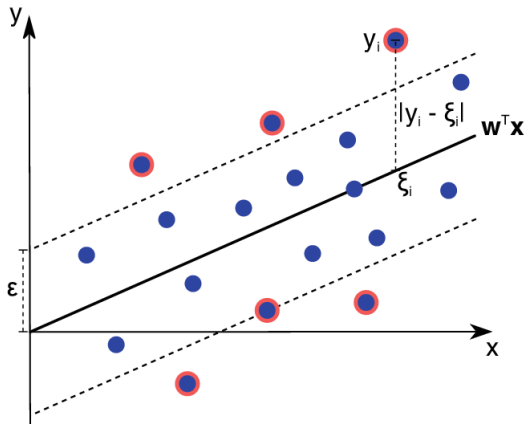

Figura 2: Visualização de *Support Vector Regression*

2.3 Árvores de Decisão

A árvore de decisão (DT - *Decision Tree*) é outro algoritmo mais utilizado em problemas de classificação. Porém, também é possível utilizá-lo em tarefas de regressão [8]. O algoritmo funciona criando uma estrutura em árvore com nós internos de testes que são feitos para prever a qual nó folha a observação será atribuída, retornando, no caso desse trabalho, um valor contínuo. Na Fig. 3 é mostrado como esse algoritmo realiza sua regressão. Divide-se os valores em setores e tenta-se fazer a previsão se baseando nesses setores criados. Esses setores são criados a partir da configuração de hiperparâmetros do modelo, como por exemplo a profundidade máxima da árvore (*"max depth"*).

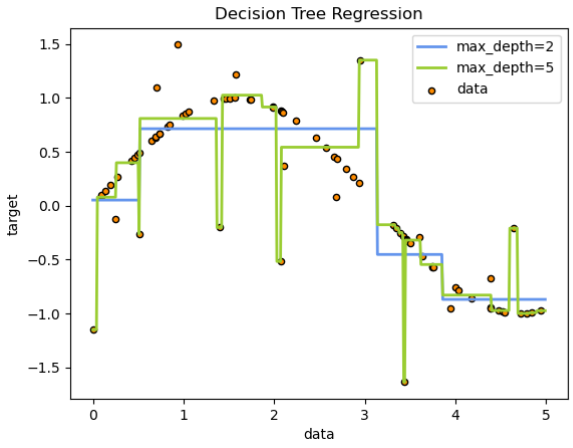

Figura 3: Visualização de *Decision Trees*

2.4 Floresta Aleatória

A floresta aleatória, do inglês *Random Forest* (RF), é um algoritmo semelhante às árvores de decisão, a diferença é que ao invés de computar apenas uma árvore de decisão, a RF calcula um grande número de árvores de decisão com profundidades diferentes para, em seguida, computar a média de todas as previsões para estimar o valor. Isso torna esse método um dos algoritmos supervisionados mais poderosos. Pode-se representar as árvores de decisão n unidas conforme descrito na Eq. 4, onde cada $f_n(x)$ representa um estimador (uma árvore) da floresta como na Fig. 4.

$$g(x) = f_0(x) + f_1(x) + f_2(x) + f_3(x) + ... + f_n(x) \quad (4)$$

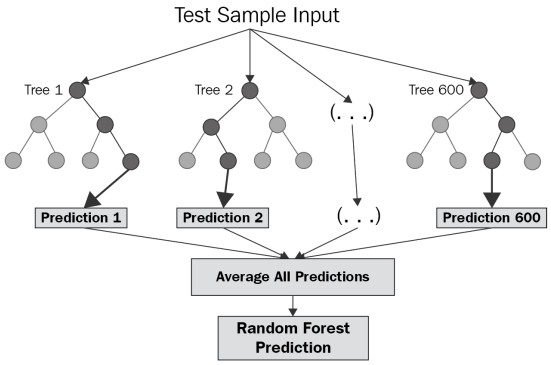

Figura 4: Visualização de Floresta Aleatória

3 METODOLOGIA

A seguir, este relatório detalha os materiais e métodos adotados para inicialmente formular o fluxo para caracterização de uma célula NAND2 e NOT, adotando técnicas de aprendizagem de máquina. Na Fig. 5 apresenta-se um diagrama que resume o processo de treinamento realizado. Assim, divide-se o processo em sete etapas principais iniciando com as simulações elétricas para extração das variáveis, seguindo pelo tratamento destas variáveis, realizando a normalização e também a separação destes dados em conjuntos de treinamento, validação e teste para os algoritmos avaliados. A Etapa quatro contempla o desenvolvimento e ajuste dos algoritmos considerados. A Etapa cinco realiza a análise dos algoritmos. Finalmente, é realizada a compilação dos resultados e também a disponibilização em uma interface online do modelo treinado. Os passos envolvidos neste fluxo serão detalhados nas próximas seções.

3.1 Extração de Variáveis

A caracterização elétrica de um circuito prevê a obtenção de dados de energia e dos atrasos. Os tempos de propagação de subida (Tp_{HL}) e descida (Tp_{LH}) são utilizados para definir os atrasos dos circuitos. Estas variáveis sofrem a influência de diversos parâmetros de configuração do circuito tais como o tamanho dos dispositivos, a largura de canal, a capacitância de carga, a temperatura, a tensão de operação, e os efeitos de variabilidade impactam esses dados também dependendo dos parâmetros adotados. Para prever o Tp_{HL}, o Tp_{LH} e a energia do circuito NAND2 CMOS, o primeiro passo foi definir claramente quais são as variáveis de entrada e seus valores. A Tabela 1 resume essas informações. Foram simulados os valores de nove variáveis de entrada: a tensão de limiar para dispositivos NMOS e PMOS pelo parâmetro Vth0 no modelo do dispositivo, a

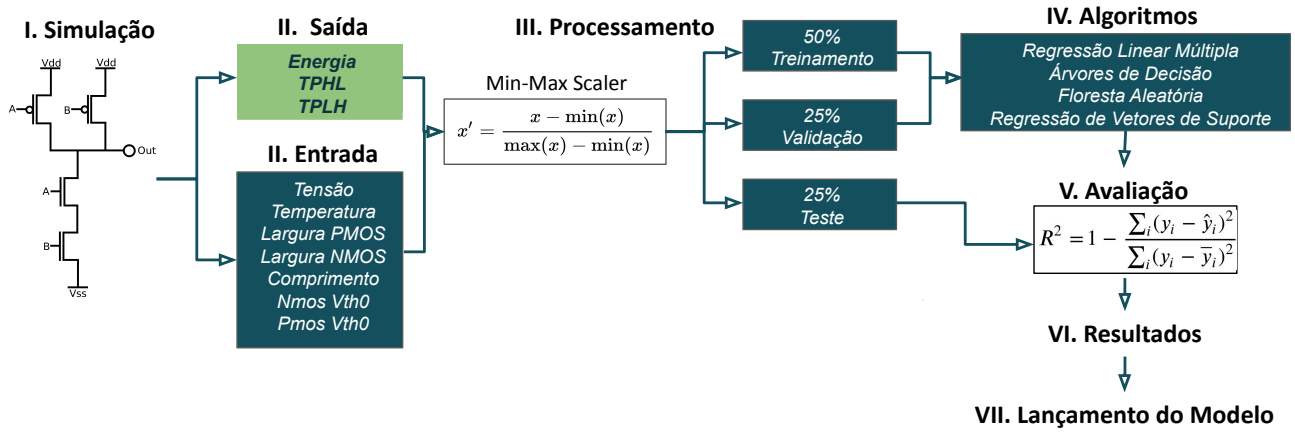

Figura 5: Diagrama do fluxo de treinamento

temperatura, a capacitância, a tensão de operação, a largura dos transistores (W) para dispositivos PMOS e NMOS e o comprimento do canal (L) para dispositivos PMOS e NMOS.

Tabela 1: Variáveis de entrada e de saída

Saída	Tp_HL	Tp_LH	Energia
Entrada	NMOS Vth0 PMOS Vth0 temperatura tensão W PMOS W NMOS L capacitância	NMOS Vth0 PMOS Vth0 temperatura tensão W PMOS W NMOS L capacitância	NMOS Vth0 PMOS Vth0 temperatura tensão W PMOS W NMOS L capacitância

Adotou-se o modelo de transistor para a tecnologia CMOS Bulk de alto desempenho de 16 nm fornecido pelo PTM [9]. O parâmetro Vth0 é variado para simular o impacto da variabilidade do processo devido à variabilidade geométrica e variabilidade aleatória de dopante na tensão de limiar do transistor. A variabilidade de processo é simulada utilizando o método de Monte Carlo, onde a alteração na tensão de limiar segue uma distribuição Gaussiana com 3σ e 10% de desvio padrão [10]. Todas as simulações elétricas adotaram o *software* de simulação Hspice da Synopsys. Predefiniu-se os valores de largura, comprimento, temperatura, capacitância e tensão, seguindo estas variações:

- **Temperatura (^oC)**: −25, 0, 25 50, 75 e 100
- **Largura PMOS (nm)**: 70, 140, 210
- **Largura NMOS (nm)**: 70 e 140, 210, 280, 340, 420
- **Comprimento do PMOS e NMOS (nm)**: 32 e 40
- **Tensão (V)**: 0.6, 0.7, 0.8 e 0.9
- **Capacitância (F)**: 1, 4, 8 e 16

Os valores da energia e dos tempos de propagação (Tp_HL e Tp_LH) foram todos obtidos pelas simulações elétricas com base em todas essas variáveis. No final, foram feitas 80 simulações transientes de 20 ns com um passo de 0,1 ns, obtendo-se ao final um total de 276.480 observações para a porta NAND2. Já para a porta inversora, as mesmas configurações de simulação foram aplicadas, com a diferença de que ao invés de 80 simulações, foram feitas 1000 simulações de Monte Carlo, resultando em 168.000 observações.

No entanto, devido à remoção de valores discrepantes que confundiriam os resultados dos modelos, o número final de observações ficou em 271.863 para a NAND2 e 120.954 para a NOT.

3.2 Normalização das variáveis

Os resultados de energia e de atrasos para a tecnologia de 16 nm são na ordem de picosegundos e fentoJoules. Por isso, foi preciso fazer um redimensionamento em cada variável, já que, além de serem muito pequenas, alguns algoritmos (como o SVR) são sensíveis a essas escalas devido à forma como seu cálculo é realizado. Para esse redimensionamento, usou-se o método conhecido como normalização, que dimensiona os dados para o intervalo [0,1]. Matematicamente, é adotada a Eq. 5, na qual se subtrai do valor X o valor mínimo e divide-se esse resultado pela diferença entre o máximo e o mínimo.

$$X' = \frac{X - min(X)}{max(X) - min(X)} \tag{5}$$

3.3 Treinamento dos Algoritmos

A abordagem para prever as três variáveis de saída (Tp_{HL}, Tp_{LH}, energia) foi criar todos os quatro modelos para cada uma delas (Regressão Linear Múltipla, Regressão de Vetores de Suporte, Árvores de Decisão e Floresta Aleatória). Assim, ao final, foram treinados 12 modelos para cada porta. O módulo de *model selection* da biblioteca *sci-kit learn* da linguagem *python* foi utilizado para dividir o conjunto de dados entre o grupo de treinamento (50%), de validação (25%) e o de teste (25%). As ferramentas utilizadas para manipular, visualizar, limpar, treinar e testar os dados foram o Colaboratory do Google e as bibliotecas da linguagem python sklearn, pandas, NumPy, matplotlib e seaborn. Todos os algoritmos foram treinados utilizando a técnica conhecida como *cross validation* (validação cruzada) com cinco *folds*. Ela é realizada para que o algoritmo faça previsões mais gerais e não tenha um *overfit* ou enviesamento nos dados que foram alimentados durante seu treinamento. Para isso, o *cross-validation* utiliza do conjunto de treinamento e do conjunto de validação para primeiro fazer avaliação de como o modelo está se comportando. Além disso, nesse processo criam-se diversas cópias do modelo que está sendo treinado e seus hiperparâmetros são variados para averiguar qual é o melhor modelo com a melhor configuração desses hiperparâmetros. Nos modelos trabalhados, apenas a Regressão Linear Múltipla não possui hiperparâmetros, logo os seguintes valores foram configurados para o *cross-valdation*:

- **Árvore de Decisão (DT) - *Max Depth***: 1, 5, 10, 25, 50
- **Floresta Aleatória (RF) - *Max Depth***: 1, 5, 10, 25, 50
- **Floresta Aleatória (RF) - *N Estimators***: 5, 25, 50, 100, 150

Tanto para a Árvore de Decisão quanto para a Floresta Aleatória, *max depth* é a profundidade máxima que cada árvore terá. Os valores de *N Estimators* da Floresta Aleatória são quantos estimadores, ou seja, quantas árvores serão criadas para o modelo. Já os valores *Gamma* e os valores *C* para a Regressão de Vetores de Suporte são referentes ao valor γ da Eq.2 e à regularização do modelo, respectivamente. Ao final do cross validation, foram obtidos os valores da Tabela 2. Não foram colocados os valores do algoritmo SVR pois o treinamento utilizando essa metodologia provou-se muito extensa (mais de três dias) e com os recursos utilizados não foi possível gerar esses dados atualmente, portanto optou-se por seguir com o valor padrão de $C = 1$ $Gamma = \frac{1}{\sigma^2}$ para todas as variáveis de saída.

Tabela 2: Resultados da validação cruzada com 5 *folds*

	Hiperparâmetros		
Métricas	DT *Max Depth*	RF *Max Depth*	RF *N Estim.*
Tp_{HL}	5	5	25
Tp_{LH}	10	10	150
Energia	10	10	150

3.4 Medida de Performance

Para avaliar a adequação de cada modelo, precisa-se avaliar o erro entre o valor previsto e o valor desejado. Neste trabalho, usou-se a Raiz do Erro Quadrático Médio (*Root Mean Square Error* - RMSE) e o coefieciente de determinação (conhecido como "r-quadrado" ou "r-dois" - R^2) como medidas de desempenho, uma vez que o RMSE é o mais comumente usado em problemas de regressão e o R^2 é mais facilmente interpretado. O RMSE dá pesos maiores para erros grandes, o que significa que é mais sensível a outliers [8], por isso foi importante fazer a remoção desses anteriormente ao treinamento. Ele é calculado usando a norma euclidiana, conforme apresentado na Eq. 6, onde m é o número de instâncias no conjunto de dados, $x^{(i)}$ e $y^{(i)}$ são vetores de todos valores de características e o valor desejado da característica i^{th} do conjunto de dados, $\mathbf{X}$ é a matriz dos vetores de características e h é a função modelo para a previsão [8].

Figura 6: QR code da plataforma

$$RMSE(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})^2} \tag{6}$$

Já a métrica R^2 define a variância explicada, como mostra a Eq. 7. Ou seja, o R^2 diz o quão bem o modelo criado está conseguindo explicar a variação total no valor alvo e fica no intervalo entre 0 e 1. Um valor R^2 de 1 significa que o modelo foi capaz de explicar 100% do comportamento modelado, então valores baixos de R^2 significam que o modelo está mal ajustado e valores altos significam um bom ajuste do modelo [11].

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y}_i)^2} \tag{7}$$

3.5 Disponibilização do Modelo

O algoritmo com melhores resultados de aprendizado considerando o comportamento das duas portas foi disponibilizado na internet no sítio eletrônico: https://electrical-characterization.herokuapp.com (acessível também pelo QR code da Fig. 6). O site foi construído a partir da biblioteca Streamlit da linguagem python e nele é possível simular os valores desejados para que o algoritmo faça suas predições em tempo real.

4 RESULTADOS

Depois do treino e validação dos modelos em 75% dos dados, verificou-se os desempenhos no conjunto de teste a partir da função de custo RMSE e do R^2. A Tabela 3 e a Tabela 4 apresentam estes resultados para a porta NAND2 CMOS e NOT CMOS, respetivamente. Nesta Tabela, os resultados para cada modelo são apresentados descrevendo o atraso de descida Tp_LH, o atraso de subida Tp_HL e a terceira coluna apresenta os resultados de energia. Estes resultados serão discutidos pela variável alvo de predição a seguir.

A Fig. 7 apresenta os valores de R^2 de forma gráfica a fim de comparar a performance de cada porta por modelo treinado. É possível perceber a tendência de melhor perfomance dos modelos baseados em árvores e que a performance dos modelos é, em sua maioria, melhor para a porta inversora (com exceção para a variável de energia). Isso pode ser explicado devido a diferença em complexidade dos circuitos dessas portas, já que enquanto a porta NOT possui apenas dois transistores, a porta NAND2 agrega o dobro.

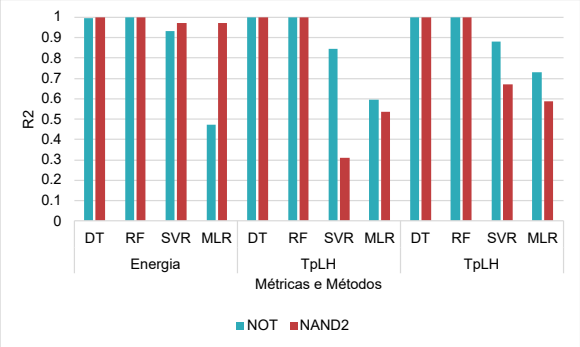

Figura 7: Comparação de desempenho de modelos por porta

Tabela 3: Resultados do RSME para Tp_HL, Tp_HL e Energia (NAND2 CMOS)

Modelo	Variáveis alvo					
	tpHL		tpLH		Energia	
	RMSE	R^2	RMSE	R^2	RMSE	R^2
RF	0.0012	0.9995	0.0009	0.9998	0.0022	0.9998
MLR	0.0395	0.5363	0.0483	0.5892	0.0354	0.9730
SVR	0.0481	0.3126	0.0431	0.6730	0.0351	0.9734
DT	0.0016	0.9991	0.0012	0.9997	0.0033	0.9997

Tabela 4: Resultados do RSME para Tp_HL, Tp_HL e Energia (NOT CMOS)

Modelo	Variáveis alvo					
	tpHL		tpLH		Energy	
	RMSE	R^2	RMSE	R^2	RMSE	R^2
RF	0.0032	0.9997	0.0028	0.9996	0.0103	0.9983
MLR	0.0956	0.5950	0.0822	0.7311	0.1778	0.4732
SVR	0.0599	0.8465	0.0545	0.8820	0.0569	0.9327
DT	0.0038	0.9996	0.0035	0.9995	0.0135	0.9969

4.1 Predição do Atraso de Propagação de Descida (Tp_{HL})

Como pode-se ver na Tabela 3 e na Tabela 4, para o Tp_HL, os maiores valores do R^2 para a NAND2 são 99.95% e 99.91% e para a NOT são 99.97% e 99.96%, que são os valores para os modelos de floresta aleatória (RF) e de árvores de decisão (DT). Este é um ótimo resultado, fato que também pode ser constatado pela Fig. 8, na qual mal pode-se ver a diferença entre a curva predita e a curva esperada (o mesmo comportamento é visto para a porta NAND2). No entanto, quando os resultados dos outros algoritmos são analisados, os valores são diferentes. Os R^2 para a Regressão Linear Múltipla (MLR) e a Regressão de Vetores de Suporte (SVR) são os menores de ambas as tabelas para o Tp_HL. Porém há uma discrepâmncia em qual foi o pior algoritmo de cada porta para essa variável de predição: enquanto o MLR foi o pior algoritmo da porta NOT (59.5%), o SVR alcançou apenas um R^2 de 31.26% na porta NAND2. Na Fig. 8 pode-se observar as distribuições das previsões de cada modelo comparadas com a distribuição correta.

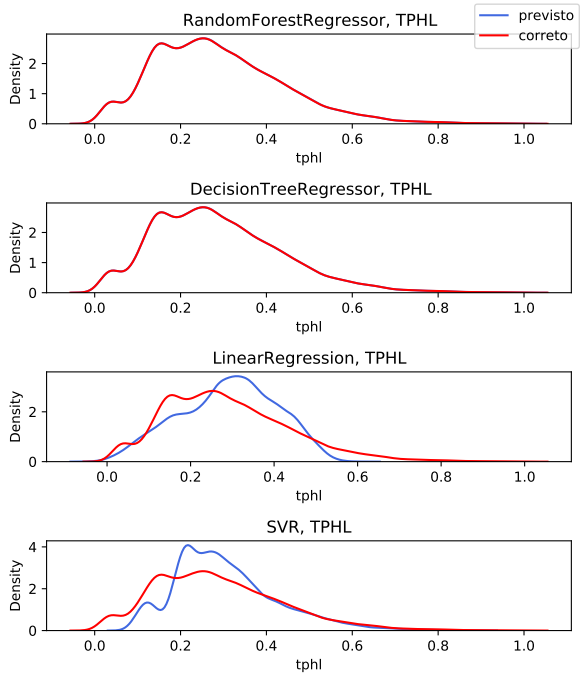

Figura 8: Predição do Atraso de Propagação de Descida Tp_{HL} para a porta NOT

4.2 Predição do Atraso de Propagação de Subida (Tp_{LH})

Os resultados para o Tp_{LH} são muito semelhantes aos do Tp_{HL}. Os números mostram que o RF ainda é o melhor método para prever a variável desejada. Dessa vez o pior algoritmo foi o MLR para as duas portas. Ademais, as semelhanças de comportamento dos valores de R^2 do DT e do RF permanecem as mesmas da previsão do Tp_{HL}. Na Fig. 9 é possível constatar esses valores de forma mais intuitiva com a curva de distribuição esperada e a curva predita por cada modelo para a porta inversora (assim como na Fig. 8, o comportamento da NAND2 é o mesmo).

4.3 Predição da Energia

Para a energia, o melhor algoritmo foi - assim como no caso do Tp_{LH} - o RF, seguido do DT, sendo o SVR o penúltimo e o MLR o último para as duas portas. O menor R^2 dessa predição foi dado

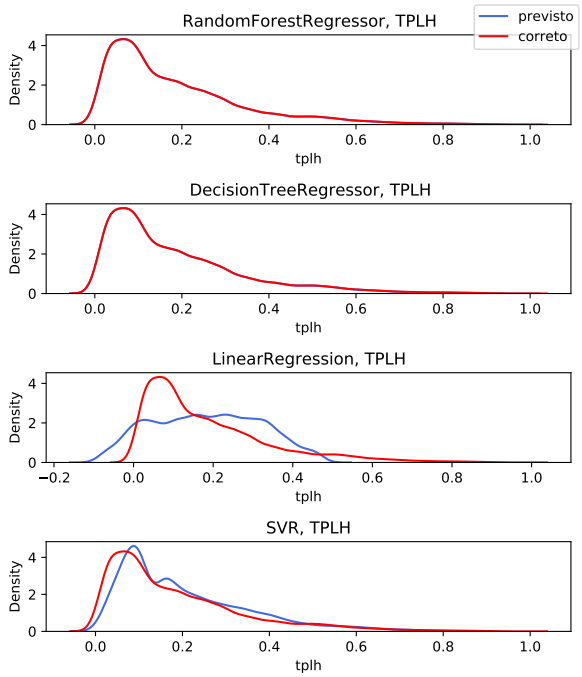

Figura 9: Predição do atraso de descida Tp_{LH} para a porta NOT

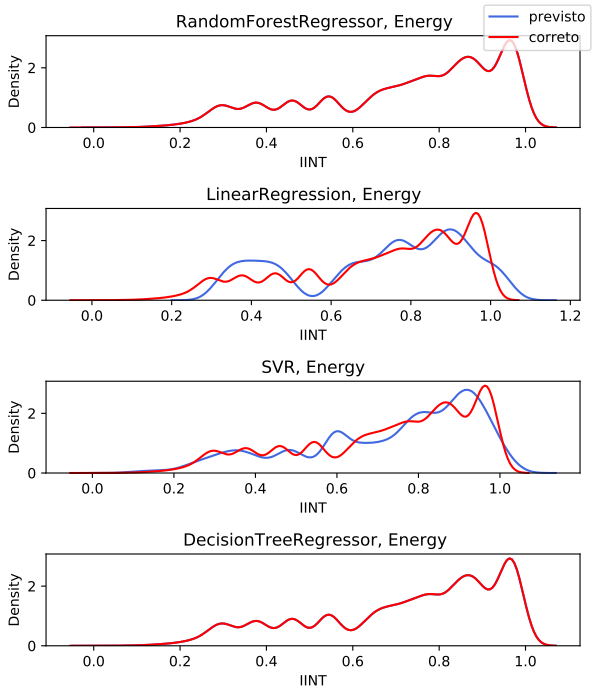

Figura 10: Predição do consumo de energia para a porta NAND2

pelo MLR da porta NOT, sendo igual a 47.32%. Na Fig. 10, pode-se observar o mesmo padrão de comportamento encontrado nas variáveis de atraso da porta NOT, dessa vez com os resultados da porta NAND2.

A pior performance do algoritmo MLR pode ser justificada pela forma como a correlação entre nossas variáveis é dada. Essa correlação pode ser observada na Fig. 11, na qual se apresentam os valores do coeficiente de correlação linear de Pearson para a porta NAND2 (o mesmo é válido para a NOT). Quanto mais perto de 1 ou -1 é o valor de correlação, mais é possível explicar as variáveis por uma relação linear (positiva ou negativa). Portanto, pode-se ver que não há muito em que o algoritmo de Regressão Linear Múltipla possa se basear para ter uma melhor performance.

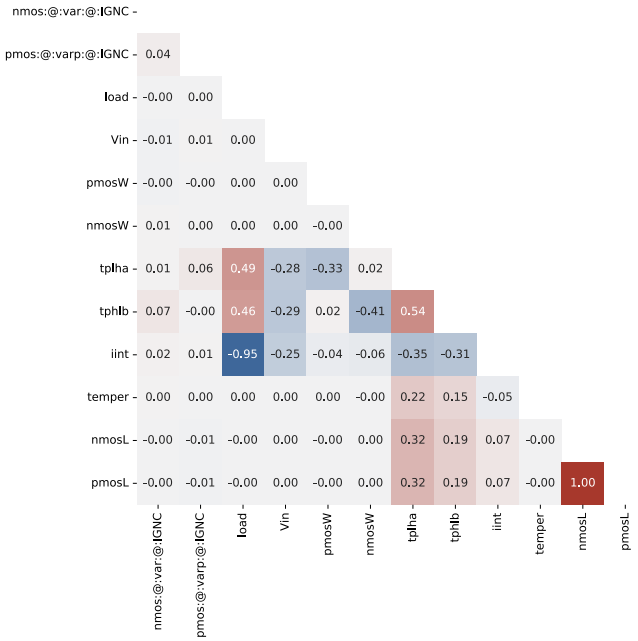

Figura 11: Valores de correlação linear de Pearson

4.4 Tempo de Inferência

Uma análise final foi realizada para avaliar o *trade-off* entre a precisão dos modelos e tempo de execução por inferência de cada um. Pelos dados gerados, os modelos DT superaram os de RF, embora RF apresentem valores um pouco superiores de R^2. Como exemplo, apresentado pela Tabela 5, uma única simulação elétrica para a porta NAND2 é executada por 17,5 s na ferramenta HSPICE, em comparação com 0,0006 s no modelo DT e 0,0398 s no modelo RF. É importante salientar a diferença grande entre o tempo de simulação da porta NAND2 e da porta NOT. Pode-se ver pela Fig. 12 que enquanto o tempo de simulação aumenta exponencialmente quando se aumenta a complexidade do circuito, o tempo de inferência dos modelos treinados não cresce significativamente. Dessa forma algoritmos de machine learning podem economizar consideravelmente o tempo de simulação desses circuitos.

Tabela 5: Tempo de Inferência versus Tempo de Simulação

Porta	Simulação	DT	RF	SVR	MLR
NOT	0.069*s*	0.0006*s*	0.0432*s*	0.0007*s*	0.0006*s*
NAND2	17.5*s*	0.0006*s*	0.0398*s*	0.0029*s*	0.0006*s*

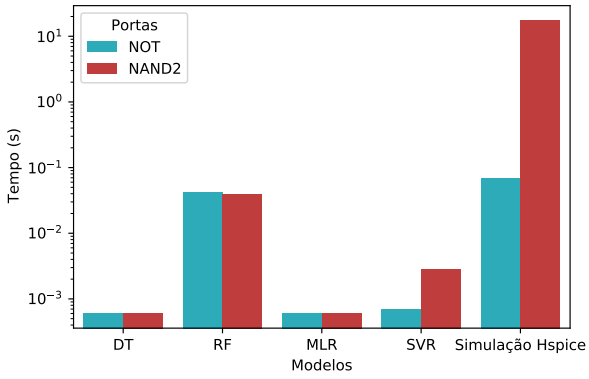

Figura 12: Tempo de Inferência e Tempo de Simulação para cada Porta

5 CONCLUSÕES

Neste trabalho, foram analisados quatro tipos de algoritmos de aprendizado de máquina para a tarefa de regressão de prever o comportamento elétrico de um circuito NAND2 CMOS e de um circuito NOT CMOS. Treinaram-se os algoritmos com os dados da simulação de Monte Carlo usando o software Hspice.

Os resultados mostram que, para todas as três variáveis de saída (Tp_{HL}, Tp_{LH} e energia), o algoritmo de melhor desempenho foi o Floresta Aleatória, seguido do Árvores de Decisão, ou seja, os melhores algoritmos foram aqueles que baseiam suas técnicas em estruturas de árvores. Contudo, para o objetivo desse trabalho, considerando que o tempo que os algoritmos levam para fazer suas predições deve ser o menor o possível comparado com a simulção tradicional, o algortimo que equilibra predição correta com menor tempo de inferência foi o Árvore de Decisão.

Os resultados inferiores de performance do algoritmo de Regressão Linear Múltipla podem ser justificados salientando-se que muitas das variáveis não possuem uma correlação linear. Quanto ao SVR, trata-se também de um algoritmo muito poderoso e que, talvez com o uso de outros *kernels* como o polinomial, é possível chegar a um erro menor. Contudo, mesmo que se chegue a essa melhor performance no SVR, esse possui um problema que não é possível evitar: o elevado tempo de treinamento, tanto com e sem a técnica de *cross-validation*. Como o conjunto de dados de simulação é grande e a tendência para trabalhos futuros é só aumentar, o SVR não escala tão bem para fazer um treinamento rápido, o que foge do objetivo do trabalho que é reduzir o tempo de simulação de circuitos por meio de algoritmos de *machine learning*.

Em trabalhos futuros pretende-se explorar outros modelos mais complexos e mais atuais da literatura como arquiteturas de redes neurais. Ademais, pretende-se simular outros circuitos mais complexos, como portas XOR e *Full-adder*.

Portanto, o aprendizado de máquina é um ferramenta interessante e poderosa para prever o comportamento de circuitos elétricos, mesmo em sua técnica mais básica, e reduzir o longo processo de simulação tradicional. Esse fluxo de treinamento pode ser usado futuramente por fabricantes que desejam disponibilizar seus dispositivos sem a necessidade de abrir mão dos direitos da sua propriedade intelectual. Ademais, pode-se a partir dos modelos treinados criar ferramentas educativas de fácil uso para alunos de fluxo de síntese de circuitos (como é o caso do protótipo construído e disponibilizado no endereço eletrônico https://electrical-characterization.herokuapp.com/).

AGRADECIMENTOS

Este trabalho foi parcialmente financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq e pela Propesq/UFSC.

REFERENCES

[1] Andrew B. Kahng. Reducing time and effort in ic implementation: A roadmap of challenges and solutions. In *Proceedings of the 55th Annual Design Automation Conference*, DAC 18, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450357005. doi: 10.1145/3195970.3199854. URL https://doi.org/10.1145/3195970.3199854.

[2] A. Chatterjee, D. Croley, V. Ramamurti, and Kui-Yu Chang. Application of machine learning to manufacturing: results from metal etch. In *Nineteenth IEEE/CPMT International Electronics Manufacturing Technology Symposium*, pages 372–377, 1996. doi: 10.1109/IEMT.1996.559762.

[3] J. K. Lee, K. Ko, and H. Shin. Analysis on process variation effect of 3d nand flash memory cell through machine learning model. In *2020 4th IEEE Electron Devices Technology Manufacturing Conference (EDTM)*, pages 1–4, 2020. doi: 10.1109/EDTM47692.2020.9117940.

[4] A. B. Kahng, U. Mallappa, and L. Saul. Using machine learning to predict path-based slack from graph-based timing analysis. In *2018 IEEE 36th International Conference on Computer Design (ICCD)*, pages 603–612, 2018. doi: 10.1109/ICCD.2018.00096.

[5] E. C. Barboza, N. Shukla, Y. Chen, and J. Hu. Machine learning-based pre-routing timing prediction with reduced pessimism. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2019.

[6] A. Kaintura, K. Foss, I. Couckuyt, T. Dhaene, O. Zografos, A. Vaysset, and B. Sorée. Machine learning for fast characterization of magnetic logic devices. In *2018 IEEE Electrical Design of Advanced Packaging and Systems Symposium (EDAPS)*, pages 1–3, 2018. doi: 10.1109/EDAPS.2018.8680898.

[7] Gabriel Lima Jacinto, Lucas Yuki Imamura, Mateus Grellert, and Cristina Meinhardt. Exploring machine learning for electrical behavior prediction: The cmos inverter case study. In *2022 35th SBC/SBMicro/IEEE/ACM Symposium on Integrated Circuits and Systems Design (SBCCI)*, pages 1–6, 2022. doi: 10.1109/SBCCI55532.2022.9893261.

[8] Aurélien GÊron. Hands-on machine learning with Scikit-Learn and TensorFlow concepts, tools, and techniques to build intelligentsystems. Paperback, April 2017. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/1491962291.

[9] Wei Zhao and Yu Cao. New generation of predictive technology model for sub-45nm design exploration. In *7th International Symposium on Quality Electronic Design (ISQED'06)*, pages 6 pp.–590, 2006. doi: 10.1109/ISQED.2006.91.

[10] Massimo Alioto, Elio Consoli, and Gaetano Palumbo. Variations in nanometer cmos flip-flops: Part i—impact of process variations on timing. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 62(8):2035–2043, 2015.

[11] Frank Kane. *Hands-on data science and python machine learning*. Packt Publishing, Birmingham, UK, 1 edition, July 2017.