

UNIVERSIDADE FEDERAL FLUMINENSE

JOSEANE SANTOS ALMEIDA

**PREDIZENDO SEMICONDUTORES DE GAP
ULTRA LARGO COM MACHINE LEARNING**

NITERÓI

2021

JOSEANE SANTOS ALMEIDA

**PREDIZENDO SEMICONDUTORES DE GAP
ULTRA LARGO COM MACHINE LEARNING**

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Física da Universidade Federal Fluminense como requisito parcial para obtenção do Grau de Mestre em Física. Área de concentração: Física da matéria condensada

Orientador:
PEDRO VENEZUELA

Coorientador:
MÁRCIO COSTA

NITERÓI

2021

Ficha catalográfica automática - SDC/BIF
Gerada com informações fornecidas pelo autor

A447p Almeida, Joseane Santos
Predizendo semicondutores de gap ultra largo com machine learning / Joseane Santos Almeida ; Pedro Paulo de Mello Venezuela, orientador ; Marcio Jorge Teles da Costa, coorientador. Niterói, 2022.
146 p. : il.

Dissertação (mestrado)-Universidade Federal Fluminense, Niterói, 2022.

DOI: <http://dx.doi.org/10.22409/PPGF.2022.m.85803584509>

1. Aprendizado de máquina. 2. Teoria de faixa de energia de sólidos. 3. Física da matéria condensada. 4. Estrutura eletrônica. 5. Produção intelectual. I. Venezuela, Pedro Paulo de Mello, orientador. II. Teles da Costa, Marcio Jorge, coorientador. III. Universidade Federal Fluminense. Instituto de Física. IV. Título.

CDD -

JOSEANE SANTOS ALMEIDA

PREDIZENDO SEMICONDUTORES DE GAP ULTRA LARGO COM MACHINE
LEARNING

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Física da Universidade Federal Fluminense como requisito parcial para obtenção do Grau de Mestre em Física. Área de concentração: Física da matéria condensada

Aprovada em julho de 2022.

BANCA EXAMINADORA

Prof. Pedro Paulo de Mello Venezuela - IF/UFF

Prof. Felipe David Crasto de Lima - CNPEM

Prof. Gustavo Martini Dalpian - UFABC

Prof. Rodrigo Garcia Amorim - ICEx/UFF

Niterói

2021

Dedico àqueles da minha família e amigos que apoiam e incentivam meus sonhos.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

À Universidade Federal Fluminense, pela infraestrutura e principalmente, o restaurante universitário.

Ao meu orientador, Pedro Venezuela, e coorientador, Márcio Costa, pela orientação serena e com comprometimento.

Epígrafe

"Ao infinito e além!"

Buzz Lightyear

Resumo

As investigações incipientes sobre os Semicondutores de Gap Ultra Largo (UWBG) tem motivado a construção de dispositivos para aplicações inacessíveis através dos semicondutores de gap inferior, e oferecem desafios de síntese e dopagem que estimulam novos estudos teóricos e experimentais. Esses dispositivos podem operar em alta potência, alta frequência e temperatura, e viabilizam a optoeletrônica ultravioleta. O gap desses materiais não possuem um limite mínimo bem definido na literatura, sendo relativamente superior ao gap de 3,4 eV do GaN. Enquanto o maior desenvolvimento nesse campo se deve aos materiais *bulk* AlGaN, Ga_2O_3 e o diamante, a pesquisa de semicondutores bidimensionais de gap ultra largo é ainda mais escassa e estimulada pelo já conhecido nitreto de boro hexagonal (h-BN). Este trabalho se propõe a descobrir novos materiais bidimensionais de gap ultra largo utilizando o Aprendizado de Máquina (*Machine Learning*). Primeiramente é realizado o processo de aprendizagem de um modelo de classificação com o intuito de categorizar os materiais em metais e isolantes. Posteriormente, um modelo de regressão aprende a prever o valor do gap dos isolantes. Após a otimização dos hiperparâmetros do modelo de classificação, a área sob a curva precisão-recall no conjunto de teste é de 0,8. Enquanto isso, o modelo de regressão apresenta erro quadrático médio (RMSE) de 0,25 eV. Por intermédio desses modelos foi possível classificar 800 novos materiais de estequiometria ABC_2 e ABC_4 , dos quais 266 foram previstos como isolantes. Entre os 266 materiais, 134 foram considerados UWBG. Ademais, foram sugeridos 479 sistemas de estequiometria AB_2 , em que 37 são considerados UWBG. Entre eles, o CdF_2 no protótipo CdI_2 tem o gap confirmado por cálculos de estrutura eletrônica utilizando a Teoria do Funcional da Densidade (DFT) implementada no código Quantum Espresso.

Palavras-chave: machine learning; aprendizado de máquina; band gap; gap; C2db; materiais bidimensionais.

Abstract

Incipt investigations on Ultrawide-Bandgap Semiconductors (UWBG) have motivated the construction of devices for applications inaccessible through narrower-bandgap semiconductors, and offer synthesis and doping challenges that stimulate new theoretical and experimental studies. These devices can operate at high power, frequency, temperature, and make UV optoelectronics possible. The bandgap of these materials does not have a well-defined minimum limit in the literature, being relatively higher than the GaN bandgap of 3.4 eV. While the greatest development in this field is due to the *bulk* materials AlGaN, Ga_2O_3 and diamond, the research of two-dimensional ultrawide-bandgap semiconductors is even scarcer and stimulated by the already known hexagonal boron nitride (h -BN). This work proposes to discover new two-dimensional ultrawide-bandgap materials using Machine Learning. First, the learning process of a classification model is performed in order to categorize materials into metals and insulators. Subsequently, a regression model learns to predict the bandgap value of insulators. After optimizing the hyperparameters of the classification model, the area under the precision-recall curve in the test set is 0.8. Meanwhile, the regression model has a mean square error (RMSE) of 0.25 eV. Through these models, it was possible to classify 800 new materials, and 266 were predicted to be insulators. Among the 266 materials, 134 were considered UWBG. Furthermore, 479 AB_2 stoichiometry systems were suggested and 37 of them was considered UWBG. Among them, the CdF_2 in the CdI_2 prototype has band gap confirmed by electronic structure calculations using the Density Functional Theory (DFT) implemented in the Quantum Espresso code.

Keywords: machine learning; band gap; gap; C2db; two-dimensional materials.

List of Figures

1	Ciclo autoconsistente (ALVES, 2011)	31
2	Estrutura de Bandas do TlN na estrutura zinc-blend. O nível de Fermi E_F foi deslocado para a origem.	38
3	Representação da última banda de um cristal com um número ímpar de elétrons. Figura extraída da referência (CAPAZ, 2019).	39
4	Representação da última banda de valência e a primeira banda de condução sem superposição de um cristal com um número ímpar de elétrons. Figura extraída da referência (CAPAZ, 2019).	40
5	Representação da última banda de valência e a primeira banda de condução com superposição de um cristal com um número ímpar de elétrons. Figura extraída da referência (CAPAZ, 2019).	41
6	À esquerda, distribuição dos materiais no banco de dados C2DB. À direita, diagrama de barras das estequiometrias disponíveis.	56
7	Conjunto de dados com estequiometria AB_2 e <i>features</i> atômicas.	58
8	Conjunto de dados com todas as estequiometrias e <i>features</i> estatísticas.	59
9	Curva precisão-recall para AB_2 com <i>features</i> atômicas.	60
10	Curva precisão-recall para todas as estequiometrias com <i>features</i> estatísticas.	60
11	Histograma dos band gaps dos isolantes no banco de dados C2DB.	61
12	Regressão do gap.	62
13	Estruturas associadas a cada protótipo.	63
13	Estruturas associadas a cada protótipo.	64
13	Estruturas associadas a cada protótipo.	65
14	Diagrama de barras do gap PBE previsto nos materiais em cada protótipo.	66

14	Diagrama de barras do gap PBE previsto nos materiais em cada protótipo.	67
15	Band gap predito <i>versus</i> massa atômica ponderada.	68
16	Diagrama de barras do gap previsto nos materiais AB ₂ em cada protótipo.	69
17	Estrutura do CdF ₂ com protótipo CdI ₂	70
18	Otimização estrutural do CdF ₂ na estrutura do protótipo CdI ₂	70
19	Estrutura de bandas do CdF ₂ com protótipo CdI ₂	71

Lista de Tabelas

1	Figuras de mérito de alguns semicondutores de gap estreito, largo e ultra largo, normalizada pelo Si.	15
2	Propriedades de alguns semicondutores de gap largo e ultra largo.	15
3	Features atômicas.	57
4	Features estatísticas.	58
5	RMSE do gap no processo de treinamento.	62
6	Gap decorrente do modelo <i>Gradient Boosting</i> e DFT para o CdF ₂ –CdI ₂ . .	71

Sumário

1	Introdução	12
1.1	Semicondutores de Gap Ultra Largo	14
2	Fundamentação Teórica	19
2.1	Problema de Muitos Corpos	19
2.1.1	Aproximação de Born-Oppenheimer	20
2.2	Teoria Funcional da Densidade	23
2.2.1	Teoremas de Hohenberg-Kohn	25
2.2.2	Equações de Kohn-Sham	27
2.2.3	Funcionais de Troca e Correlação	31
2.3	Elétrons em um potencial periódico	33
2.4	Base de Ondas Planas e Estrutura de Bandas	36
2.5	Funções de base	41
2.5.1	Teoria dos Pseudopotenciais	41
2.5.1.1	Pseudopotenciais de norma conservada	43
2.5.1.2	Pseudopotenciais Ultrasuaves	44
2.5.2	Projetor de ondas aumentadas	44
2.6	Quantum Espresso - QE	46
3	Aprendizado de Máquina	49
3.1	Árvore de Decisão	50
3.2	Floresta Aleatória	51

3.3 AdaBoost	52
3.4 Gradient Boosting	53
4 Resultados e Discussões	55
4.1 Construção do <i>dataset</i>	55
4.2 Classificação	59
4.3 Regressão	61
4.4 Predição	62
4.5 Cálculos da Estrutura Eletrônica	69
5 Conclusões e Perspectivas	72
REFERÊNCIAS	74
Apêndice A - Algoritmos	82
A.1 <i>Dataset</i> AB ₂	82
A.2 <i>Dataset</i> geral	84
A.3 <i>Dataset</i> com materiais novos de estequiometrias ABC ₂ e ABC ₄	88
A.4 <i>Dataset</i> de materiais novos com estequiometria AB ₂ - <i>features</i> estatísticas .	93
A.5 <i>Dataset</i> de materiais novos com estequiometria AB ₂ - <i>features</i> atômicas .	98
A.6 Classificação geral	99
A.7 Classificação AB ₂	113
A.8 Regressão geral	125
A.9 Regressão AB ₂	133

1 Introdução

A indústria de dispositivos semicondutores modernos tem menos de um século e vem revolucionando nossas vidas diárias. O uso de tecnologias baseadas em silício (Si) e gerânio (Ge) foram incentivadas pela criação do primeiro transistor em 1947, inventado por Bardeen e Brattain no *Bell Telephone Laboratories* ([BARDEEN; BRATTAIN, 1950](#)), e promoveram aplicações que começaram com rádios e walkie-talkies e se expandiram para rádios de polícia e satélites de comunicação militar. O silício é relativamente barato e fácil de se obter em alto nível de pureza, sendo utilizado amplamente na construção de diodos, circuitos integrados e retificadores. A partir de 1950, iniciou-se a segunda geração de semicondutores com os compostos III-Vs, principalmente III-P e III-As, em que medidas mostraram não apenas uma maior mobilidade eletrônica que o Si e Ge, bem como um gap maior e direto ([TSAO et al., 2018](#)).

Posteriormente, a terceira geração de semicondutores é marcada por seus gaps largos (*wide band gap* - WBG), maiores que $\approx 2,3\text{ eV}$. Na optoeletrônica dessa geração, com o financiamento governamental e industrial, foi construído o diodo emissor de luz (LED) baseado em InGaN. Enquanto isso, para dispositivos eletrônicos o material não necessita de gap direto ou eficiência na emissão de luz, e assim o carbeto de silício (SiC) e o nitreto de gálio (GaN) se tornam viáveis. Alguns dos desenvolvimentos iniciais de semicondutores de gap largo ocorreram em aplicações militares na Agência de Projetos de Pesquisa Avançada de Defesa (DARPA) em um projeto de centenas de milhões de dólares ([JIN et al., 2016](#)). Esses materiais têm como propriedades vantajosas conseguir operar em ambientes hostis, pois funcionam com maiores frequências, maiores voltagens e temperaturas mais altas. O projeto consistia em três etapas: de 2002 a 2004 focado no desenvolvimento de materiais e atingir a comercialização de substrato de SiC de 2 a 4 polegadas; de 2005 a 2007 produzir o amplificador de potência de radiofrequência (RF); de 2008 a 2009 juntamente com a empresa Raytheon construir o circuito integrado monolítico de micro-ondas (MMIC) baseado em GaN, diminuindo o custo em 75 %. Em 2014, a Raytheon utiliza o nitreto de gálio no radar antimíssil "Patriot" que permitiria detecção de 360 graus. Paralelamente,

na Europa foi desenvolvida a tecnologia GaN High-electron-mobility transistor (HEMT), além de produzirem amplificadores de alta potência e radares baseados no GaN.

Outra aplicação importante para os semicondutores de gap largo é a construção de LEDs com comprimento de onda menores. Com o estudo recente dos semicondutores de gap ultra largo (*Ultrawide-bandgap semiconductors* - UWBG) foi possível construir dispositivos emissores no ultravioleta C (UVC), que podem desativar vírus e bactérias. Tais emissores são importantes para a indústria alimentícia (substituindo lâmpadas de mercúrio, por exemplo) (MENGLÉ, 2020) e podem ser utilizados contra o vírus pandêmico Sars-Cov-2 (BUONANNO; WELCH; BRENNER, 2020). Entre os UWBG, os materiais mais representativos são *AlGaN*, *Ga₂O₃* e o diamante (HIGASHIWAKI; KAPLAR; PERNOT, 2021).

Percebe-se que as gerações de semicondutores foram marcadas por materiais tridimensionais. Em 2004, a descoberta do grafeno estimulou o estudo de materiais bidimensionais (2D) proporcionando suas sínteses, estudo de propriedades físicas e novas aplicações (MC-CREARY et al., 2020). Materiais bidimensionais são aqueles compostos por uma ou poucas camadas de átomos, no qual as ligações são mais fortes no plano do que na direção de empilhamento. O grafeno é composto por átomos de carbono em rede hexagonal, ligados por hibridização *sp*², e possuem alta mobilidade de portadores e condutividade térmica. Além disso, devido a força menor entre as camadas para os sistemas bidimensionais de van der Waals, é possível empilhar diferentes materiais 2D formando heteroestruturas. Entre as aplicações mais almejadas em estruturas 2D estão a eletrônica e a optoeletrônica flexível em nanoscalo. Os esforços na síntese e estudo de propriedades possibilitaram a descoberta de muitos materiais 2D, tais como, h-BN, siliceno, germaneno, estaneno, fosforeno, dicalcogenetos de metais de transição (TMDs), MXenes, borofeno, estruturas Janus de TMDs, e materiais 2D não advindos de interação de van der Waals, por exemplo o hemateno. Embora parte desses materiais sejam baseados na estrutura do grafeno, eles possuem diferentes propriedades e abrangem diferentes estruturas de bandas. Entre eles, são semicondutores o h-BN, fosforeno, TMDs, Janus TMDs e o hemateno. O estaneno é um isolante topológico e os demais são metais. Os materiais 2D semicondutores possuem um largo espectro de aplicações, seja na optoeletrônica (LEDs, laser, fotodetectores) ou em sensores (gás, químico, magnético, biológico) (KHAN et al., 2020) (SCHLEDER; ACOSTA; FAZZIO, 2020).

1.1 Semicondutores de Gap Ultra Largo

Paralelamente aos estudos de semicondutores de gap largo, semicondutores de gap ultra largo (*ultra wide band gap semiconductors* - UWBG) começaram a ser investigados recentemente. Embora não exista uma definição única para o gap limite que separe essas duas classes de materiais, geralmente os UWBG possuem gaps relativamente superiores ao gap do *GaN* (3,4 eV). Em consequência de muitas figuras de mérito, responsáveis por medirem a performance de dispositivos, crescerem não linearmente com o gap, os UWBG tem performance superior aos UWG para várias aplicações (TSAO et al., 2018).

Johnson (JOHNSON, 1965), em 1965, propôs a figura de mérito JFOM, usada comumente para determinar a capacidade de frequência de um transistor e é dada por (BALLESTÍN-FUERTES et al., 2021),

$$JFOM = \frac{E_c v_s}{2\pi}, \quad (1.1)$$

escrita em termos do campo elétrico crítico (E_c), que varia aproximadamente com o quadrado do gap, e a velocidade de saturação de deriva dos portadores (v_s). Keyes (KEYES, 1972) derivou a figura de mérito KFOM, que fornece uma limitação térmica dos transistores. Sua expressão é dada em termos da condutividade térmica λ ,

$$KFOM = \lambda \left[\frac{cv_s}{4\pi\epsilon} \right]^{1/2}, \quad (1.2)$$

Baliga (BALIGA, 1982), em 1983, desenvolveu a figura de mérito que define as perdas de condução em dispositivos unipolares de baixa frequência,

$$BFOM = \epsilon\mu E_G^3, \quad (1.3)$$

em que μ é a mobilidade do portador e E_G é o gap do semicondutor.

Os atuais estudos de síntese e propriedades físicas dos UWBG, principalmente Al-GaN/AlN, Ga₂O₃ e diamante, proporcionam a possibilidade de utilizar esses e novos materiais para a construção de dispositivos. A tabela 1 mostra algumas figuras de mérito para materiais de diferentes gaps, normalizadas pelo valor referente ao silício. Enquanto isso, propriedades importantes para aplicações de gap largo são ilustradas na tabela 2, incluindo métricas importantes para análise do seu uso em dispositivos: a qualidade dos substratos e a habilidade em ser dopado do tipo n e do tipo p. Embora algumas aplicações não necessitem de bons valores para essas três métricas, a maioria deles requer tais

condições, como por exemplo, o diodo laser.

Tabela 1: Figuras de mérito de alguns semicondutores de gap estreito, largo e ultra largo, normalizada pelo Si.

	Si	GaAs	4H-SiC	2H-GaN	$\beta - Ga_2O_3$	2H-AlN	Diamante
JFOM	1	2,6	10,1	6,5	23,2	29,6	22,3
KFOM	1	0,5	3,3	0,8	0,2	1,9	21,0
BFOM	1	13	13,8	18,9	9,7	25,0	82,0

Fonte: ([BALLESTÍN-FUERTES et al., 2021](#))

Tabela 2: Propriedades de alguns semicondutores de gap largo e ultra largo.

Material	GaN	4H-SiC	AlGaN/AlN	$\beta - Ga_2O_3$	Diamante
Band gap (eV)	4,4	3,3	Até 6	4,9	5,5
Condutividade térmica ($Wm^{-1}K^{-1}$)	253	370	253-319	11-27	2290-3450
Qualidade do substrato (deslocamentos por cm^2)	$\approx 10^4$	$\approx 10^2$	$\approx 10^4$	$\approx 10^4$	$\approx 10^5$
Diâmetro do substrato (polegadas)	8 (no Si)	8	2	4	1
Dopagem tipo p demonstrada	Boa	Boa	Fraca	Não	Boa
Dopagem tipo n demonstrada	Boa	Boa	Moderada	Boa	Moderada

Fonte: ([TSAO et al., 2018](#))

Para aplicações eletrônicas, as ligas de *AlGaN* oferecem adequadas propriedades físicas fundamentais. Dentre elas, um gap direto que pode variar de 3,4 eV até 6,0 eV; alto campo elétrico de ruptura ($> 10\text{ MV cm}^{-1}$ para AlN); alta mobilidade eletrônica (mobilidades no bulk até $1000\text{ cm}^2\text{ V}^{-1}\text{ s}^{-1}$); altas velocidades de saturação ($> 10^7\text{ cm s}^{-1}$) e uma certa facilidade em ser dopado tipo n com silício. Para aplicações optoeletrônicas, pode-se emitir no UVA, UVB e UVC. Entretanto, há a ausência de substratos *single-crystal* com qualidade suficiente para o crescimento epitaxial e controle insuficiente da heteroepitaxia nesses substratos. Outra adversidade, comum a todos os semicondutores de gap ultra largo, é a dificuldade de controlar a dopagem.

O diamante possui campo elétrico de ruptura maior que 10 MV cm^{-1} , alta mobilidade de elétrons e buracos ($> 2000\text{ cm}^2\text{ V}^{-1}\text{ s}^{-1}$) e velocidades de deriva saturadas de $2,3 \times 10^7\text{ cm s}^{-1}$ para elétrons e $1,4 \times 10^7\text{ cm s}^{-1}$ para buracos. Além disso, sua alta condutividade térmica possibilita a construção de dispositivos operando em maiores temperaturas. Ademais, defeitos centrais como vacância de nitrogênio (N-V) ou de silício (Si-V) tem sido propostos para o uso em sistemas de informação quântica. Um dos avanços que incentivaram os estudos do diamante foram o desenvolvimento de substratos com relativa baixa densidade de defeitos ($< 10^5\text{ cm}^{-2}$) e novas formas de síntese para substratos maiores. O segundo avanço está relacionando a dopagem do tipo p com átomos de boro e do tipo n com fósforo. Ao mesmo tempo em que esses níveis são profundos (0,3 eV para B e 0,57 para P), as concentrações de impurezas podem ser maiores que 10^{20} cm^{-3} ,

promovendo baixa resistividade em temperatura ambiente pela condução por *hopping* e consequentemente tornando possível construir contatos elétricos de baixa resistividade. O hopping é um mecanismo de condução que se destaca em baixas temperaturas, em que o transporte de portadores na banda de valência e condução não são significativos. Nesse regime, tais portadores estão presos em poços de potencial criados pelas impurezas e possuem uma probabilidade de "saltar" para níveis de energia distintos em impurezas distintas, e posteriormente são aprisionados novamente. Esse salto eletrônico ocorre em razão do deslocamento dos poços pelas vibrações atômicas ([TUMELERO, 2010](#)).

No que diz respeito ao $\beta-Ga_2O_3$ para aplicações eletrônicas, o material dispõe de um gap maior que 4,5 eV, alto campo elétrico de ruptura de aproximadamente 9 MV cm^{-1} e um bom controle na dopagem do tipo n usando Si ou Sn. Uma das maiores vantagens desse material é a construção de substratos grandes, de alta qualidade e com preços acessíveis. No entanto, a incapacidade de dopagem tipo p impede a construção de dispositivos bipolares, tal qual os LEDs. Já que os estados de valência são predominantemente advindos do orbital 2p do oxigênio, estes possuem natureza localizada, levando a formação de pólarons e aprisionamento dos buracos, atrapalhando a dopagem do tipo p ([VARLEY et al., 2012](#)). Outra expressiva desvantagem é a baixa condutividade térmica.

O processo de dopagem enfrenta também outros desafios que são comuns aos UWBG. Um deles se refere ao aumento da energia de ionização da impureza com o gap. A energia de ionização, E_I , é a energia que separa o dopante em seu estado fundamental do extremo da banda eletrônica relacionada, ou seja, do mínimo da banda de condução para uma impureza doadora ou do máximo da banda de valência para a impureza aceitadora. Ao utilizar o modelo do átomo de hidrogênio para estimar a energia de ionização, tal energia depende da massa efetiva de elétrons e buracos, e consequentemente do gap. No entanto, como a massa efetiva de buracos é maior que a dos elétrons, a energia de ionização dos buracos cresce de forma mais significativa com o gap, provocando uma maior dificuldade para dopagens do tipo p. Como os UWBG possuem maior energia de ionização, diminui-se a quantidade de portadores oriundos da excitação térmica, freando a construção de dispositivos em temperatura ambiente. ([RAVICHANDRAN; WANG; WAGER, 2016](#)).

Além do aumento da energia de ionização, os UWBG tem uma tendência maior em criar impurezas compensadoras durante a dopagem. A energia de formação dessas impurezas dependem linearmente do nível de Fermi, E_F . Quanto maior o gap, a concentração de impurezas dopantes altera significativamente o nível de Fermi. Ao realizar uma dopagem do tipo p, por exemplo, o nível de Fermi se desloca em direção a banda de valência,

diminuindo a energia de formação de impurezas doadoras. Sendo assim, tais defeitos que atuam como doadores são mais presentes em materiais do tipo n (YAN; WEI, 2008) (WALLE; NEUGEBAUER, 2004).

Gradualmente, investigações de diferentes *bulks* UWBG tem sido feitas. Por exemplo, MgGa₂O₄ (GALAZKA et al., 2004), GeO₄ (CHAE et al., 2021) e com destaque, o nitreto de Boro (BN). Esse é isoeletrônico e isoestrutural com os alótropos do carbono. Assim como o grafite com ligações sp^2 , o material em sua fase hexagonal (h-BN) é menos rígido que em sua estrutura cúbica (c-BN), similar ao diamante com ligações sp^3 . Além dessas estruturas estáveis, o nitreto de boro pode ser encontrado na fase wurtzita metaestável, amorfa (α -BN), turbostrática (t-BN) e romboédrica (r-BN). Na estrutura wurtzita, a formação de ligas com compostos III-V é estimulada pela produção de emissores UV, obtendo gaps maiores que 8 eV (KUDRAWIEC; HOMMEL, 2020). Enquanto isso, na estrutura cúbica, o campo elétrico de ruptura é maior que 15 MV cm^{-1} e o gap pode chegar a 6,4 eV, sendo possível realizar a dopagem do tipo n com silício e do tipo p com berílio. Ademais, teoricamente, tem a terceira maior condutividade térmica ($\approx 2145\text{ W m}^{-1}\text{ K}^{-1}$) quando ele é isotopicamente puro. Entretanto, a estrutura hexagonal é mais estável, fomentando o desafio de realizar processos de crescimento em que não ocorra transição estrutural (TSAO et al., 2018) (L. LINDSAY; REINECKE, 2013).

A facilidade em sintetizar *single layers* na estrutura hexagonal do BN tem despertado o interesse para aplicações eletrônicas bidimensionais. Em 2018, Wickramaratne et. al investigaram as propriedades eletrônicas do h-BN como função do número de camadas. No bulk, o gap é indireto ($\Gamma \rightarrow K$) de 5,95 eV e torna-se direto (K) com o valor de 6,08 eV para a monocamada (DARSHANA WICKRAMARATNE; WALLE, 2018). Ao realizar uma diferente rota de síntese, com solvente Ba-B-N em altas pressões e temperatura, Shevitski et al identificaram um novo *color center* que emite luz na frequência do azul. O modelo apresentado discute que a forma de síntese promove impurezas de bário intersticiais e vacâncias. Ao realizar irradiação por feixe de elétrons, tais impurezas ocupam as vacâncias e emitem luz com pico em 435 nm (SHEVITSKI et al., 2019).

O estudo de materiais de gap ultra largo bidimensionais além do h-BN ainda é escasso. Sabendo que o gap aumenta com a diferença de eletronegatividade dos átomos constituintes, e que halogênios possuem maior eletronegatividade que qualquer outro elemento na mesma linha da tabela periódica, propor combinações de metais com elementos halogênicos pode produzir materiais de gap ultra largo. Dessa forma, surgiram estudos teóricos recentes sobre o CaFCl (YE et al., 2020) e BaFCl (ZHU, 2020). O CaFCl é considerado

um UWBG com gap direto de 6,62 eV e adequado para dispositivos tipo n. Enquanto isso, o BaFCl dispõe de um gap indireto igual a 5,58 eV que pode variar para um gap direito a partir de uma pequena tensão, além de revelar alta absorção óptica no UV. Em 2020, Baskurt *et al* analisaram a estabilidade de monocamadas CaX_2 ($X = F, Cl, Br, I$), e mostraram que eles são materiais bidimensionais de gap ultra largo indireto ([BASKURT et al., 2020](#)). De outro modo, embasado nas investigações anteriores do $h - BN$, Yanfeng Ge *et al* encontraram para a monocamada de óxido de berílio hexagonal (h-BeO) um gap indireto de 7,05 eV e alta mobilidade eletrônica de $473 \text{ cm}^2 \text{ V}^{-1} \text{ s}^{-1}$ em temperatura ambiente ([GE et al., 2020](#)).

Atualmente, a descoberta de novos materiais UWBG bidimensionais segue a abordagem tradicional, apoiada em tentativa e erro ou em casos análogos. No entanto, isso gera um grupo restrito de materiais e que podem demandar simulações com grande custo computacional. Com o crescimento das bases de dados, o uso da inteligência artificial na tentativa de prever propriedades de materiais tem fornecido uma diferente forma de conceber materiais novos. Nesse trabalho, busca-se utilizar o aprendizado de máquina (*machine learning* - ML) para prever novos UWBG bidimensionais e, assim, direcionar a comunidade científica em novas pesquisas nesse campo.

2 Fundamentação Teórica

2.1 Problema de Muitos Corpos

De acordo com a teoria quântica a solução para o problema de muitos corpos consiste em encontrar os autovalores e autovetores do hamiltoniano. A equação de Schrödinger independente do tempo em unidades atômicas é ¹:

$$\hat{H}\psi(\mathbf{r}, \mathbf{R}) = \left\{ -\sum_i^N \frac{1}{2} \nabla_i^2 - \sum_A^M \frac{1}{2M_A} \nabla_A^2 + \sum_A^M \sum_{B>A}^M \frac{Z_A Z_B}{|\mathbf{R}_A - \mathbf{R}_B|} - \sum_\mu^N \sum_A^M \frac{Z_A}{|\mathbf{r}_\mu - \mathbf{R}_A|} + \sum_\mu^N \sum_{\nu>\mu}^N \frac{1}{|\mathbf{r}_\mu - \mathbf{r}_\nu|} \right\} \psi(\mathbf{r}, \mathbf{R}) = E\psi(\mathbf{r}, \mathbf{R}). \quad (2.1)$$

Os dois primeiros termos do hamiltoniano referem-se a energia cinética dos elétrons \hat{T}_e e dos núcleos \hat{T}_N , respectivamente. Os demais termos, nessa ordem, referem-se as interações elétricas clássicas núcleo-núcleo (\hat{V}_n), elétron-núcleo (\hat{V}_{en}) e elétron-elétron (\hat{V}_{ee}). Essa equação de autovalores é uma equação diferencial de $3N + 3M$ coordenadas acopladas, onde N é o número de elétrons e M é o número de núcleos. Deve-se notar que o presente hamiltoniano descreve um sistema nuclear-eletrônico, em que $\mathbf{r} = (\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_N)$ é o operador vetorial posição dos elétrons e $\mathbf{R} = (\mathbf{R}_1, \mathbf{R}_2, \dots, \mathbf{R}_M)$ é o operador vetorial posição dos núcleos. Entretanto, a maioria das propriedades físicas devem-se ao *comportamento dos elétrons*. A primeira aproximação que nos permite separar o movimento nuclear do elétronico é a aproximação de Born-Oppenheimer. Essa é um caso limite da aproximação adiabática (VIANNA; CANUTO S.; FAZZIO, 2004).

¹Em unidades atômicas $\hbar = e = m = 1$

2.1.1 Aproximação de Born-Oppenheimer

O termo de interação \hat{V}_{en} depende tanto das coordenadas eletrônicas quanto das coordenadas nucleares. Se não fosse sua relevante contribuição para a solução, poderíamos concluir que o hamiltoniano total é a soma $\hat{H}(\mathbf{r}, \mathbf{R}) = \hat{H}_1(\mathbf{r}) + \hat{H}_2(\mathbf{R})$ e consequentemente as funções de onda são o produto $\phi(\mathbf{R})\chi(\mathbf{r})$ e a energia total é $E = E_1 + E_2$. Como os núcleos são mais massivos que os elétrons, esses mudam seu estado rapidamente com as posições nucleares, o que inversamente não ocorre. Desde que a escala temporal dos elétrons é diferente da escala temporal dos núcleos, podemos considerar as funções eletrônicas dependendo parametricamente das posições nucleares e encontrá-las para diferente configurações de núcleos fixos. Em cada configuração o hamiltoniano é escrito como (VIANNA; CANUTO S.; FAZZIO, 2004) :

$$\hat{H} = \hat{H}_{el} + \hat{V}_N, \quad (2.2)$$

em que,

$$\hat{H}_{el} = \hat{T}_e + \hat{V}_{ee} + \hat{V}_{en}. \quad (2.3)$$

E as funções eletrônicas, $\chi_m(\mathbf{r}; \mathbf{R})$, podem ser encontrados pela equação:

$$\hat{H}_{el}\chi_m(\mathbf{r}; \mathbf{R}) = \varepsilon_m(R)\chi_m(\mathbf{r}; \mathbf{R}). \quad (2.4)$$

Assim, o termo de repulsão nuclear acrescenta um termo constante para a energia total de determinada configuração dos núcleos fixos:

$$E_m(\mathbf{R}) = \varepsilon_m(R) + \sum_A^M \sum_{B>A}^M \frac{Z_A Z_B}{|\mathbf{R}_A - \mathbf{R}_B|}. \quad (2.5)$$

E para encontrar a função de onda completa, podemos utilizar o conjunto completo $\{\chi_m(\mathbf{r}; \mathbf{R})\}$ para expandí-la

$$\Psi(\mathbf{r}, \mathbf{R}) = \sum_m \phi_m(\mathbf{R})\chi_m(\mathbf{r}; \mathbf{R}). \quad (2.6)$$

Encontramos os coeficientes da expansão $\phi_m(\mathbf{R})$ ao substituir a solução geral no hamiltoniano completo:

$$\hat{H} \sum_m \phi_m(\mathbf{R}) \chi_m(\mathbf{r}; \mathbf{R}) = E \sum_m \phi_m(\mathbf{R}) \chi_m(\mathbf{r}; \mathbf{R}), \quad (2.7)$$

$$\left\{ - \sum_A^M \frac{1}{2M_A} \nabla_A^2 + \sum_A^M \sum_{B>A}^M \frac{Z_A Z_B}{|\mathbf{R}_A - \mathbf{R}_B|} + \hat{H}_{el} - E \right\} \sum_m \phi_m(\mathbf{R}) \chi_m(\mathbf{r}; \mathbf{R}) = 0. \quad (2.8)$$

E assim,

$$\left\{ - \sum_A^M \frac{1}{2M_A} \nabla_A^2 + (E_m(\mathbf{R}) - E) \right\} \sum_m \phi_m(\mathbf{R}) \chi_m(\mathbf{r}; \mathbf{R}) = 0. \quad (2.9)$$

Sabendo da identidade vetorial $\nabla^2(\phi\chi) = \phi\nabla^2\chi + 2\nabla\phi\nabla\chi + \phi\nabla^2\phi$, a aplicação do operador de energia cinética nuclear pode ser reescrito como,

$$\begin{aligned} - \sum_m \sum_A^M \frac{1}{2M_A} \nabla_A^2 \phi_m(\mathbf{R}) \chi_m(\mathbf{r}; \mathbf{R}) &= - \sum_m \sum_A^M \frac{1}{2M_A} [\chi_m(\mathbf{r}; \mathbf{R}) \nabla^2 \phi_m(\mathbf{R}) + \\ &\quad + 2\nabla\phi_m(\mathbf{R}) \nabla\chi_m(\mathbf{r}; \mathbf{R}) + \phi_m(\mathbf{R}) \nabla^2 \chi_m(\mathbf{r}; \mathbf{R})]. \end{aligned} \quad (2.10)$$

Multiplicando à esquerda na eq.(2.9) por χ_n^* e integrando nas coordenadas eletrônicas, alguns termos ficam:

$$\begin{aligned} \sum_m (E_m(\mathbf{R}) - E) \phi_m(\mathbf{R}) \int \chi_n^*(\mathbf{r}; \mathbf{R}) \chi_m(\mathbf{r}; \mathbf{R}) d\mathbf{r} &= \sum_m (E_m(\mathbf{R}) - E) \phi_m(\mathbf{R}) \delta_{nm} \\ &= (E_n(\mathbf{R}) - E) \phi_n(\mathbf{R}), \end{aligned} \quad (2.11)$$

e,

$$\begin{aligned} - \sum_m \sum_{A=1}^M \frac{1}{2M_A} \nabla_A^2 \phi_m(\mathbf{R}) \int \chi_n^*(\mathbf{r}; \mathbf{R}) \chi_m(\mathbf{r}; \mathbf{R}) d\mathbf{r} &= - \sum_m \sum_{A=1}^M \frac{1}{2M_A} \nabla_A^2 \phi_m(\mathbf{R}) \delta_{nm} \\ &= \sum_{A=1}^M \frac{1}{2M_A} \nabla_A^2 \phi_n(\mathbf{R}), \end{aligned} \quad (2.12)$$

e para os outros termos a aplicação é direta. Assim,

$$\begin{aligned} - \sum_{A=1}^M \frac{1}{2M_A} \nabla_A^2 \phi_n(\mathbf{R}) + (E_n(\mathbf{R}) - E) \phi_n(\mathbf{R}) &= - \sum_m \sum_{A=1}^M \frac{1}{2M_A} \cdot \\ \left\{ 2 \int \chi_n^*(\mathbf{r}; \mathbf{R}) \nabla_A \chi_m(\mathbf{r}; \mathbf{R}) d^3 r \cdot \nabla_A + \int \chi_n^*(\mathbf{r}; \mathbf{R}) \nabla_A^2 \chi_m(\mathbf{r}; \mathbf{R}) d^3 r \right\} \phi_m(\mathbf{R}). \end{aligned} \quad (2.13)$$

Definindo,

$$C_{nm}(\mathbf{r}, \nabla) = \sum_{A=1}^M \frac{1}{M_A} (X_{nm}^{(A)} \nabla_A + Y_{nm}^{(A)}), \quad (2.14)$$

em que,

$$X_{nm}^{(A)} = \int \chi_n^*(\mathbf{r}; \mathbf{R}) \nabla_A \chi_m(\mathbf{r}; \mathbf{R}) d^3 r, \quad (2.15)$$

e,

$$Y_{nm}^{(A)} = \frac{1}{2} \int \chi_n^*(\mathbf{r}; \mathbf{R}) \nabla_A^2 \chi_m(\mathbf{r}; \mathbf{R}) d^3 r. \quad (2.16)$$

Assim, a equação 2.9 fica:

$$\left(- \sum_{A=1}^M \frac{1}{2M_A} \nabla_A^2 + E_n(\mathbf{R}) \right) \phi_n(\mathbf{R}) = E \phi_n(\mathbf{R}) + \sum_m C_{nm}(\mathbf{r}, \nabla) \phi_m(\mathbf{R}). \quad (2.17)$$

Separando os termos diagonais Y_{nn} e sabendo que $X_{nn} = 0$, reescrevemos:

$$\left(- \sum_{A=1}^M \frac{1}{2M_A} \nabla_A^2 + E_n(\mathbf{R}) - C_{nn}(\mathbf{r}, \nabla) \right) \phi_n(\mathbf{R}) = E \phi_n(\mathbf{R}) + \sum_{m \neq n} C_{nm}(\mathbf{r}, \nabla) \phi_m(\mathbf{R}). \quad (2.18)$$

Ao tornar somente o termo Y_{nn} não nulo (*aproximação adiabática*), obtemos a equação de Schrodinger independente do tempo para os núcleos:

$$\left(- \sum_{A=1}^M \frac{1}{2M_A} \nabla_A^2 + E_n(\mathbf{R}) - C_{nn}(\mathbf{r}, \nabla) \right) \phi_n(\mathbf{R}) = E \phi_n(\mathbf{R}) \quad (2.19)$$

E os núcleos se movem sobre o potencial efetivo:

$$\hat{V}_{ef} = E_n(\mathbf{R}) - C_{nn}(\mathbf{r}, \nabla). \quad (2.20)$$

A aproximação de Born- Oppenheimer consiste em considerar todos os termos de $C_{nm}(\mathbf{r}, \nabla)$ iguais a zero, em que os núcleos se movem sob o potencial efetivo $E_n(\mathbf{R})$, encontrado no problema eletrônico e a função de onda total é dada por:

$$\Psi(\mathbf{r}, \mathbf{R}) = \phi_n(\mathbf{R})\chi_n(\mathbf{r}; \mathbf{R}). \quad (2.21)$$

Embora tenhamos simplificado nosso problema, o termo de interação elétron-elétron (\hat{V}_e) não nos permite resolver a equação de Schorindger analiticamente, como fazemos para o átomo de hidrogênio e o oscilador harmônico. Sendo assim, recorremos a diversos métodos aproximativos, dentre eles Hartree-Fock e DFT. A partir desta seção estaremos focados no hamiltoniano eletrônico e, por isso, \hat{T} representará a energia cinética dos elétrons, \hat{U} a repulsão elétron-elétron e \hat{V} a atração elétron-núcleo.

2.2 Teoria Funcional da Densidade

A Teoria Funcional da Densidade revolucionou os cálculos de estrutura eletrônica ao reduzir o problema de N elétrons interagentes, descritos por uma função de onda de $3N$ variáveis espaciais acopladas, em encontrar a densidade eletrônica de 3 variáveis espaciais (HOHENBERG, 1964). As origens da teoria ocorreram com Thomas (THOMAS, 1927), Fermi (FERMI, 1927) e Dirac (DIRAC, 1930). Diferentemente do método Hartree-Fock, a DFT trata de funcionais da densidade eletrônica e sua base teórica está presente nos teoremas de Hohenberg e Kohn (RICHARD M.RICHARD, 2004)(VIANNA; CANUTO S.; FAZZIO, 2004). Primeiramente, vamos compreender o conceito de densidade eletrônica.

Assim como $|\psi(\mathbf{r})|^2$ representa a densidade de probabilidade de encontrar o elétron em um volume d^3r em torno de \mathbf{r} , $|\psi(r_1, r_2, \dots, r_N, \mathbf{R}_1, \mathbf{R}_2, \dots, \mathbf{R}_M)|^2$ representa a densidade de probabilidade de encontrar simultaneamente o elétron 1 em torno do ponto r_1 , o elétron 2 em torno do ponto r_2 , e assim sucessivamente. E a probabilidade é dada pelas integrais (FELICIANO, 2014):

$$\int |\psi(\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_N, \mathbf{R}_1, \mathbf{R}_2, \dots, \mathbf{R}_M)|^2 d\mathbf{r}_1 d\mathbf{r}_2 \dots d\mathbf{r}_N d\mathbf{R}_1 \dots d\mathbf{R}_M. \quad (2.22)$$

Enquanto a probabilidade de encontrar o elétron 1 na posição \mathbf{r} , e os outros elétrons em qualquer lugar é:

$$P(\mathbf{r}_1 = \mathbf{r}) = \int |\psi(\mathbf{r}, \mathbf{r}_2, \dots, \mathbf{r}_N, \mathbf{R}_1, \mathbf{R}_2, \dots, \mathbf{R}_M)|^2 d\mathbf{r}_2 d\mathbf{r}_3 \dots d\mathbf{r}_N d\mathbf{R}_1 \dots d\mathbf{R}_M. \quad (2.23)$$

A densidade eletrônica é definida como a probabilidade de encontrar qualquer elétron na posição \mathbf{r} , ou seja,

$$n(\mathbf{r}) = P(\mathbf{r}_1 = \mathbf{r}) + P(\mathbf{r}_2 = \mathbf{r}) + \dots + P(\mathbf{r}_N = \mathbf{r}). \quad (2.24)$$

Visto que os elétrons são partículas indistinguíveis, $P(\mathbf{r}_1 = \mathbf{r}) = P(\mathbf{r}_2 = \mathbf{r}) = \dots = P(\mathbf{r}_N = \mathbf{r})$, temos:

$$n(\mathbf{r}) = N \int |\psi(\mathbf{r}, \mathbf{r}_2, \dots, \mathbf{r}_N, \mathbf{R}_1, \mathbf{R}_2, \dots, \mathbf{R}_M)|^2 d\mathbf{r}_2 d\mathbf{r}_3 \dots d\mathbf{r}_N d\mathbf{R}_1 \dots d\mathbf{R}_M, \quad (2.25)$$

em que N é o número total de elétrons. Se a função de onda total é normalizada pela unidade, ou seja,

$$\int |\psi(\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_N, \mathbf{R}_1, \mathbf{R}_2, \dots, \mathbf{R}_M)|^2 d\mathbf{r}_1 d\mathbf{r}_2 \dots d\mathbf{r}_N d\mathbf{R}_1 \dots d\mathbf{R}_M = 1, \quad (2.26)$$

ao integrar a densidade de carga eletrônica em \mathbf{r} , obtemos:

$$\int n(\mathbf{r}) d\mathbf{r} = N. \quad (2.27)$$

Outra forma de expressar matematicamente a densidade é pelo valor esperado da função delta ([VIANNA; CANUTO S.; FAZZIO, 2004](#)),

$$n(\mathbf{r}) = \langle \psi | \sum_{i=1}^N \delta(\mathbf{r} - \mathbf{r}_i) | \psi \rangle. \quad (2.28)$$

2.2.1 Teoremas de Hohenberg-Kohn

Teorema 1. *Para qualquer sistema de partículas interagentes em um potencial externo $V_{ext}(r)$, esse potencial é determinado unicamente, exceto por uma constante, pela densidade do estado fundamental $n_0(r)$.*

Demonstração. Provamos o teorema por contradição. Suponhamos que existem dois potenciais externos $v_{ext}^{(1)}$ e $v_{ext}^{(2)}$ que levam a mesma densidade do estado fundamental $n_0(r)$. Associados a estes potenciais, os hamiltonianos $H^{(1)}$ e $H^{(2)}$ tem autofunções do estado fundamental distintas $\psi^{(1)}$ e $\psi^{(2)}$ e de acordo com o princípio variacional, a energia do estado fundamental associada a $v_{ext}^{(1)}$ é:

$$E^{(1)} = \langle \psi^{(1)} | H^{(1)} | \psi^{(1)} \rangle. \quad (2.29)$$

Como $\psi^{(1)}$ é a função de onda que minimiza o funcional $E^{(1)}[\psi]$, a seguinte relação se estabelece:

$$E^{(1)} = \langle \psi^{(1)} | H^{(1)} | \psi^{(1)} \rangle < \langle \psi^{(2)} | H^{(1)} | \psi^{(2)} \rangle. \quad (2.30)$$

Reescrevendo o termo à direta:

$$\begin{aligned} \langle \psi^{(2)} | H^{(1)} | \psi^{(2)} \rangle &= \langle \psi^{(2)} | H^{(1)} + H^{(2)} - H^{(2)} | \psi^{(2)} \rangle = \\ &= \langle \psi^{(2)} | H^{(2)} | \psi^{(2)} \rangle + \langle \psi^{(2)} | H^{(1)} - H^{(2)} | \psi^{(2)} \rangle. \end{aligned} \quad (2.31)$$

Como os termos de energia cinética e interação elétron-elétron são os mesmos, temos:

$$\langle \psi^{(2)} | H^{(2)} | \psi^{(2)} \rangle + \langle \psi^{(2)} | V_{ext}^{(1)} - V_{ext}^{(2)} | \psi^{(2)} \rangle. \quad (2.32)$$

Então,

$$E^{(1)} < \langle \psi^{(2)} | H^{(2)} | \psi^{(2)} \rangle + \langle \psi^{(2)} | V_{ext}^{(1)} - V_{ext}^{(2)} | \psi^{(2)} \rangle. \quad (2.33)$$

Como,

$$\hat{V}_{ext} = \sum_{\mu=1}^N \left(- \sum_{A=1}^M \frac{Z_A}{|r_\mu - R_A|} \right) = \sum_{\mu=1}^N v_{ext}(r_\mu), \quad (2.34)$$

seu valor esperado é ([ALVES, 2011](#)),

$$\begin{aligned} \langle \Psi | \sum_{\mu=1}^N v_{ext}(r_\mu) | \Psi \rangle &= \sum_{\mu=1}^N \langle \Psi | \int d\mathbf{r} \delta(\mathbf{r} - \mathbf{r}_\mu) v_{ext}(\mathbf{r}) | \Psi \rangle = \\ &= \int d\mathbf{r} v_{ext}(\mathbf{r}) \underbrace{\langle \Psi | \sum_{\mu=1}^N \delta(\mathbf{r} - \mathbf{r}_\mu) | \Psi \rangle}_{n_0(\mathbf{r})} = \\ &= \int d\mathbf{r} v_{ext}(\mathbf{r}) n_0(\mathbf{r}). \end{aligned} \quad (2.35)$$

Logo,

$$E^{(1)} < E^{(2)} + \int d^3(\mathbf{r}) [v_{ext}^{(1)} - v_{ext}^{(2)}] n_0(\mathbf{r}). \quad (2.36)$$

No entanto, se começarmos a demonstração de forma análoga para $E^{(2)}$, obtemos:

$$E^{(2)} < E^{(1)} + \int d^3(\mathbf{r}) [v_{ext}^{(2)} - v_{ext}^{(1)}] n_0(\mathbf{r}). \quad (2.37)$$

Somando as duas equações:

$$E^{(1)} + E^{(2)} < E^{(2)} + E^{(1)}, \quad (2.38)$$

o que é um absurdo, logo $v_{ext}^{(1)} = v_{ext}^{(2)}$.

□

Como a densidade determina unicamente o potencial externo e assim todas as auto-funções do hamiltoniano, temos que todas as propriedades do sistema são determinadas somente pela densidade do estado fundamental, ou igualmente, que os valores esperados de todos os observáveis são funcionais únicos da densidade do estado fundamental:

$$\langle \psi | O | \psi \rangle = O[n_0(r)].$$

Teorema 2. *Pode-se definir um funcional universal da energia $E[n(r)]$, válido para qual-*

quer potencial externo v_{ext} . A densidade e a energia do estado fundamental são tais que minimizam esse funcional.

Demonstração. Podemos escrever de forma geral a energia como funcional da densidade da seguinte forma:

$$E[n(\mathbf{r})] = \langle \Psi | \hat{T} + \hat{U} + \hat{V} | \Psi \rangle, \quad (2.39)$$

$$E[n(\mathbf{r})] = \langle \Psi | \hat{T} + \hat{U} | \Psi \rangle + \langle \Psi | \hat{V} | \Psi \rangle, \quad (2.40)$$

$$E[n(\mathbf{r})] = F[n(\mathbf{r})] + \langle \Psi | \hat{V} | \Psi \rangle, \quad (2.41)$$

em que somente o último termo não é universal e depende do material analisado.

Seja a densidade do estado fundamental $n_0(r)$ associada a ψ_0 , então a energia desse estado é:

$$E[\psi_0] = \langle \psi_0 | H | \psi_0 \rangle.$$

E seja qualquer outra densidade $n^{(i)}$ associada a função de onda $\psi^{(i)}$, pelo teorema variacional:

$$E[\psi_0] = \langle \psi_0 | H | \psi_0 \rangle < \langle \psi^{(i)} | H | \psi^{(i)} \rangle,$$

que é equivalente à

$$E_0 = E[n_0] < E[n].$$

Portanto, a densidade que minimiza o funcional é a densidade o estado fundamental.

□

2.2.2 Equações de Kohn-Sham

Após os teoremas de Hohenberg e Kohn, podemos encontrar a energia do estado fundamental dada a densidade eletrônica e esta é função apenas de três variáveis espaciais. Ou seja,

$$E[n(\mathbf{r})] = F[n(\mathbf{r})] + \langle \Psi | \hat{V} | \Psi \rangle. \quad (2.42)$$

No entanto, não sabemos a forma exata do funcional $F[n]$. Ao reescrever o funcional da energia cinética, temos:

$$\hat{T}[n] = \hat{T}_s[n] + \hat{T}_c[n], \quad (2.43)$$

em que $\hat{T}_s[n]$ representa a energia cinética de elétrons não interagentes e $T_c[n]$ representa a energia cinética adicional em razão da correlação quântica. Mesmo que o termo $\hat{T}_s[n]$ não seja conhecido, ele pode ser escrito como um funcional dos orbitais $\{\phi_i\}$ e estes são funcionais da densidade.

$$T_s[n] = -\frac{1}{2} \sum_{i=1}^N \langle \phi_i | \nabla^2 | \phi_i \rangle. \quad (2.44)$$

Já para a repulsão elétron-elétron,

$$U[n] = U_H[n] + U_{xc}[n], \quad (2.45)$$

em que $\hat{U}_H[n]$ representa a interação colombiana clássica (termo de Hartree) e $\hat{U}_{xc}[n]$ representa parte da energia de correlação e a energia de troca. Logo, o funcional $F[n]$ fica,

$$F[n] = T_s[n] + T_c[n] + U_H[n] + U_{xc}[n], \quad (2.46)$$

$$F[n] = -\frac{1}{2} \sum_{i=1}^N \langle \phi_i | \nabla^2 | \phi_i \rangle + \int d^3r d^3r' \frac{n(r)n(r')}{|r - r'|} + E_{xc}[n], \quad (2.47)$$

no qual,

$$U_H[n] = \int d^3r d^3r' \frac{n(r)n(r')}{|r - r'|}, \quad (2.48)$$

e a energia de troca e correlação,

$$E_{xc}[n] = T_c[n] + U_{xc}[n]. \quad (2.49)$$

Então o funcional da energia assume a forma,

$$E[n] = T_s[n] + E_{xc}[n] + \int d^3r v(\mathbf{r})n(\mathbf{r}) + \frac{1}{2} \int d^3r d^3r' \frac{n(\mathbf{r}).n(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|}. \quad (2.50)$$

O primeiro termo do funcional se refere à energia cinética de um gás de elétrons não

interagentes. O termo posterior descreve a energia de troca e correlação e o seguinte se refere à interação coulombiana clássica entre os elétrons e os núcleos. Além disso, o último termo especifica a energia coulombiana clássica entre os elétrons, geralmente chamado de termo de Hartree.

Pelo segundo teorema de Hohenberg e Kohn sabemos que a densidade do estado fundamental é tal que minimiza esse funcional com o vínculo do número de partículas fixo:

$$\int d^3r n(r) = N. \quad (2.51)$$

Utilizando os multiplicadores de Lagrange ϵ_i ,

$$\delta \left\{ E[n] - \epsilon_i \left[\int n(r) d^3r - N \right] \right\} = 0,$$

obtemos pela definição de derivada funcional,

$$\int \delta n(r) \left\{ \frac{\delta}{\delta n(r)} \left(E[n] - \epsilon_i \left[\int n(r) d^3r - N \right] \right) \right\} d^3r = 0,$$

$$\int \delta n(r) \left\{ \frac{\delta E[n]}{\delta n} - \epsilon_i \frac{\delta}{\delta n} \int n(r) d^3r - \underbrace{\epsilon_i \frac{\delta N}{\delta n}}_{=0} \right\} d^3r = 0,$$

$$\int \delta n(r) \left\{ \frac{\delta E[n]}{\delta n} - \epsilon_i \underbrace{\frac{\delta}{\delta n} \int n(r) d^3r}_{=1} \right\} d^3r = 0,$$

$$\int \delta n(r) \left\{ \frac{\delta E[n]}{\delta n} - \epsilon_i \right\} d^3r = 0.$$

As derivadas dos termos de $E[n]$ que possuem forma funcional explícita também podem ser encontrados por sua variação,

$$\begin{aligned} \delta \left(\int v(\mathbf{r}) n(\mathbf{r}) d^3r \right) &= \int v(\mathbf{r}) [n(\mathbf{r}) + \delta n(\mathbf{r}) - n(\mathbf{r})] d^3r = \\ &= \int v(\mathbf{r}) \delta n(\mathbf{r}) d^3r \Rightarrow \end{aligned}$$

$$\Rightarrow \frac{\delta}{\delta n} \left(\int v(\mathbf{r}) n(\mathbf{r}) d^3 r \right) = v(\mathbf{r}).$$

E para a energia de Hartree,

$$\begin{aligned} \delta U_H[n] &= \frac{1}{2} \int \int d^3 r d^3 r' \frac{[n(\mathbf{r}) + \delta n(\mathbf{r})][n(\mathbf{r}') + \delta n(\mathbf{r}')] }{|\mathbf{r} - \mathbf{r}'|} - \frac{1}{2} \int \int d^3 r d^3 r' \frac{n(\mathbf{r}) n(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} \approx \\ &\approx \int \int d^3 r d^3 r' \frac{n(\mathbf{r}') \delta n(\mathbf{r})}{|\mathbf{r} - \mathbf{r}'|} = \int \delta n(\mathbf{r}) d^3 r \int d^3 r' \frac{n(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} \Rightarrow \\ &\Rightarrow \frac{\delta U_H[n]}{\delta n} = \int d^3 r' \frac{n(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|}. \end{aligned}$$

Finalmente, fazendo $v_{xc}[n] = \frac{\delta E_{xc}}{\delta n}$, temos:

$$\int \delta n(r) \left[\frac{\delta T_s}{\delta n} + v_{xc}[n] + \int d^3 r' \frac{n(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} + v(\mathbf{r}) - \epsilon_i \right] d^3 r = 0. \quad (2.52)$$

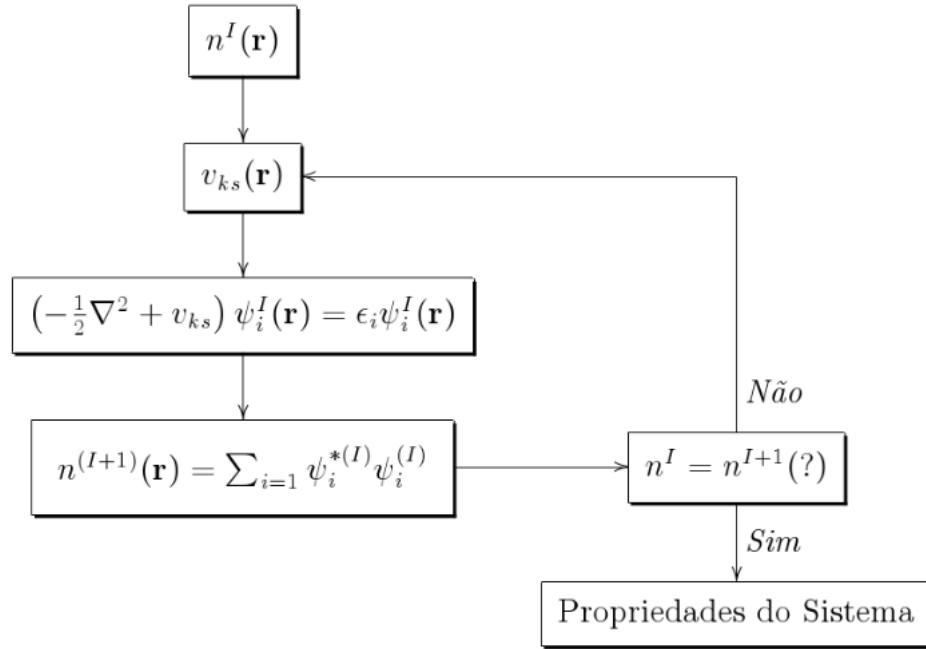
Introduzindo a densidade em termos dos orbitais,

$$n(\mathbf{r}) = \sum_i^N \phi_i^*(\mathbf{r}) \phi_i(\mathbf{r}), \quad (2.53)$$

chegamos na equação de Kohn-Sham:

$$\left[-\frac{1}{2} \nabla_i^2 + \frac{\delta E_{xc}}{\delta n} + \int d^3 r' \frac{n(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} + v(\mathbf{r}) \right] \phi_i = \epsilon_i \phi_i. \quad (2.54)$$

A equação de Kohn-Sham pode ser percebida como uma equação de elétrons independentes sob um potencial efetivo e a solução deste problema auxiliar é suficiente para determinar a energia total. No entanto, os orbitais dependem da densidade e ela é escrita em termos dos orbitais. Por esse motivo, o processo é autoconsistente. O primeiro passo reduz-se em resolver a equação com uma densidade eletrônica escolhida. Posteriormente, as soluções são utilizadas para construir a densidade resultante. O processo se repete até que a densidade final seja igual a densidade inicial, dentro do critério de convergência definido. A seguir, está esquematizada a técnica:

Figura 1: Ciclo autoconsistente ([ALVES, 2011](#)).

2.2.3 Funcionais de Troca e Correlação

Embora a exatidão da teoria seja garantida pelos teoremas de Hohenberg-Kohn, a forma do funcional de troca e correlação não é precisamente conhecida. Aqui surge a primeira aproximação após a aproximação de Born-Oppenheimer que por consequência não nos permite encontrar a densidade exata do estado fundamental ([SHOLL D.S.; STECKEL, 2009](#)). A primeira abordagem foi proposta por Hohenberg e Kohn em 1965 com a *local density approximation* (LDA), ao tratar localmente a densidade como a densidade de um gás de elétrons homogêneo², ou seja, a energia de troca e correlação em cada ponto do sistema é de um gás de elétrons homogêneo com a mesma densidade ([KOHN W.; SHAM, 1965](#)). Assim,

$$E_{xc}^{LDA} = \int d\mathbf{r} n(\mathbf{r})\epsilon_{xc}[n(\mathbf{r})], \quad (2.55)$$

em que $\epsilon_{xc}[n(\mathbf{r})]$ é a energia de troca e correlação por partícula de um gás de elétrons homogêneo de densidade $n(\mathbf{r})$. O potencial associado é sua derivada funcional:

²Em um gás homogêneo, os elétrons interagem entre si e estão submetidos a um potencial constante dos núcleos.

$$v_{xc} = \frac{\delta E_{xc}^{LDA}}{\delta n(\mathbf{r})}. \quad (2.56)$$

O termo de troca se refere à indistinguibilidade dos elétrons e o termo de correlação se refere as demais interações quânticas que não estão presentes no termo de troca. Logo, podemos considerar suas contribuições separadamente:

$$\epsilon_{xc}[n(\mathbf{r})] = \epsilon_x[n(\mathbf{r})] + \epsilon_c[n(\mathbf{r})]. \quad (2.57)$$

O termo de troca é obtido exatamente e sua forma em termos do raio de Wigner-Seitz, $r_s = \sqrt[3]{\frac{3}{4\pi n}}$, é:

$$\epsilon_x[n(\mathbf{r})] = -\frac{0,4582}{r_s}. \quad (2.58)$$

O termo de correlação não é obtido de maneira exata. Entretanto, Ceperley e Alder, empregando o método de Monte Carlo Quântico, obtiveram $\epsilon_c[n]$ para diferentes valores da densidade eletrônica ([CERPELEY D. M. ALDER, 1980](#)). Entre as parametrizações feitas para ϵ_{xc} , a parametrização de Perdew-Zunger é a mais utilizada e o termo de correlação é dado por:

$$\epsilon_c = -\frac{0,1423}{1 + 1,95529\sqrt{r_s} + 0,3334r_s}; \quad r_s \geq 1, \quad (2.59)$$

ou,

$$\epsilon_c = -0,0480 + 0,0311\ln r_s - 0,0116r_s; \quad r_s < 1, \quad (2.60)$$

no qual $r_s \geq 1$ é para baixas densidades e $r < 1$ para altas densidades eletrônicas ([ALVES, 2011](#)).

Para sistemas que a densidade varia fortemente, a aproximação local da densidade não gera bons resultados ([VIANNA; CANUTO S.; FAZZIO, 2004](#)). Para descrever de uma maneira mais precisa esta variação, foi criada uma nova aproximação, denominada GGA³. Nesse caso, em cada ponto consideramos não somente a densidade, mas seu gradiente. Assim,

$$E_{xc}^{GGA} = \int f(n(\mathbf{r}), \nabla n(\mathbf{r})) d\mathbf{r}. \quad (2.61)$$

³Do inglês, Generalized Gradient Approximation

Existem diferentes formas do funcional GGA, incluindo funcionais híbridos⁴. O funcional PBE (PERDEW J. P.; ERNZERHOF M.; BURKE, 1996) é muito utilizado, não possui parâmetros empíricos e descreve ambos os termos de troca e correlação (SANTOS, 2017).

2.3 Elétrons em um potencial periódico

Teorema de Bloch

Sabendo-se que os cristais possuem um arranjo periódico, é coerente supor que os elétrons estão em potencial efetivo $U(\mathbf{r})$ que possue a mesma periodicidade da rede, ou seja, $U(\mathbf{r} + \mathbf{R}) = U(\mathbf{r})$. Assim, dado que os elétrons neste potencial efetivo são independentes, o hamiltoniano do sistema tem o formato,

$$\hat{H}\psi = \left(-\frac{\hbar^2}{2m}\nabla^2 + U(\mathbf{r})\right)\psi = \epsilon\psi. \quad (2.62)$$

O seguinte teorema enuncia o que sabemos da solução desta equação diferencial, mesmo sem saber a forma de $U(\mathbf{r})$ (CAPAZ, 2019).

Teorema 3. *Os autoestados ψ do hamiltoniano de um elétron $H = -\hbar^2/2m + U(\mathbf{r})$, onde $U(\mathbf{r} + \mathbf{R}) = U(\mathbf{r})$ para todos os vetores da rede de Bravais, tem a forma de uma onda plana multiplicada por uma função com a periodicidade da rede de Bravais:*

$$\psi_{n\mathbf{k}}(\mathbf{r}) = e^{i\mathbf{k}\cdot\mathbf{r}}u_{n\mathbf{k}}(\mathbf{r}), \quad (2.63)$$

em que

$$u_{n\mathbf{k}}(\mathbf{r} + \mathbf{R}) = u_{n\mathbf{k}}(\mathbf{r}). \quad (2.64)$$

para todo R na rede de Bravais.

Nota-se que em geral a função de onda não é periódica. Utilizando as equações anteriores, obtemos:

$$\psi_{n\mathbf{k}}(\mathbf{r} + \mathbf{R}) = e^{i\mathbf{k}\cdot(\mathbf{r} + \mathbf{R})}u_{n\mathbf{k}}(\mathbf{r} + \mathbf{R}) = e^{i\mathbf{k}\cdot\mathbf{r}}e^{i\mathbf{k}\cdot\mathbf{R}}u_{n\mathbf{k}}(\mathbf{r}) = e^{i\mathbf{k}\cdot\mathbf{R}}\psi_{n\mathbf{k}}(\mathbf{r}). \quad (2.65)$$

⁴Mistura-se o termo de troca de Hartree-Fock e o termo de troca do DFT.

Ou seja, ao transladar a função de onda por um vetor da rede de Bravais \mathbf{R} , a função de onda adquire uma fase global $e^{i\mathbf{k}\cdot\mathbf{R}}$. Esta é uma maneira diferente de enunciar o teorema de Bloch.

Demonstração. Seja $T_{\mathbf{R}}$ o operador de translação que aplicado à uma função, desloca seu argumento por um vetor da rede de Bravais,

$$T_{\mathbf{R}}f(\mathbf{r}) = f(\mathbf{r} + \mathbf{R}). \quad (2.66)$$

Como o hamiltoniano é invariante sob essas translações, temos:

$$T_{\mathbf{R}}[H(\mathbf{r})\psi(\mathbf{r})] = H(\mathbf{r} + \mathbf{R})\psi(\mathbf{r} + \mathbf{R}) = H(\mathbf{r})\psi(\mathbf{r} + \mathbf{R}) = H(\mathbf{r})T_{\mathbf{R}}\psi(\mathbf{r}), \quad (2.67)$$

ou seja, o hamiltoniano comuta com $T_{\mathbf{R}}$, $[H(\mathbf{r}), T_{\mathbf{R}}] = 0$. Do mesmo modo, dois operadores $T_{\mathbf{R}}$ e $T_{\mathbf{R}'}$ também comutam,

$$T_{\mathbf{R}}T_{\mathbf{R}'}\psi(\mathbf{r}) = \psi(\mathbf{r} + \mathbf{R} + \mathbf{R}') = \psi(\mathbf{r} + \mathbf{R}' + \mathbf{R}) = T_{\mathbf{R}'}T_{\mathbf{R}}\psi(\mathbf{r}). \quad (2.68)$$

Além disso,

$$T_{\mathbf{R}}T_{\mathbf{R}'} = T_{\mathbf{R}'}T_{\mathbf{R}} = T_{\mathbf{R}+\mathbf{R'}}. \quad (2.69)$$

Logo, o conjunto $H, T_{\mathbf{R}}, T_{\mathbf{R}'}, T_{\mathbf{R}''}, \dots$ possui as mesmas autofunções:

$$\begin{aligned} H(\mathbf{r})\psi &= \epsilon\psi, \\ T_{\mathbf{R}}\psi &= c(\mathbf{R})\psi. \end{aligned} \quad (2.70)$$

Nosso objetivo é determinar os autovalores de $T_{\mathbf{R}}$. Ao aplicar duas translações sucessivas, temos:

$$T_{\mathbf{R}}T_{\mathbf{R}'}\psi = c(\mathbf{R})c(\mathbf{R}')\psi. \quad (2.71)$$

E utilizando a eq.(2.67), concluímos:

$$c(\mathbf{R})c(\mathbf{R}') = c(\mathbf{R} + \mathbf{R}'). \quad (2.72)$$

Matematicamente, a função de onda que contém tal propriedade é a função exponencial, seja ela complexa ou real. Assim:

$$c(\mathbf{R}) = e^{i\mathbf{k}\cdot\mathbf{R}}. \quad (2.73)$$

Portanto, a função de onda transladada difere da exponencial $e^{i\mathbf{k}\cdot\mathbf{R}}$. Fisicamente ela deve ser complexa, pois uma fase global não muda a probabilidade. É importante lembrar que estamos levando em conta um cristal de periodicidade perfeita e não há motivos para a probabilidade mudar ao transladar por um vetor da rede \mathbf{R} , afinal a vizinhança para o elétron é idêntica. Vamos demonstrar esta afirmação de outra forma, através das condições de contorno de Born-Von Karman. \square

Condições de Contorno de Born-Von Karman

Seja um cristal com dimensões L_1 , L_2 e L_3 nas direções dos vetores primitivos \mathbf{a}_1 , \mathbf{a}_2 e \mathbf{a}_3 . Um vetor da rede em alguma dessas direções pode ser escrito de maneira geral como $\mathbf{R}_i = N_i \mathbf{a}_i$, em que N_i é o número de células primitivas na direção i . Uma possibilidade para as condições de contorno seria anular a função de onda na superfície desse cristal. No entanto, tais condições levariam a soluções de onda estacionária. Porém, transporte de carga e energia são melhor descritos com ondas propagantes (NEIL W. A, 1976). Sendo assim, aplicamos as condições de contorno de Born-von Karman⁵ e obtemos:

$$\psi(\mathbf{r}) = \psi(\mathbf{r} + N_i \mathbf{a}_i) = e^{i\mathbf{k}\cdot(N_i \mathbf{a}_i)} \psi(\mathbf{r}), \quad (2.74)$$

que resulta em $e^{i\mathbf{k}\cdot(N_i \mathbf{a}_i)} = 1$. Como os vetores primitivos da rede recíproca, \mathbf{b}_1 , \mathbf{b}_2 e \mathbf{b}_3 , formam uma base no espaço recíproco, podemos escrever o vetor de onda \mathbf{k} nessa base:

$$\mathbf{k} = x_1 \mathbf{b}_1 + x_2 \mathbf{b}_2 + x_3 \mathbf{b}_3. \quad (2.75)$$

Utilizando a definição dos vetores primitivos da rede recíproca, $\mathbf{a}_i \cdot \mathbf{b}_j = 2\pi\delta_{ij}$ e o resultado da eq.(2.74), temos:

$$e^{i2\pi N_i x_i} = 1 \Rightarrow x_i = \frac{m_i}{N_i}, \quad m_i \text{ inteiro.} \quad (2.76)$$

Como as componentes x_i são reais, o vetor de onda \mathbf{k} também é real. A forma geral do vetor de onda é:

$$\mathbf{k} = \sum_{i=1}^3 \frac{m_i}{N_i} \mathbf{b}_i. \quad (2.77)$$

⁵As condições periódicas de Born-von Karman afirmam que $\psi(\mathbf{r} + \mathbf{R}) = \psi(\mathbf{r})$.

Dessa expressão, vemos que o volume ocupado no espaço recíproco por qualquer \mathbf{k} é:

$$\Delta\mathbf{k} = \frac{\mathbf{b}_1}{N_1} \cdot \left(\frac{\mathbf{b}_2}{N_2} \times \frac{\mathbf{b}_3}{N_3} \right) = \frac{1}{N} \mathbf{b}_1 \cdot (\mathbf{b}_2 \times \mathbf{b}_3). \quad (2.78)$$

Desde que $\mathbf{b}_1 \cdot (\mathbf{b}_2 \times \mathbf{b}_3)$ é o volume da célula primitiva na rede recíproca, concluímos que *o número de vetores \mathbf{k} permitidos na célula unitária da rede recíproca é igual ao número de células primitivas na rede direta*. A expressão acima também fornece $\Delta\mathbf{k}$ em função do volume total V do cristal no espaço direto:

$$\Delta\mathbf{k} = \frac{(2\pi)^3}{V}. \quad (2.79)$$

Uma vez que as funções de Bloch não são autofunções do operador \mathbf{p} , os vetores de onda \mathbf{k} não são o momento do elétron⁶. Além disso, todos os vetores \mathbf{k} podem ser confinados na primeira zona de Brillouin. Como qualquer \mathbf{k}' que não pertença a essa região pode ser reescrito como $\mathbf{k} + \mathbf{G}$, onde \mathbf{k} é um vetor da primeira zona e \mathbf{G} é um vetor da rede recíproca, temos:

$$e^{i\mathbf{k}' \cdot \mathbf{R}} \psi(\mathbf{r}) = e^{i(\mathbf{k} + \mathbf{G}) \cdot \mathbf{R}} \psi(\mathbf{r}) = e^{i\mathbf{k} \cdot \mathbf{R}} \underbrace{e^{i\mathbf{G} \cdot \mathbf{R}}}_{=1} \psi(\mathbf{r}) = e^{i\mathbf{k} \cdot \mathbf{R}} \psi(\mathbf{r}). \quad (2.80)$$

2.4 Base de Ondas Planas e Estrutura de Bandas

O método computacional se torna prático para resolver a equação de Kohn-Sham quando encontramos um conjunto de funções base que a torne uma equação algébrica e assim permita utilizar a diagonalização de matrizes. Como a equação denota um sistema de elétrons não interagentes sob o potencial periódico efetivo $V_{ef}(\mathbf{r})$, podemos utilizar os resultados do teorema de Bloch. A equação de Kohn-Sham pode ser reescrita como (ALVES, 2011):

$$\left[-\frac{1}{2} \nabla^2 + V_{ef}(\mathbf{r}) \right] \psi_{n\mathbf{k}} = \epsilon_n(\mathbf{k}) \psi_{n\mathbf{k}}. \quad (2.81)$$

Logo, a periodicidade da função $u_{n\mathbf{k}}(\mathbf{r})$ nos permite expandí-la em ondas planas com vetores da rede recíproca \mathbf{G} , que leva a:

⁶ $\hbar\mathbf{k}$ é qualificado como *momento cristalino*

$$\psi_{n\mathbf{k}} = \sum_{\mathbf{G}} C_{n\mathbf{k}}(\mathbf{G}) e^{i(\mathbf{k}+\mathbf{G}) \cdot \mathbf{r}}. \quad (2.82)$$

Como o potencial também é periódico, podemos igualmente reescrevê-lo como uma série de Fourier com vetores da rede recíproca \mathbf{Q} :

$$V_{ef}(\mathbf{r}) = \sum_{\mathbf{Q}} V_{\mathbf{Q}} e^{i\mathbf{Q} \cdot \mathbf{r}}. \quad (2.83)$$

Substituindo a eq.(2.82) na eq.(2.81), temos:

$$\left[-\frac{1}{2} \nabla^2 + V_{ef}(\mathbf{r}) \right] \left(\sum_{\mathbf{G}} C_{n\mathbf{k}}(\mathbf{G}) e^{i(\mathbf{k}+\mathbf{G}) \cdot \mathbf{r}} \right) = \epsilon_n(\mathbf{k}) \left(\sum_{\mathbf{G}} C_{n\mathbf{k}}(\mathbf{G}) e^{i(\mathbf{k}+\mathbf{G}) \cdot \mathbf{r}} \right). \quad (2.84)$$

A aplicação do operador energia cinética na função de onda leva à:

$$-\frac{\nabla^2}{2} \psi_{n\mathbf{k}} = \frac{1}{2} \sum_{\mathbf{G}} C_{n\mathbf{k}}(\mathbf{G}) (\mathbf{k} + \mathbf{G})^2 e^{i(\mathbf{k}+\mathbf{G}) \cdot \mathbf{r}}. \quad (2.85)$$

E a aplicação de V_{ef} leva à:

$$V_{ef}(\mathbf{r}) = \sum_{\mathbf{Q}} \sum_{\mathbf{G}} V_{\mathbf{Q}} C_{n\mathbf{k}}(\mathbf{G}) e^{i(\mathbf{k}+\mathbf{G}+\mathbf{Q}) \cdot \mathbf{r}}. \quad (2.86)$$

Além disso, multiplicando a eq.(2.84) à esquerda por $e^{-i(\mathbf{k}+\mathbf{G}')}$ e integrando sob o volume da célula unitária Ω para utilizar a ortogonalidade das ondas planas, obtemos:

$$\begin{aligned} \frac{1}{2} \sum_{\mathbf{G}} C_{n\mathbf{k}}(\mathbf{G}) (\mathbf{k} + \mathbf{G})^2 & \underbrace{\int d\mathbf{r} e^{-i(\mathbf{k}+\mathbf{G}') \cdot \mathbf{r}} e^{i(\mathbf{k}+\mathbf{G}) \cdot \mathbf{r}}}_{\Omega \delta_{\mathbf{G}, \mathbf{G}'}} + \epsilon_n(\mathbf{k}) \underbrace{\int d\mathbf{r} e^{-i(\mathbf{k}+\mathbf{G}') \cdot \mathbf{r}} e^{i(\mathbf{k}+\mathbf{G}) \cdot \mathbf{r}}}_{\Omega \delta_{\mathbf{G}, \mathbf{G}'}} + \\ & + \sum_{\mathbf{Q}} \sum_{\mathbf{G}} V_{\mathbf{Q}} C_{n\mathbf{k}}(\mathbf{G}) \underbrace{\int d\mathbf{r} e^{-i(\mathbf{k}+\mathbf{G}') \cdot \mathbf{r}} e^{i(\mathbf{k}+\mathbf{G}+\mathbf{Q}) \cdot \mathbf{r}}}_{\Omega \delta_{\mathbf{Q}, \mathbf{G}' - \mathbf{G}}}. \end{aligned} \quad (2.87)$$

Portanto,

$$\left[\frac{1}{2} (\mathbf{k} + \mathbf{G})^2 + \epsilon_n(\mathbf{k}) \right] C_{n\mathbf{k}}(\mathbf{G}) + \sum_{\mathbf{G}} V_{\mathbf{G}' - \mathbf{G}} C_{n\mathbf{k}}(\mathbf{G}) = 0. \quad (2.88)$$

Como há para cada G uma soma sobre todos os vetores da rede recíproca, é mais conveniente realizar a mudança de índice $G \rightarrow G'$ e $G' \rightarrow G$ no somatório. Logo:

$$\left[\frac{1}{2}(\mathbf{k} + \mathbf{G})^2 + \epsilon_n(\mathbf{k}) \right] C_{n\mathbf{k}}(\mathbf{G}) + \sum_{G'} V_{\mathbf{G}-\mathbf{G}'} C_{n\mathbf{k}}(\mathbf{G}') = 0. \quad (2.89)$$

Sendo $V_{\mathbf{G}-\mathbf{G}'}$ a transformada de Fourier do potencial cristalino dada por:

$$V_{\mathbf{G}-\mathbf{G}'} = \int_{\Omega} d\mathbf{r} V_{ef}(\mathbf{r}) e^{i(\mathbf{G}-\mathbf{G}') \cdot \mathbf{r}}. \quad (2.90)$$

Ao fatorar o termo $C_{n\mathbf{k}}(\mathbf{G}')$ na expressão (2.89), obtemos:

$$\sum_{G'} \left\{ \left[\frac{1}{2}(\mathbf{k} + \mathbf{G}')^2 + \epsilon_n(\mathbf{k}) \right] \delta_{\mathbf{G},\mathbf{G}'} + V_{\mathbf{G}-\mathbf{G}'} \right\} C_{n\mathbf{k}}(\mathbf{G}') = 0. \quad (2.91)$$

Fixado o valor de \mathbf{k} , este é um conjunto infinito de equações acopladas, uma para cada G da rede recíproca, com G' variáveis. Logo, para determinar os autovalores $\epsilon_n(\mathbf{k})$, diagonalizamos a matriz quadrada do hamiltoniano:

$$\hat{H} = \frac{1}{2}(\mathbf{k} + \mathbf{G}')^2 \delta_{\mathbf{G},\mathbf{G}'} + V_{\mathbf{G}-\mathbf{G}'}, \quad (2.92)$$

cujo tamanho é definido pela energia de corte da maneira subsequente:

$$\frac{1}{2}(\mathbf{k} + \mathbf{G})^2 \leq E_{cut}. \quad (2.93)$$

Ademais, para cada \mathbf{k} há um conjunto discreto n (índice de banda) de autovalores. As soluções configuram uma estrutura de bandas, que está exemplificada na figura abaixo.

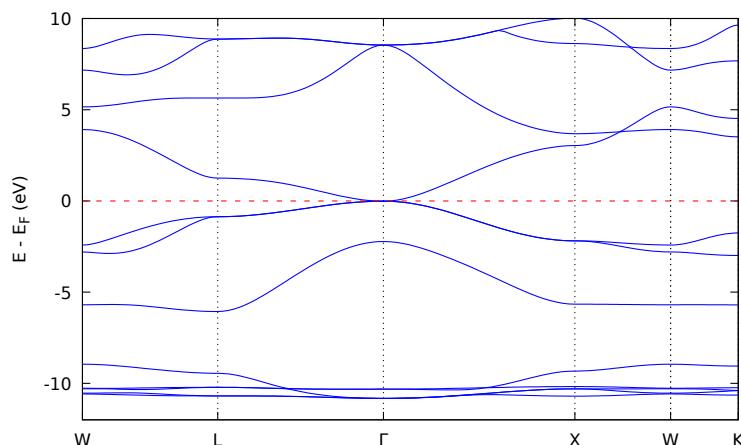


Figura 2: Estrutura de Bandas do TiN na estrutura zinc-blend. O nível de Fermi E_F foi deslocado para a origem.

A figura 2 representa a estrutura de bandas do Nitreto de Tálio (TIN) ([ALMEIDA, 2020](#)). Como vimos ao discutir as condições de contorno de Born-Von Karman, pode-se escolher calcular $\epsilon_n(k_x, k_y, k_z)$ dentro da primeira zona de brillouin. Ainda assim, sua interpretação gráfica requer quatro dimensões. Para solucionar esse obstáculo, o cálculo é feito ao longo de um caminho que inclui pontos de alta simetria, tais como Γ (G), X , W , K , etc.

Como a quantidade de pontos k depende do número de células unitárias N no espaço real, cada banda possui $2N$ estados considerando o spin. Em $T = 0K$, os elétrons se organizam no estado fundamental começando a preencher os níveis de menor energia até o nível de Fermi E_F , obedecendo o princípio de exclusão de Pauli. Dependendo da quantidade de elétrons, a última banda de valência pode estar parcialmente ou totalmente preenchida, gerando comportamentos elétricos distintos ([CAPAZ, 2019](#)).

Se a quantidade de elétrons é ímpar, a última banda não ficará completamente ocupada, pois em cada banda é permitido um número par de elétrons ($2N$). Consequentemente, existem estados imediatamente superiores que estão livres para serem ocupados por algum tipo de excitação. Materiais com a última banda ocupada semi-preenchida são metálicos e a ocupação dos elétrons está explanada na figura 3. Ademais, a condutividade de um metal diminui com a temperatura, visto que pode ativar modos de vibração que desaceleram a condução ([ALTMANN, 1991](#)).

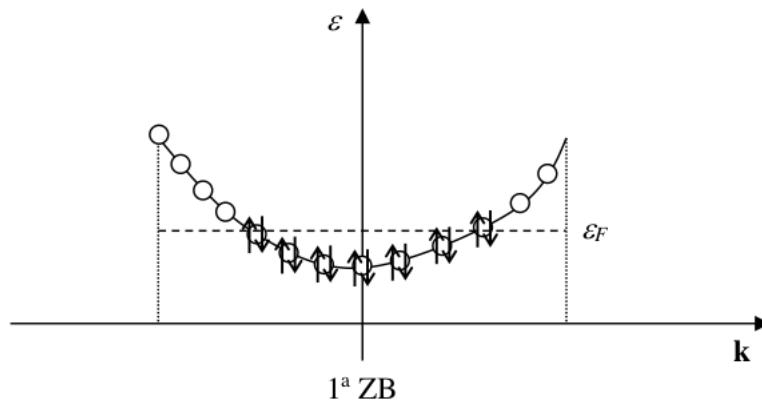


Figura 3: Representação da última banda de um cristal com um número ímpar de elétrons. Figura extraída da referência ([CAPAZ, 2019](#)).

Se o número de elétrons é par, duas situações podem ocorrer. Quando a última banda de valência está preenchida e a primeira banda de condução está desocupada, além de não haver superposição entre elas, o material é isolante. A região entre essas bandas não possuem estados permitidos e a diferença de energia associada é caracterizada como

gap de energia. Os semicondutores são, por definição, um caso particular dos isolantes e geralmente possuem um gap menor. Quando a temperatura aumenta, os elétrons tendem a ocupar a banda superior e a condução intensifica, em oposição ao comportamento dos metais. Aos semicondutores em que esse processo é natural chamamos de intrínsecos. Nos semicondutores extrínsecos é necessário acrescentar impurezas que criem estados na região proibida para que seja viável realizar transições para a banda de condução. A figura 4 exibe a ocupação dos elétrons na última banda de valência em sistemas isolantes.

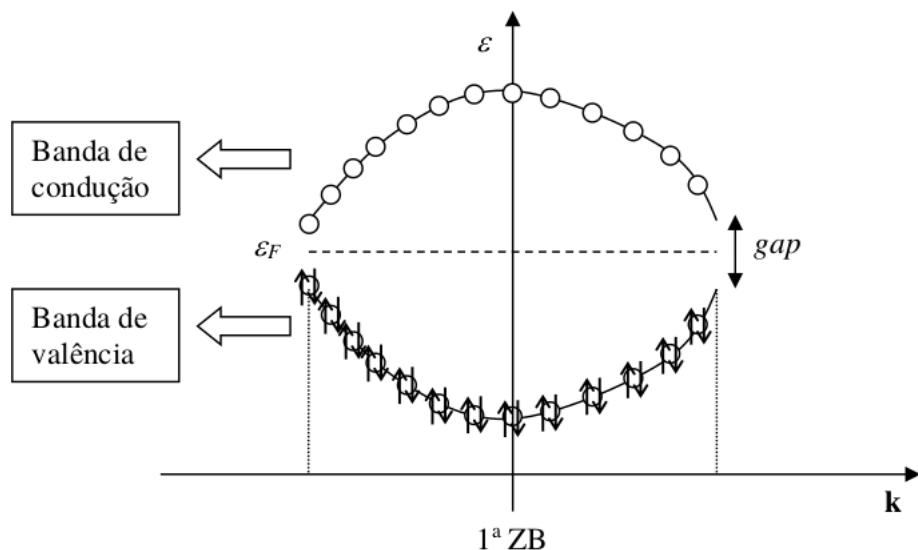


Figura 4: Representação da última banda de valência e a primeira banda de condução sem superposição de um cristal com um número ímpar de elétrons. Figura extraída da referência ([CAPAZ, 2019](#)).

No caso em que as bandas se superpõem, haverá duas bandas semipreenchidas, caracterizando novamente um comportamento metálico. Nesse caso, chamamos o material de semimetal e a figura 5 representa um exemplo de superposição.

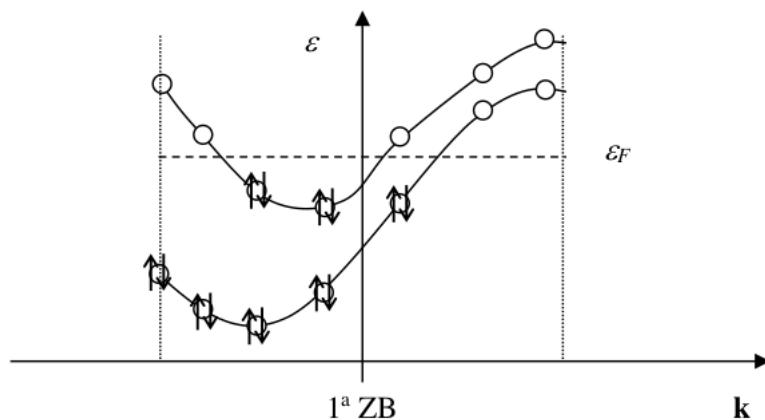


Figura 5: Representação da última banda de valência e a primeira banda de condução com superposição de um cristal com um número ímpar de elétrons. Figura extraída da referência ([CAPAZ, 2019](#)).

2.5 Funções de base

2.5.1 Teoria dos Pseudopotenciais

O emprego das ondas planas como uma base parece vantajoso: os elementos de matrizes são simples, onde o operador de energia cinética é diagonal e os elementos de matrizes que representam o potencial são somente a transformada de fourier do potencial cristalino ([GROSSO G.; PARRAVICINI, 2000](#)). Contudo, as funções de onda dos elétrons mais próximos ao núcleo são fortemente localizadas no espaço real, o que leva a utilização de um grande número de vetores recíprocos em sua expansão. Esses elétrons são chamados elétrons de caroço e variam pouco sua densidade eletrônica. Enquanto isso, os elétrons das câmadas mais externas - de valência - são mais energéticos e participam diretamente nas ligações químicas ([ALVES, 2011](#)). Visto que os elétrons de caroço eram bem descritos pelo método tight binding, Hering propôs, em 1940, que suas funções continuassem sendo especificadas por somas de Bloch construídas de orbitais atômicos ⁷ e que os elétrons de valência fossem caracterizados por ondas planas ortogonalizadas (Orthogonalized plane waves-OPW). Visto que essa idéia serviu de base para os pseudopotenciais, vamos analisá-la de um ponto de vista geral.

O princípio da ortogonalização

$$^7\psi_c(\mathbf{k}, \mathbf{r}) = \frac{1}{\sqrt{N}} \sum_{\mathbf{R}_n} e^{i(\mathbf{k} \cdot \mathbf{R}_n)} \Phi_c(\mathbf{r} - \mathbf{R}_n)$$

Sejam os estados ψ_c do caroço que satisfazem a equação de autovalores:

$$H\psi_c = \epsilon_c \psi_c. \quad (2.94)$$

Uma proposta para o estado de valência $|\psi\rangle$ é uma modificação de uma onda plana $|\phi\rangle$ que seja ortogonal a todos os estados $|\psi_c\rangle$:

$$|\psi\rangle = |\phi\rangle - \sum_c |\psi_c\rangle \langle \psi_c| \phi \rangle. \quad (2.95)$$

Como os estados do caroço são localizados, os estados OPW são adequados para os elétrons mais energéticos, pois a contribuição das funções ψ_c são pequenas e $|\psi\rangle$ se aproxima do comportamento de uma onda plana (MISRA, 2012). Entretanto, tais estados não são ortonormais e a matriz de sobreposição,

$$\tilde{S}_{ij} = \langle \psi_i | \psi_j \rangle = \delta_{ij} - \langle \phi | \left[\sum_c |\psi_c\rangle \langle \psi_c| \right] | \phi \rangle, \quad (2.96)$$

deve ser não singular de determinante positivo. Ao substituir a expressão (2.95) na equação de autovalores para os elétrons de valência:

$$H\psi = \epsilon\psi, \quad (2.97)$$

e utilizar a eq.(2.96), obtemos:

$$(H + V^c) |\phi\rangle = \epsilon |\phi\rangle, \quad (2.98)$$

em que,

$$V^c = \sum_c (\epsilon - \epsilon_c) |\psi_c\rangle \langle \psi_c|, \quad (2.99)$$

e que contribui para a matriz com os elementos $\langle \phi_i | V^c | \phi_j \rangle$. Em 1959, o trabalho publicado por Phillips e Kleinman indicaram que a convergência se torna mais rápida ao utilizar $|\phi\rangle$ como um conjunto de ondas planas simetrizadas, diferentemente de uma onda plana única proposta por Hering (PHILLIPS; KLEINMAN., 1959). Além disso, visto que $\epsilon - \epsilon_c$ é positivo, podemos tratar o operador V^c como um potencial repulsivo. Por consequência, conseguimos substituir o potencial cristalino forte por um pseudopotencial fraco capaz de reproduzir a energia dos estados de valência:

$$(T + V + V^c) |\phi\rangle = \epsilon |\phi\rangle \Rightarrow (H + V^{PS}) |\phi^{PS}\rangle = \epsilon |\phi^{PS}\rangle. \quad (2.100)$$

Na equação acima, ϵ é o exato autovalor da equação de Kohn-Sham, mas agora podemos obtê-lo de uma função suave ϕ^{PS} através da escolha de V^{PS} (VIANNA; CANUTO S.; FAZZIO, 2004). Os pseudopotenciais podem ser empíricos ou ab-ínitio. O primeiro envolve um conjunto de parâmetros ajustáveis que buscam reproduzir dados experimentais. O segundo é construído pela solução da equação de Kohn-Sham para o caso atômico. Os trabalhos iniciais mais relevantes para a construção de pseudos do segundo tipo foram feitos por Bachelet, Hamann e Schlüter (BACHELET G. B.; HAMMAN, 1982) e Troullier e Martins (TROULLIER N.; MARTINS, 1993) seguindo o procedimento proposto por Zunger e Cohen (ZUNGER A.; COHEN, 1978).

2.5.1.1 Pseudopotenciais de norma conservada

Os pseudopotenciais de norma conservada (NCP) se beneficiam primeiramente de um cálculo atômico auto-consistente *all electron* (AE) e as funções atômicas são soluções da equação:

$$\left[-\frac{1}{2} \frac{d^2}{dr^2} + \frac{l(l+1)}{2r^2} + V(\mathbf{r}) \right] rR_{nl} = \epsilon_{nl} rR_{nl}, \quad (2.101)$$

nos quais n e l são números quânticos principal e do momento angular, respectivamente. O potencial atômico $V(\mathbf{r})$ é dado por:

$$V(\mathbf{r}) = -\frac{Z}{r} + V_H + V_{xc}, \quad (2.102)$$

em que o primeiro termo da soma representa a interação colombiana elétron-núcleo, V_H é o potencial de Hartree e V_{xc} é o potencial de troca e correlação. Posteriormente, os pseudopotenciais de norma conservada são criados de acordo com os seguintes critérios de Hamann, Schlüter e Chiang (HAMANN; M.; CHIANG, 1979):

1. As pseudofunções de valência não devem conter nodos;
2. Os autovalores ϵ_l^{AE} para os estados de valência são iguais aos autovalores ϵ_l^{PS} obtidos do pseudopotencial;

$$\epsilon_l^{AE} = \epsilon_l^{PS}. \quad (2.103)$$

3. A pseudofunção de onda radial é igual a função de onda radial obtida do cálculo all

electron a partir de um determinado raio de corte r_c ;

$$R_l^{AE}(r) = R_l^{PS}(r) \quad r > r_c. \quad (2.104)$$

4. Para a pseudofunção e para a função all electron, a carga eletrônica na região limitada por r_c devem ser iguais (conservação de norma);

$$\int_0^{r_c} |R_l^{AE}(r)|^2 r^2 dr = \int_0^{r_c} |R_l^{PS}(r)|^2 r^2 dr. \quad (2.105)$$

5. A derivada logarítmica da pseudofunção e da função all electron devem ser iguais em $r = r_c$.

$$\frac{d}{dr} \ln(R_{nl}^{AE}) = \frac{d}{dr} \ln(R_{nl}^{PS}). \quad (2.106)$$

2.5.1.2 Pseudopotenciais Ultrasuaves

Apesar dos pseudopotenciais de norma conservada serem efetivos para descrever estados eletrônicos de diversas estruturas cristalinas, ainda é necessário um grande conjunto de ondas planas para orbitais de valência altamente localizados, tais como nos metais de transição. A condição de norma conservada impede, por exemplo, descrever pseudofunções mais suaves que a função all-electron de orbitais $2p$ do oxigênio e $3d$ do níquel. Este problema foi resolvido com os pseudopotenciais ultrasuaves desenvolvidos por Vanderbilt. Tal proposta não contém a conservação de norma e permite um raio de corte maior ([VANDERBILT, 1990](#)).

2.5.2 Projetor de ondas aumentadas

Ao empregar os pseudopotenciais, toda a informação sobre a função de onda perto dos núcleos é perdida, influenciando o cálculo de determinadas propriedades. Uma das abordagens para solucionar o problema é o método do projetor de ondas aumentadas (projector augmented wave method-PAW), proposto por Blöchl ([BLÖCHL, 1994](#)). Nessa perspectiva, constrói-se o operador linear \hat{T} o qual leva uma função de onda auxiliar suave $|\tilde{\psi}_n\rangle$ na verdadeira função de onda all-electron de Kohn-Sham $|\psi_n\rangle$ ([ROSTGAARD, 2009](#)):

$$|\psi_n\rangle = \hat{T} |\tilde{\psi}_n\rangle. \quad (2.107)$$

Essa transformação leva às novas equações de Kohn-Sham que devem ser resolvidas:

$$\hat{T}^\dagger \hat{H} \hat{T} |\tilde{\psi}_n\rangle = \epsilon \hat{T}^\dagger \hat{T} |\tilde{\psi}_n\rangle. \quad (2.108)$$

Nesse momento, é necessário obter as propriedades do operador \hat{T} de modo que as funções $\tilde{\psi}_n$ sejam suaves. Dado que a certa distância do caroço as funções já possuem essa característica, construimos o operador de transformação de forma que ele atue somente dentro de uma determinada esfera centrada em cada átomo a e que não se sobreponem:

$$\hat{T} = \hat{1} + \sum_a \hat{T}^a. \quad (2.109)$$

no qual \hat{T}^a não atua fora do raio de corte r_c^a . No interior da esfera, podemos expandir a verdadeira função de onda em funções de onda parciais auxiliares $|\phi_i^a\rangle$ e cada uma delas está associada à uma função parcial suave $|\tilde{\phi}_i^a\rangle$ que obedece:

$$|\phi_i^a\rangle = (\hat{1} + \hat{T}^a) |\tilde{\phi}_i^a\rangle \Leftrightarrow \hat{T}^a |\tilde{\phi}_i^a\rangle = |\phi_i^a\rangle - |\tilde{\phi}_i^a\rangle. \quad (2.110)$$

Uma vez que T^a não atua fora da esfera, estabelecemos a igualdade:

$$\forall a, \phi_i^a(\mathbf{r}) = \tilde{\phi}_i^a(\mathbf{r}), r > r_c^a. \quad (2.111)$$

Se as funções de onda parciais formam uma base, podemos expandir a função de onda suave do caroço como:

$$|\tilde{\psi}_n\rangle = \sum_i P_{ni}^a |\tilde{\phi}_i^a\rangle, \quad r < r_c^a. \quad (2.112)$$

e consequentemente,

$$|\psi_n\rangle = \hat{T} |\tilde{\psi}_n\rangle = \hat{T} \sum_i P_{ni}^a |\tilde{\phi}_i^a\rangle = \sum_i P_{ni}^a |\phi_i^a\rangle, \quad r < r_c^a, \quad (2.113)$$

em que os coeficientes devem ser dados por uma projeção em cada esfera:

$$P_{ni}^a = \langle \tilde{p}_i^a | \tilde{\psi} \rangle. \quad (2.114)$$

Blöchl chamou $\langle \tilde{p}_i^a |$ de funções projetoras e estabeleceu que elas devem satisfazer a relação de completeza:

$$\sum_i |\tilde{\phi}_i^a\rangle \langle \tilde{p}_i^a| = 1 \Rightarrow \langle \tilde{p}_i^a | \tilde{\phi}_j^a \rangle = \delta_{ij}. \quad (2.115)$$

Usando esta relação, temos:

$$\begin{aligned} \hat{T}^a &= \sum_i \hat{T}^a |\tilde{\phi}_i^a\rangle \langle \tilde{p}_i^a| = \sum_i \left(|\phi_i^a\rangle - |\tilde{\phi}_i^a\rangle \right) \langle \tilde{p}_i^a| \Rightarrow \\ &\Rightarrow \hat{T} = \hat{1} + \sum_a \sum_i \left(|\phi_i^a\rangle - |\tilde{\phi}_i^a\rangle \right) \langle \tilde{p}_i^a|. \end{aligned} \quad (2.116)$$

Como efeito, a função de onda all-electron de Kohn-Sham é:

$$\psi_n(\mathbf{r}) = \tilde{\psi}_n(\mathbf{r}) + \sum_a \sum_i \left(\phi_i^a(\mathbf{r}) - \tilde{\phi}_i^a(\mathbf{r}) \right) \langle \tilde{p}_i^a | \tilde{\psi}_n \rangle. \quad (2.117)$$

À vista disso, as funções de KS são decompostas em funções auxiliares suaves por todo o espaço e funções oscilantes próximas ao núcleo. A decomposição permite tratá-las separadamente.

2.6 Quantum Espresso - QE

Para investigar as propriedades estruturais e eletrônicas dos semicondutores de gap ultra largo foi utilizado o programa Quantum Espresso ([GIANNOZZI, 2009](#)). Esse se baseia na Teoria do Funcional da Densidade, descrita neste capítulo 2, e constitui um conjunto de códigos computacionais para descrever cálculos de estrutura eletrônica e modelagem de materiais em nanoscalas. O QE é desenvolvido e mantido pelo DEMOCRITOS *Modeling Center for Research in Atomistic Simulation*, que pertence a *Scuola Internazionale Superiore di Studi Avanzati* (SISSA), em Trieste ([ALVES, 2011](#)). Ademais, é um software livre que descreve a interação elétron-íon através dos pseudopotenciais de norma conservada, ultrasoft ou ondas planas aumentadas (PAW). Além disso, disponibiliza diversos funcionais de troca e correlação: LDA, GGA (PW91, PBE, B88-P86, BLYP), meta-GGA e funcionais hidrídicos (PBE0, B3LYP, HSE). O Quantum Espresso pode efetuar, dentre outras possibilidades, as seguintes tarefas:

- Cálculo das energias totais autoconsistentes, forças, stress e orbitais de Kohn-Sham;
- Magnetismo não colinear e acoplamento spin-órbita;
- Dinâmica Molecular de Car-Parrinello e Dinâmica Molecular de Born-Oppenheimer;

- Dispersão de fônons e acoplamento elétron-fônon;
- Excitações eletrônicas e propriedades de transporte.

3 Aprendizado de Máquina

Desde sua origem, a ciência tem lidado com diferentes formas de investigar os fenômenos naturais. Em seu primeiro paradigma, tais investigações eram baseadas de forma empírica, como por exemplo, o uso do fogo. Posteriormente, com o desenvolvimento da mecânica clássica, passamos a ter uma ciência *concept-driven*, cujas teorias não apenas explicam os fenômenos anteriores como apontam para novos que ainda devem ser observados. A medida em que os modelos ficam mais complexos, a solução analítica se torna inviável. Na ciência de materiais isso é recorrente, já que lida-se com sistemas de muitos corpos interagentes. Com o desenvolvimento tecnológico crescente de hardware e da Teoria do Funcional da Densidade, essa necessidade foi suprida através de simulações computacionais, nos levando ao terceiro paradigma. Nos dias atuais, a busca por formas de tratar, analisar e visualizar a grande quantidade de dados gerada pelas simulações e experimentos dá origem ao quarto paradigma, denominado *tool-driven*. Para tais quantidades de dados, essas tarefas não são simples e necessitam de ferramentas apropriadas. Dentre elas, o aprendizado de máquina, no qual é possível tratar, descobrir padrões e fazer previsões ([SCHLEDER, 2021](#)).

A abordagem tradicional na descoberta de novos materiais para aplicações específicas consiste em tentativa e erro, simulando casos particulares ou guiados por similaridade de casos já conhecidos. Com o uso da inteligência artificial é possível treinar modelos (utilizando as bases de dados disponíveis), no intuito de prever materiais com as propriedades desejadas, diminuindo assim o custo computacional e o direcionando para materiais em potencial. No entanto, a criação de dados é feita de forma mais acelerada que a produção de informação, conhecimento e seu uso em tecnologias. Sendo assim, um rápido avanço nas técnicas de análise se faz necessário para diminuir essa lacuna.

As etapas de um projeto de aprendizado de máquina consistem na definição do problema, aquisição de dados, representação e otimização dos algoritmos de aprendizagem. Cada dado de entrada (instância) é constituído por um conjunto de características (*features*) que estão relacionadas com a propriedade alvo, podendo assumir valores reais

(*target*) ou categóricos (*label*). Se o conjuntos desses dados inclui a propriedade alvo no treinamento do modelo, temos um aprendizado supervisionado, e que pode ser aplicado em problemas de classificação ou regressão. No aprendizado não supervisionado a propriedade alvo não é incluída, e os algoritmos podem ser usados em problemas de clusterização, detecção de anomalia e redução de dimensionalidade, por exemplo. Ao utilizar um conjunto de dados em que apenas algumas instâncias incluem *target/label*, o aprendizado é chamado semissupervisionado. No aprendizado por reforço, através de um sistemas de recompensas e penalidades sobre ações passadas, um agente infere as melhores decisões a serem tomadas em um ambiente. Na aquisição de dados, algumas bases de dados são de livre compartilhamento, por exemplo, AFLOWlib ([CURTAROLO, 2012](#)), Materials Project ([JAIN, 2013](#)), NOMAD ([DRAXL C.; SCHEFFLER, 2019](#)) e C2DB ([HAASTRUP, 2018](#)). As *features* utilizadas na representação de compostos são nomeadas descritores e devem conter a informação necessária para predizer a propriedade de interesse. Para uma quantidade de *features* significativa, diversos modelos sofrem com a maldição da dimensionalidade. Nesse caso, a distância entre as instâncias aumenta e um número exponencial delas deve ser inserido para obter a mesma performance. Sendo assim, desvendar um número mínimo de descritores que forneçam um bom desempenho é uma tarefa importante e que depende do conhecimento do profissional.

Na medida em que a representação é concluída, pode-se utilizá-la na aprendizagem dos modelos. Muitos deles são considerados caixas pretas (*black boxes*) e quanto mais complexos eles são, menor é sua interpretabilidade. É comum a utilização de modelos como regressão logística, *support vector machine*(SVC), *k-nearest neighbors*(kNN), polinômios e árvore de decisão. Da mesma forma, é frequente o uso de métodos ensemble, o qual utiliza diversos modelos em conjunto para construir um melhor preditor. A seguir, há uma introdução ao modelo de árvore de decisão e 3 ensembles que o utilizam como base.

3.1 Árvore de Decisão

A predição feita por uma árvore de decisão se baseia em percorrer o nó-raiz até os nós-folhas. Em cada passo, o algoritmo divide o conjunto de treinamento operando com uma *feature* k e um limite t_k ([GÉRON, 2019](#)). Para problemas de classificação binária, cada divisão no espaço de entrada é baseado na minimização da função custo a seguir:

$$J(k, t_k) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right}. \quad (3.1)$$

em que $G_{left/right}$ representam a impureza de $Gini$ nos nós à esquerda e direita, respectivamente, e $m_{left/right}$ é o número de instâncias nessas direções. De forma geral, o coeficiente de $Gini$ está relacionado à separabilidade dos dados ao escolher determinada *feature*, e é dado por:

$$G_i = 1 - \sum_{k=1}^n p_{i,k}^2, \quad (3.2)$$

no qual $p_{i,k}$ é a frequência relativa de instâncias da classe k no i-ésimo nó. Para os problemas de regressão deve ser minimizada a seguinte função:

$$J(k, t_k) = \frac{m_{left}}{m} MSE_{left} + \frac{m_{right}}{m} MSE_{right}, \quad (3.3)$$

escrita em termos do erro quadrático médio(MSE) para os nós à esquerda e a direita, e é descrito por:

$$MSE = \sum_{i=1}^m \frac{1}{m} (\hat{y} - y^{(i)})^2, \quad (3.4)$$

considerando \hat{y} como o rótulo e $y^{(i)}$ o valor predito. Nessa situação, uma vez que empreguem-se valores reais e não classes, o número predito é a média dos valores no correspondente nó.

Embora as árvores de decisão sejam fáceis de interpretar e escaláveis, seu desempenho é sensível a rotação e pequenas variações nos dados. Uma forma de diminuir essa instabilidade é considerar a média ou o voto majoritário das previsões de várias árvores, o que nos leva ao modelo de Florestas Aleatórias.

3.2 Floresta Aleatória

A Floresta aleatória geralmente é um método *bagging*, ou seja, para cada árvore treinada é admitido um conjunto de dados com o mesmo tamanho do conjunto original e as instâncias antecessoras são escolhidas aleatoriamente, podendo se repetir. Em cada passo, apenas um grupo u' dentre as u variáveis(*features*) é escolhido para ocupar os nós. Além disso, a ordem dos descritores depende da minimização da impureza *gini* ou do erro quadrático médio, como visto anteriormente. A predição é feita pelo voto majoritário nos problemas de classificação, ou pela média das previsões de cada árvore nos problemas de regressão.

Diferentemente dos métodos *bagging*, *boosting* são modelos aplicados em sequência,

em que o próximo busca corrigir as falhas na predição do modelo anterior. Formas populares de fazer isso empregando árvores de decisão são os preditores *Adaboost* e *Gradiente Boosting*.

3.3 AdaBoost

No algoritmo *AdaBoost*, após o primeiro treinamento são atribuídos pesos às instâncias que foram erroneamente classificadas e o segundo classificador atua levando em conta esses pesos. Inicialmente, cada instância possui o peso $1/m$, em que m é o número total de instâncias. Posteriormente, atribui-se uma taxa de erro ponderada ao preditor j ,

$$r_j = \frac{\sum_{\substack{i=1 \\ \hat{y}_j^{(i)} \neq y^i}}^m w^{(i)}}{\sum_{i=1}^m w^{(i)}}, \quad (3.5)$$

no qual $\hat{y}_j^{(i)}$ é a predição do classificador j para a instância i . Em seguida, calcula-se o peso do preditor α_j ,

$$\alpha_j = \eta \log \frac{1 - r_j}{r_j}, \quad (3.6)$$

em que η é a taxa de aprendizagem de cada árvore e o peso aumenta a medida que o preditor é mais acurado. Agora pode-se calcular os novos pesos das instâncias,

$$\begin{cases} w^i, & \hat{y}_j^{(i)} = y^i \\ w^i e^{(\alpha_j)}, & \hat{y}_j^{(i)} \neq y^i, \end{cases}$$

e então, normalizá-los. A classe predita pelo modelo Adaboost é a classe escolhida pelo conjunto de preditores com a maior soma de pesos, ou seja,

$$\hat{y}(\mathbf{x}) = \arg \max_k \sum_{\substack{j=1 \\ \hat{y}_j(\mathbf{x})=k}}^N \alpha_j. \quad (3.7)$$

3.4 Gradient Boosting

O algoritmo do Gradient Boosting trabalha de forma similar ao Adaboost, uma vez que emprega ávores de decisão em sequência tentando corrigir os erros dos preditores anteriores. No entanto, ao invés de serem atribuídos pesos às instâncias mal classificadas, o novo preditor treina os erros residuais do seu antecessor, ou seja, concentra-se na diferença entre os valores preditos e os valores tabelados. Na regressão, o primeiro resíduo é calculado em relação à média dos valores alvo. Posteriormente, a árvore de decisão prediz os novos resíduos. Após o treinamento de todas as árvores, o valor final corresponde à soma da média com os sucessivos resíduos, escalados por uma taxa de aprendizado idêntica para cada árvore. Nos algoritmos de classificação é possível utilizar resíduos se trabalharmos com a probabilidade de determinada classe k ocorrer, definida pela função logística ([HASTIE; TIBSHIRANI; FRIEDMAN, 2017](#)):

$$p_k = \frac{e^{f_k(x)}}{\sum_{l=1}^K e^{f_l(x)}}, \quad (3.8)$$

em que,

$$f_k(x) = \log p_k(x) - \frac{1}{K} \sum_{l=1}^K \log p_l(x). \quad (3.9)$$

Após computar os resíduos,

$$r_{ikm} = y_{ik} - p_k(x_i), \quad i = 1, 2, 3, \dots, N \quad (3.10)$$

cada folha j pode conter um conjunto deles, R_{jkm} , no qual $j = 1, 2, 3, \dots, J_m$. Em seguida, nesse conjunto, realiza-se a subsequente transformação,

$$\gamma_{jkm} = \frac{K-1}{K} \frac{\sum_{x_i \in R_{jkm}} r_{ikm}}{\sum_{x_i \in R_{jkm}} |r_{ikm}|(1 - |r_{ikm}|)}. \quad (3.11)$$

Dessa forma, pode-se atualizar o valor predito:

$$f_{km}(x) = f_{k,m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jkm} \quad (3.12)$$

Então, após sucessivos treinamentos com M árvores, a classe é predita pela eq.(3.8) com f_{kM} . Geralmente considera-se $p = 0.5$ como o limiar para determinar se a instância

pertence a classe k . No entanto, esse valor pode ser mudado para garantir melhores resultados.

4 Resultados e Discussões

O projeto de pesquisa consistiu em quatro etapas. Na primeira, foram construídos os *datasets*, ou seja, tabelas manipuláveis com os dados necessários para as próximas etapas. Posteriormente foram treinados modelos de classificação e regressão. Em seguida, a previsão de novos semicondutores de gap ultra largo. Os algoritmos utilizados nessas fases do trabalho estão disponíveis no apêndice A. Por último, é realizado o cálculo da estrutura de bandas para um material UWBG utilizando DFT. Nas próximas sessões será detalhado o processo e os resultados de cada passo.

4.1 Construção do *dataset*

O conjunto de dados foi selecionado da base de dados de cristais bidimensionais C2DB, versão 2018-2. Em comparação com as bases de dados de *bulks*, há menor quantidade de bases com materiais 2D e menos desenvolvidas. A base C2DB é elaborada apenas para sistemas bidimensionais (monocamadas isoladas), calculadas principalmente pela Teoria do Funcional da Densidade (DFT), e possui aproximadamente 1500 sistemas distribuídos em mais de 30 diferentes estruturas cristalinas, obtidos pelas diferentes bases atômicas combinadas em protótipos distintos. Essas combinações totalizam 3712 materiais. Além disso, oferece uma variedade de propriedades: estruturais, termodinâmicas, elásticas, eletrônicas, magnéticas e ópticas. Seu acesso é livre e pode ser inteiramente baixada ou acessada de forma online ([HAASTRUP, 2018](#)). Embora a quantidade desses dados seja insuficiente para uma boa performance de modelos de aprendizado de máquina em outras áreas do conhecimento, na ciência de materiais isso é inicialmente frequente e abordagens tem sido estudadas para amenizar o problema enquanto as bases de dados se desenvolvem ([STOLL; BENNER, 2021](#)). Uma forma, apresentada por Zhang et. al., é incorporar uma estimativa rude da propriedade no espaço de features e assim, aumentar a acurácia do modelo de aprendizagem sem aumentar muito os graus de liberdade no espaço de features. Uma outra abordagem é restringir o espaço configuracional de materiais, por exemplo,

predizendo gap de famílias específicas de semicondutores, com determinadas estequiometrias ou estruturas cristalinas (ZHANG; LING, 2018). No entanto, isso também restringe o poder preditivo do algoritmo. Neste trabalho, escolheu-se empregar todas as estequiometrias disponíveis na base de dados, dentre elas, A, AB, AB_2 , ABC, ABC_2 e ABC_4 (fig.6). Além disso, para evitar aumentar a complexidade no conjunto de dados, optou-se por utilizar sistemas não magnéticos, conduzindo ao total de 2685 materiais. Uma vez que a base de dados possui mais que o triplo de metais em relação a quantidade de isolantes (fig.6) e o interesse se encontra no segundo grupo, determinou-se primeiro realizar a tarefa de classificação e posteriormente, regressão.

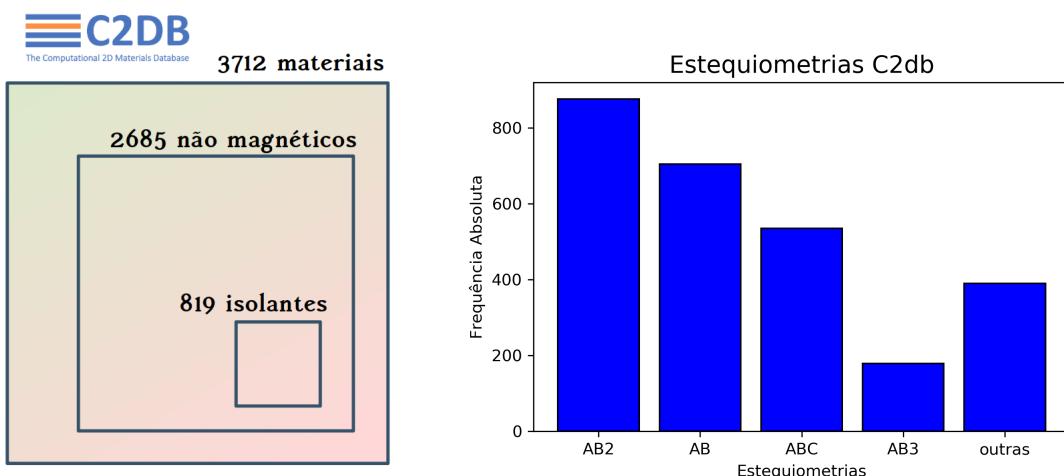


Figura 6: À esquerda, distribuição dos materiais no banco de dados C2DB. À direita, diagrama de barras das estequiometrias disponíveis.

Desde que a base da dados C2DB é constituída por apenas simulações computacionais, não utilizou-se dados experimentais. Ao lidar apenas com dados experimentais, para a maioria das estruturas cristalinas, o número de sistema estáveis conhecidos é pequeno e isso diminui o poder preditivo de novas estruturas (SCHMIDT et al., 2019).

Em relação à representação, aplicou-se descritores elementares (propriedades relacionadas aos elementos constituintes). Tal representação obedece os requisitos necessários, é completa, descritiva, simples e única. Não há uma representação elementar universal para os problemas de classificação e regressão. No entanto, utilizar propriedades atômicas como o número atômico, eletronegatividade, etc. tem se mostrado uma boa forma de prever o gap. Inicialmente, construímos um dataset exclusivamente para a estequiometria AB_2 , totalizando 506 metais e 251 isolantes. A razão desta etapa inicial é a simplicidade da sua construção, permitindo uma generalização mais comprehensível para as demais estequiometrias. As *features* aplicadas foram as características atômicas descritas na tabela 3, e essas foram retiradas do material suplementar do artigo de Schleider *et. al.* (SCHLEDER;

ACOSTA; FAZZIO, 2020).

Propriedade	Descrição
Z	número atômico
X	eletronegatividade de Pauling
G	grupo na tabela periódica
v	valência
E, I	afinidade eletrônica e potencial de ionização
ϵ^{ho} , ϵ^{lu}	autovalores de Kohn-Sham mais alto e mais baixo ocupado
α	polarizabilidade
r	raio atômico não ligado
r_v	raio do último orbital de valência ocupado
r_s , r_p , r_d	raios em que densidade de probabilidade radial dos orbitais de valência s, p e d são máximas
r_c	raio covalente
C	coluna da tabela periódica
n_o	número de orbitais desocupados

Tabela 3: Features atômicas.

Para evitar que ocorra a mesma representação em materiais distintos com mesma estequiometria, foram acrescentados os protótipos, também disponíveis na base de dados C2DB 2018-12. Cada protótipo carrega informações da simetria do material e é rotulado por um sistema representativo que é termodinamicamente estável nessa estrutura. Por exemplo, a fase H do dissulfeto de molibdênio (MoS_2) é um dos protótipos disponíveis. Então, esse mesmo material nessa fase pode ser escrito como $\text{MoS}_2-\text{MoS}_2$, enquanto em sua fase T pode ser escrito como $\text{MoS}_2-\text{CdI}_2$. Isso ocorre pois a fase T é termodinamicamente estável para o CdI_2 , outro protótipo disponível no banco de dados (HAASTRUP, 2018).

O acesso ao banco de dados C2DB é feito através da biblioteca *Atomic Simulation Environment* (ASE), escrita na linguagem de programação Python (LARSEN et al., 2017). Com a disponibilidade da fórmula química dos sistemas, o reconhecimento de quais átomos estão presente e a quantidade de cada espécie atômica é feito por meio da biblioteca *Python Materials Genomics* (pymatgen) (ONG et al., 2013). Desta forma, é possível a construção do *dataset* parcialmente ilustrado na figura 7.

Material	Band gap	prototype	Z_x	...	PeriodicColumn_y	PeriodicColumn_upto18_y	NumberUnfilledOrbitals_y	Polarizability_y	
0	FeO2	0.000000	GeS2	26	...	6.0	16.0	2.0	5.24
1	NiO2	1.281060	CdI2	28	...	6.0	16.0	2.0	5.24
2	GeO2	3.006473	GeS2	32	...	6.0	16.0	2.0	5.24
3	GeO2	3.641316	CdI2	32	...	6.0	16.0	2.0	5.24
4	GeO2	1.456923	MoS2	32	...	6.0	16.0	2.0	5.24
5	TiO2	3.480544	GeS2	22	...	6.0	16.0	2.0	5.24
6	TiO2	2.700829	CdI2	22	...	6.0	16.0	2.0	5.24
7	TiO2	1.195454	MoS2	22	...	6.0	16.0	2.0	5.24
8	Ti2O4	2.854873	WTe2	22	...	6.0	16.0	2.0	5.24
9	CoO2	0.000000	GeS2	27	...	6.0	16.0	2.0	5.24

Figura 7: Conjunto de dados com estequiometria AB_2 e *features* atômicas.

Para viabilizar a utilização de propriedades atômicas com diferentes estequiometrias seria necessário a construção de diferentes *datasets*, afinal um único conjunto de dados possuiria instâncias de diferentes dimensões. Isso ocorre devido ao fato dos compostos disporem de diferentes números de espécies atômicas, n_s . Sendo assim, para cada material com um número N de átomos na célula unitária, foram consideradas seis funções estatísticas para cada propriedade atômica. Por exemplo, para o MoS₂ e a propriedade eletronegatividade (X) tem-se entre suas funções estatísticas a média:

$$\bar{X}_{\text{MoS}_2} = \frac{X_{\text{Mo}} + X_S}{2} = \frac{2,16 + 2,58}{2} = 2,37. \quad (4.1)$$

Essas funções foram as mesmas utilizadas no artigo de Schleider *et. al.* ([SCHLEDER; ACOSTA; FAZZIO, 2020](#)), e encontram-se detalhadas na tabela 4. A figura 8 apresenta parte do conjunto de dados em que se emprega essas relações.

Feature	Descrição	Fórmula
$\bar{\gamma}$	Valor médio	$\bar{\gamma} = \sum_{i=1}^{n_s} \gamma_i / n_s$
$\tilde{\gamma}$	média ponderada pelo número de cada tipo de átomo	$\tilde{\gamma} = \sum_{i=1}^{n_s} \gamma_i n_i / N$
γ_M	valor máximo	$\gamma_M = \text{Max}(\gamma_i)$
γ_m	valor mínimo	$\gamma_m = \text{Min}(\gamma_i)$
$\bar{\gamma}_\sigma$	desvio padrão com relação à média	$\bar{\gamma}_\sigma = \sqrt{\sum_{i=1}^{n_s} (\bar{\gamma} - \gamma_i)^2 / n_s}$
$\tilde{\gamma}_\sigma$	desvio padrão com relação à média ponderada	$\tilde{\gamma}_\sigma = \sqrt{\sum_{i=1}^{n_s} (\tilde{\gamma} - \gamma_i)^2 / n_s}$

Tabela 4: Features estatísticas.

	Material	prototype	Band gap	media_Z	...	max_Polarizability	min_Polarizability	desvio_Polarizability	desvio_pon_Polarizability
0	Ag2O2	AuSe	0.164045	27.5	...	52.50	5.24	23.630	23.630
1	Ag2S2	AuSe	0.961595	31.5	...	52.50	19.37	16.565	16.565
2	Ag2Se2	AuSe	0.750715	40.5	...	52.50	26.24	13.130	13.130
3	Ag2Te2	AuSe	0.478298	49.5	...	52.50	37.00	7.750	7.750
4	Au2O2	AuSe	0.175493	43.5	...	36.10	5.24	15.430	15.430
5	Au2S2	AuSe	1.217789	47.5	...	36.10	19.37	8.365	8.365
6	Au2Se2	AuSe	0.956757	56.5	...	36.10	26.24	4.930	4.930
7	Au2Te2	AuSe	0.622154	65.5	...	37.00	36.10	0.450	0.450
8	Co2S2	NiSe	0.000000	21.5	...	53.00	19.37	16.815	16.815
9	Cu2O2	AuSe	0.000000	18.5	...	53.44	5.24	24.100	24.100

Figura 8: Conjunto de dados com todas as estequiometris e *features* estatísticas.

4.2 Classificação

O algoritmo de classificação foi preparado para categorizar os materiais em metais e isolantes. Os treinamentos foram feitos utilizando principalmente a biblioteca *sklearn* do python ([PEDREGOSA et al., 2011](#)). Antes do treinamento são atribuídos números inteiros distintos para cada protótipo, e definido gap igual a -1 para os metais e gap igual a 1 para os isolantes. Os *datasets* foram divididos em conjuntos de treino e teste com proporção 80/20 e de forma estratificada de acordo com as classes. Ou seja, 80% do dataset foi utilizado para o treinamento dos algoritmos e 20% para estimar o erro de generalização. A estratificação é importante pois as classes (metais e isolantes) são desbalanceadas. Os algoritmos *Gradient Boosting*, AdaBoost, *Random Forest* (Floresta Aleatória) e Árvore de decisão foram treinados ambos para o *dataset* com apenas estequiometria AB₂ quanto para o *dataset* geral. Para o maior conjunto de dados esse processo foi realizado com diferentes tipos de validação cruzada. Não houve diferença significativa na precisão entre as validações cruzadas empregadas, cuja dessemelhança é da ordem de 10⁻².

A métrica utilizada para prever o melhor modelo foi a curva de precisão-recall. Isso porque é ligeiramente mais importante que metais não sejam classificados como isolantes do que isolantes classificados erroneamente como metais. O modelo que obteve a melhor performance foi o *Gradiente Boosting*, como pode se ver na curva das figuras [9](#) e [10](#). Ele é considerado o melhor classificador por manter ambos precisão e recall altos ao longo de diversos limites de probabilidades. Uma forma mais quantitativa de observar esse fato é encontrar a área sob a curva precisão-recall. Diferentemente da regra trapezoidal, estimou-se a área através da precisão média (AP), a qual não utiliza interpolação ([METRICS..., c2007-2022](#)). Davis ([DAVIS; GOADRICH, 2006](#)) e Flach ([FLACH; KULL,](#)

2015) apresentaram que a interpolação linear de pontos na curva precisão-recall pode superestimar a performance do algoritmo. A precisão média é a média ponderada das precisões alcançadas em cada limiar, no qual o aumento do recall em relação ao limiar anterior é usado como peso:

$$AP = \sum_n (R_n - R_{n-1})P_n, \quad (4.2)$$

em que P_n e R_n são a precisão e o recall no n -ésimo limiar, respectivamente. Seu valor varia entre 0 e 1, e é melhor quanto mais próximo for do limite superior.

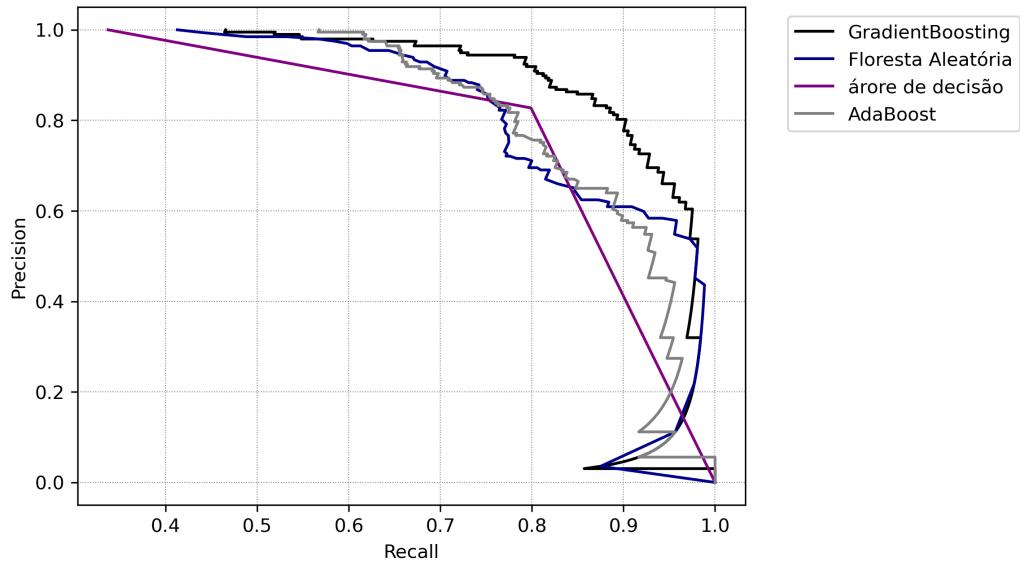


Figura 9: Curva precisão-recall para AB2 com *features* atômicas.

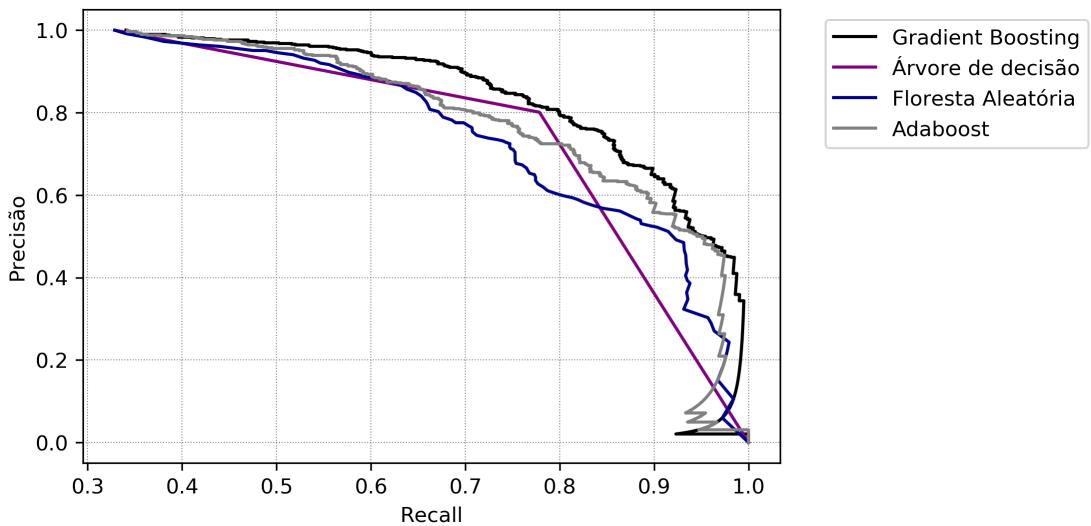


Figura 10: Curva precisão-recall para todas as estequiometrias com *features* estatísticas.

Escolhido o classificador, seus hiperparâmetros foram otimizados com *Randomized Search*. Nesse caso, escolhe-se de forma aleatória diferentes conjuntos de hiperparâmetros no grid proposto, treinando o modelo com validação cruzada. A área sob a curva precisão-recall no conjunto de teste para o *dataset* com estequiometria AB₂ é igual a 0,78 e para o *dataset* geral é 0,8. Embora o primeiro conjunto de dados possua dados mais específicos, sua quantidade de instâncias não é suficiente para ultrapassar a precisão média do maior *dataset*.

4.3 Regressão

Posteriormente, treinaram-se algoritmos de regressão para os isolantes da base de dados C2DB no intuito de aprender a prever o gap. Nesse caso, foram utilizados os valores de gap direto e indireto e que foram originados de simulações com o funcional de troca e correlação PBE. O histograma apresentado na figura 11 mostra a distribuição dos valores do gap em toda a base de dados.

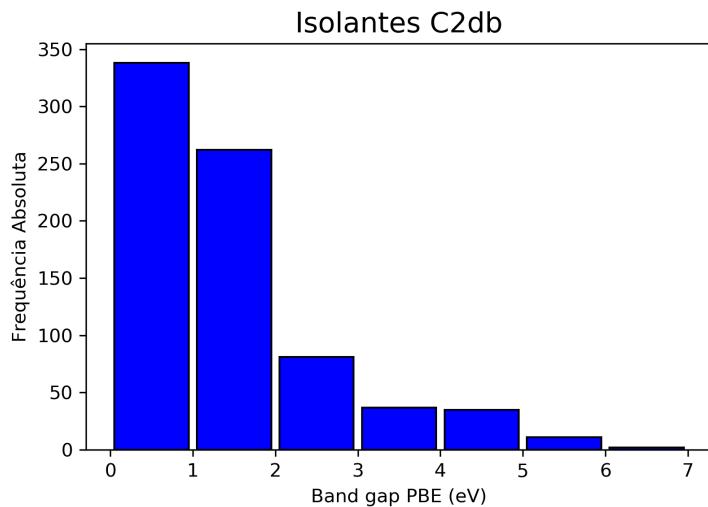


Figura 11: Histograma dos band gaps dos isolantes no banco de dados C2DB.

O conjunto de treino e teste foi particionado aleatoriamente em uma proporção 80/20 para o *dataset* geral e na proporção 90/10 para o *dataset* com estequiometria AB₂. A métrica utilizada para escolher o melhor modelo foi a raiz do erro quadrático médio (RMSE). A tabela 5 mostra o erro de cada algoritmo ao realizar a validação cruzada.

Algoritmo	RMSE (eV) - <i>datasetAB₂</i>	RMSE (eV) <i>dataset</i> geral
<i>Gradient Boosting</i>	0,32	0,31
Adaboost	0,45	0,47
Floresta Aleatória	0,36	0,40
Árvore de Decisão	0,75	0,62

Tabela 5: RMSE do gap no processo de treinamento.

O *Gradient Boosting* foi escolhido e otimizado com *Randomized Search*, apresentando um RMSE igual a 0,25 eV no conjunto de teste para o *dataset* geral e 0,12 eV para o *dataset* AB₂. Desde que o foco neste trabalho é em semicondutores de gap ultra largo, obter valores de gap que diferem de aproximadamente 0,25 eV é considerado um bom resultado. À esquerda da figura 12 ve-se o gráfico que apresenta o desempenho do modelo Gradiente Boosting na regressão para o maior conjunto de dados, e à direita, seu desempenho para o conjunto AB₂.

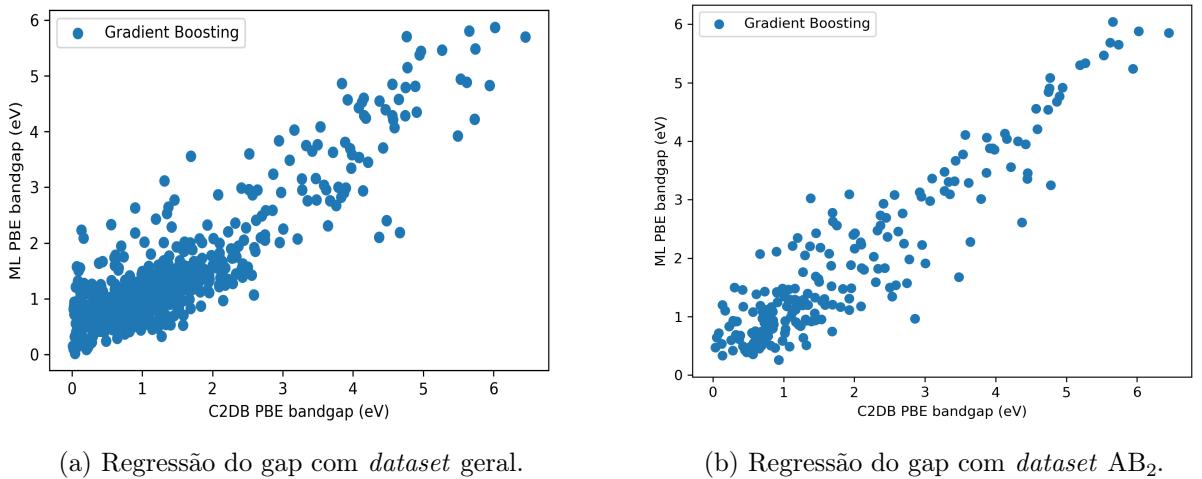


Figura 12: Regressão do gap.

4.4 Predição

Alguns dos novos materiais bidimensionais utilizados para prever o gap são os materiais propostos no artigo do Schleider *et al.* (SCHLEDER; ACOSTA; FAZZIO, 2020). Eles utilizaram técnicas de aprendizado de máquina para classificar os materiais do banco de dados C2DB como possuindo baixa, média e alta estabilidade termodinâmica. Eles geraram 100 combinações atômicas em 10 protótipos diferentes, formando um total de

1000 materiais. Os compostos com estequiometria ABC_2 são encontrados nos protótipos RhO, PbSe, GaSe, AuSe, NiSe e FeSe. Os átomos A são do grupo III da tabela periódica, o átomo B do grupo V e o átomo C pertence ao grupo VI. Enquanto isso, compostos com estequiometria ABC_4 foram propostos nos protótipos MoS₂, CdI₂, PdS₂ e ReS₂. Posteriormente, os autores do artigo realizaram uma seleção de materiais adequados para fotoeletrólise da água, encontrando 10 potenciais candidatos. Desses, o Sn₂SeTe e PbTe ainda não tinham sido reportados para essa aplicação.

Neste trabalho, como o objetivo é prever o valor do gap, utilizou-se 8 desses 10 protótipos, pois dois deles (RhO e NiSe) não possuem materiais isolantes associados na base de dados. A figura 13 apresenta cada protótipo em três diferentes direções. O átomo A representa um elemento da coluna III da tabela periódica. O átomo B indica um elemento da coluna V e os átomos rotulados por C representam em alguns casos o oxigênio e em outros, o enxofre.

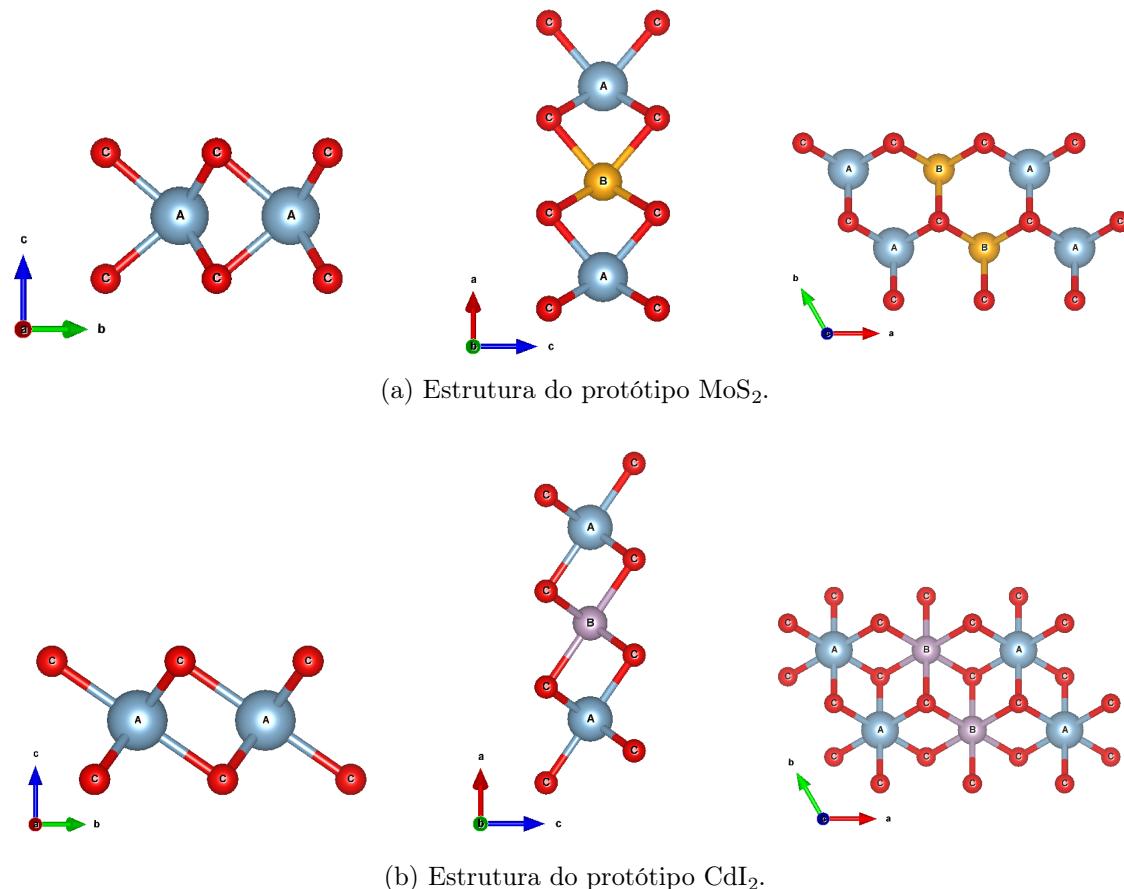


Figura 13: Estruturas associadas a cada protótipo.

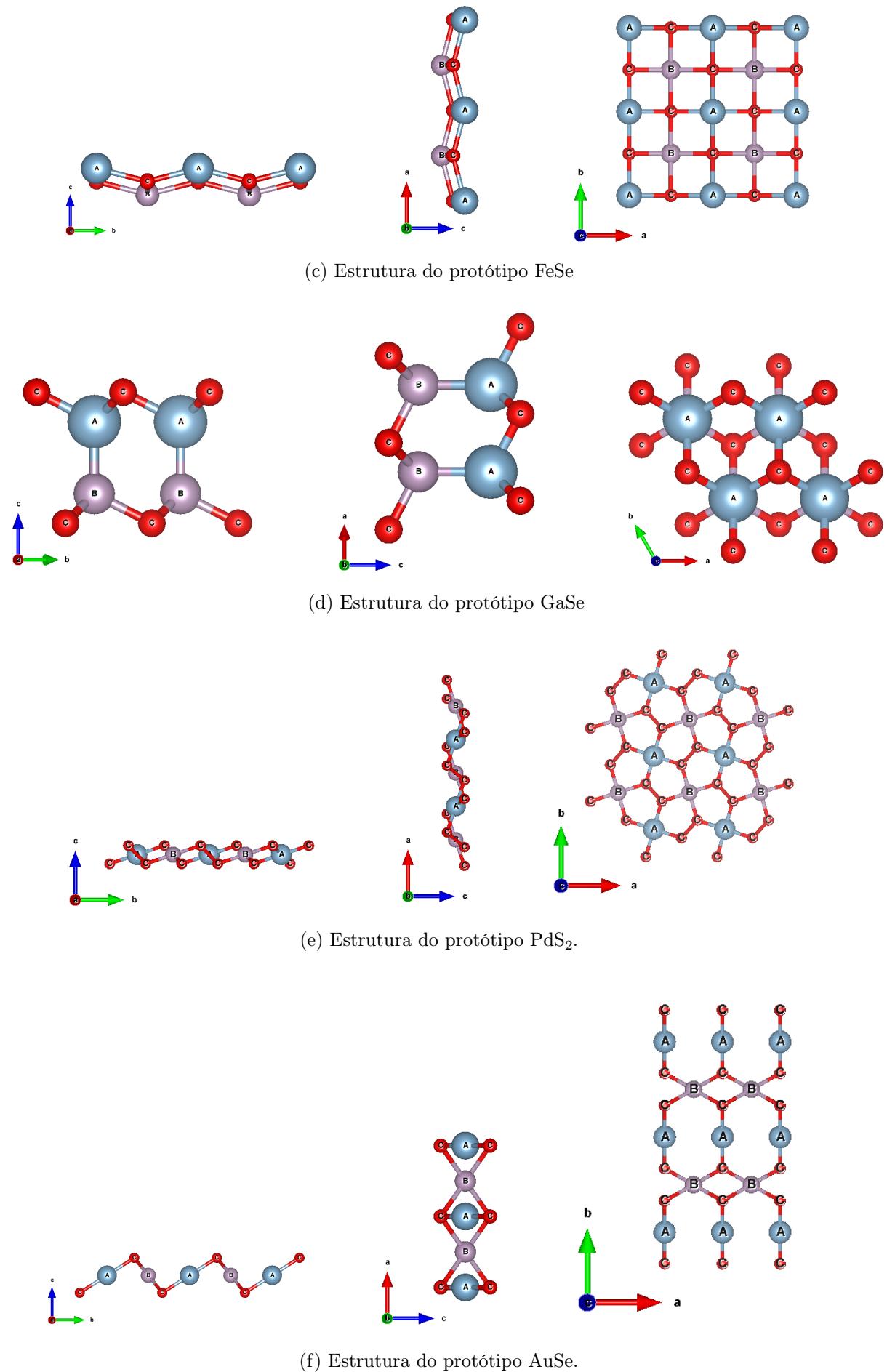


Figura 13: Estruturas associadas a cada protótipo.

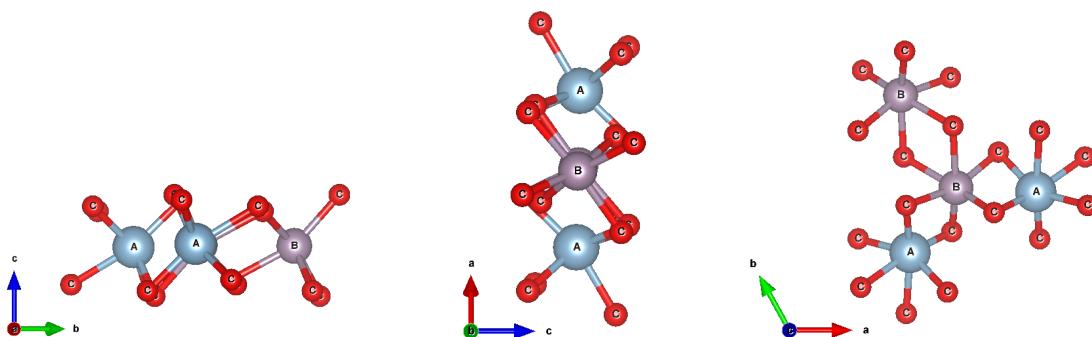
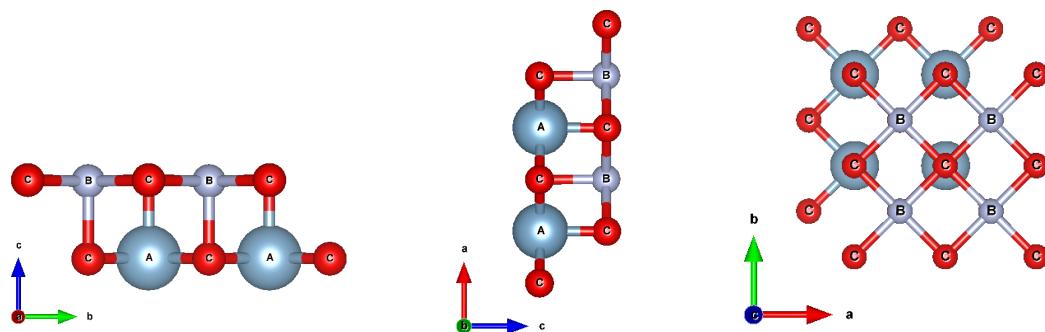
(g) Estrutura do protótipo ReS_2 .(h) Estrutura do protótipo PbSe .

Figura 13: Estruturas associadas a cada protótipo.

Com esses materiais e protótipos, construiu-se um novo *dataset* acrescentando as features estatísticas descritas na tabela 4. Então, o melhor modelo obtido no treinamento da classificação (*Gradient Boosting*) foi aplicado para categorizar os novos materiais em metais e isolantes. Dos 800 materiais propostos, 266 foram classificados como isolantes. Em sequência, o gap desses isolantes foram preditos pelo modelo de regressão (*Gradient Boosting*). Sabe-se que o funcional de troca e correlação PBE subestima o valor do gap. Para os semicondutores *bulk* de gap ultra largo, diferentemente dos gaps apresentados na tabela 2, o diamante calculado com esse funcional apresenta um gap igual a 4,12 eV (LIU et al., 2013), e o sistema $\beta\text{-Ga}_2\text{O}_3$ possui gap PBE igual a $\approx 2,35$ eV (ZEMAN et al., 2021). Enquanto isso, a liga AlGaN é constituída por AlN com gap PBE igual a 4,02 eV e GaN com gap igual a 1,69 eV (MOESA et al., 2011). Sendo assim, consideramos que os materiais previstos com gap maior que 2 eV são potencialmente UWBG e podem ser confirmados posteriormente com cálculos DFT utilizando potenciais híbridos, ou confirmados com cálculos GW. Considerando esse limite inferior, o algoritmo previu 134 materiais de gap ultra largo. Na figura abaixo encontram-se os diagramas de barra desses materiais com o valor do gap previsto em cada protótipo.

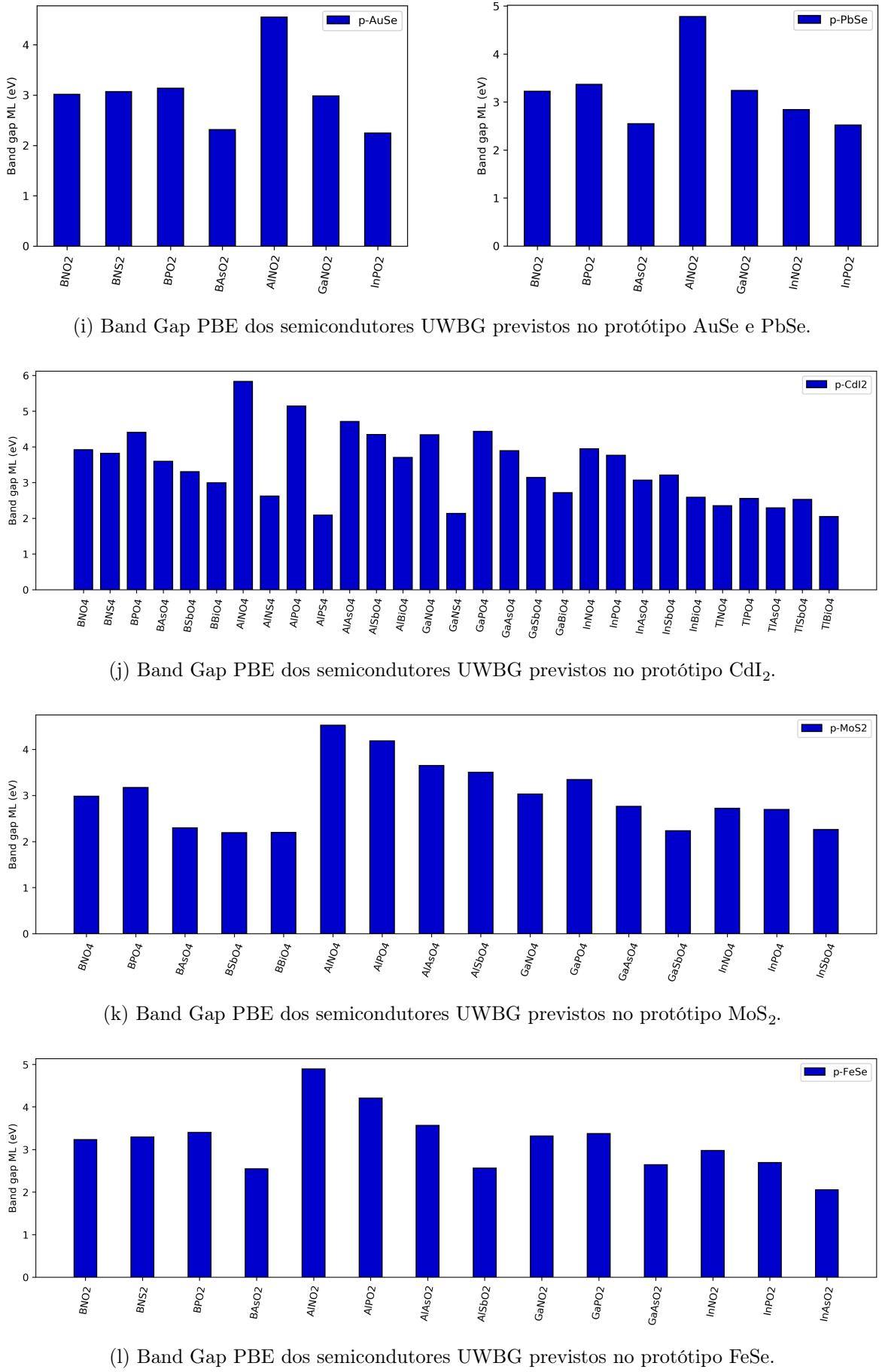
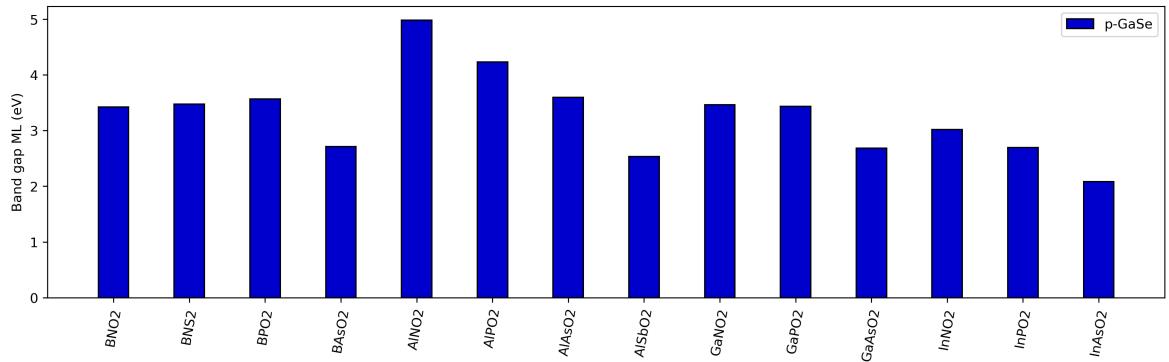


Figura 14: Diagrama de barras do gap PBE previsto nos materiais em cada protótipo.



(m) Band Gap PBE dos semicondutores UWBG previstos no protótipo GaSe.

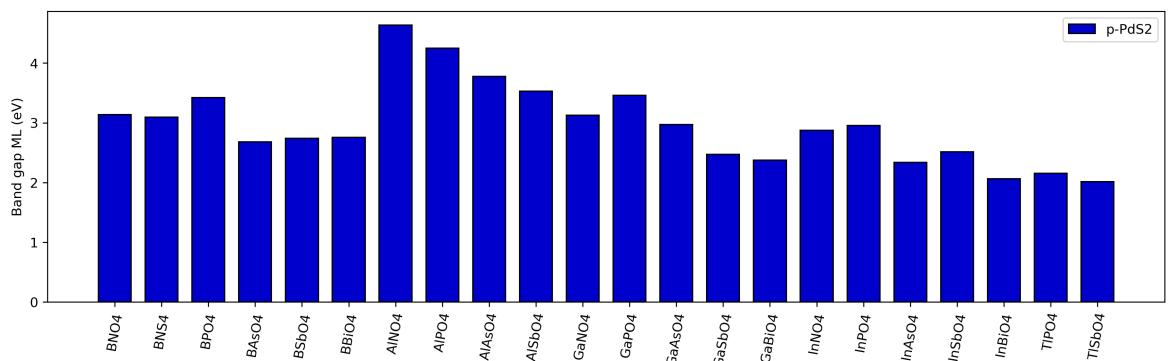
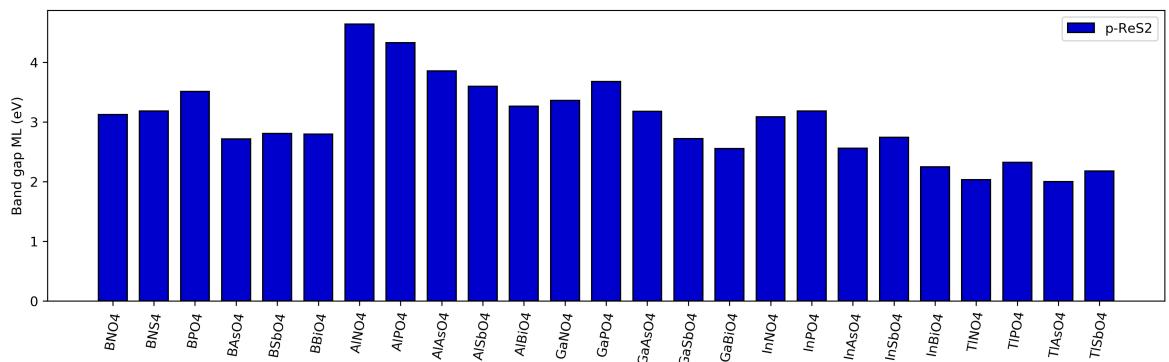
(n) Band Gap PBE dos semicondutores UWBG previstos no protótipo PdS₂.(o) Band Gap PBE dos semicondutores UWBG previstos no protótipo ReS₂.

Figura 14: Diagrama de barras do gap PBE previsto nos materiais em cada protótipo.

É possível observar através desses diagramas que os protótipos AuSe e PbSe possuem menos materiais UWBG que os demais. Além disso, independentemente da estrutura, a maioria deles são óxidos. A exceção ocorre quando a terceira espécie atômica é o enxofre e estão presentes nos protótipos AuS₂, PbSe, CdI₂, FeSe, GaSe, PdSe, ReSe. É também possível notar que em todas as estruturas o material com maior gap é aquele em que as duas primeiras espécies atômicas na fórmula química representam o AlN.

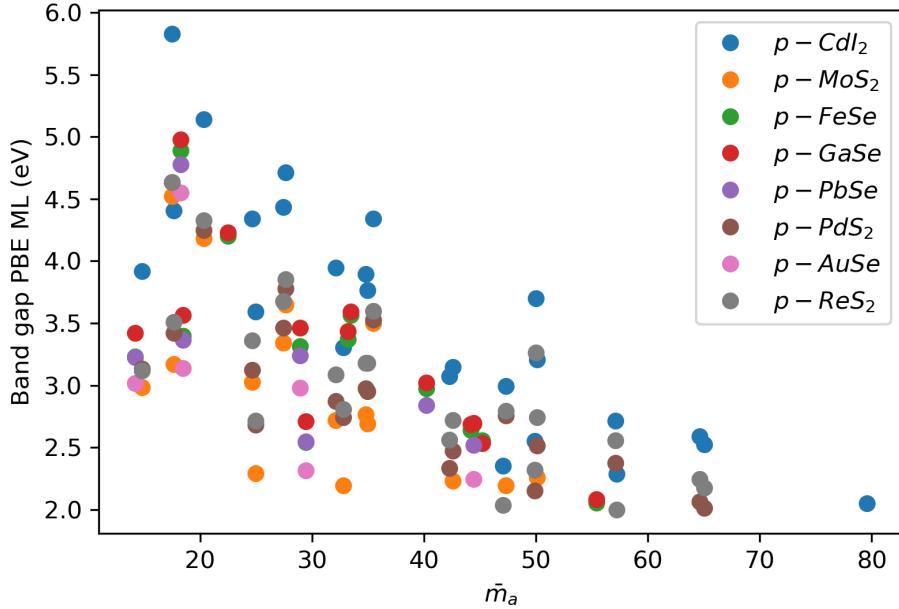


Figura 15: Band gap predito *versus* massa atômica ponderada.

A figura (15) exibe o comportamento do gap com a massa atômica ponderada dos compostos em cada protótipo, em que o peso é o número de átomos de cada espécie atômica. Embora existam compostos que possuam massa atômica ponderada próxima e gaps distintos (ex. BPO_4 e AlNO_4), a tendência geral é que em cada protótipo o gap diminua com a massa. Ademais, os materiais pertencentes à mesma estrutura termodinamicamente estável do CdI_2 apresentam o maior gap.

Além das estequiometrias ABC_2 e ABC_4 , foram propostos materiais da forma AB_2 . No lugar do átomo A aplicou-se alguns metais de transição (Sc , Ti , Cu , Zn , Zr , Nb , Mo , Ru , Rh , Pd , Ag , Cd , Hf , Ta , W , Re , Os , Ir , Pt , Au) e no lugar do átomo B alguns calcogênios e halogênios (F , Cl , Br , I , S , Se , Te). Os protótipos empregados em cada uma dessas combinações são MoS_2 , CdI_2 , GeS_2 , WTe_2 , ReS_2 e PdS_2 , gerando 840 materiais. O critério para a escolha dos protótipos é a presença deles na base de dados em materiais com a mesma estequiometria. Entre esses compostos considerou-se 479 sistemas para a classificação e regressão, pois os demais já existem na base de dados C2DB (versão 2018-2). Ao eliminar os materiais já presentes no banco de dados, os protótipos com maiores representantes são o WTe_2 , ReS_2 e PdS_2 . Ao total, 258 materiais são previstos como isolantes, dos quais 37 são considerados UWBG (gap PBE > 2 eV).

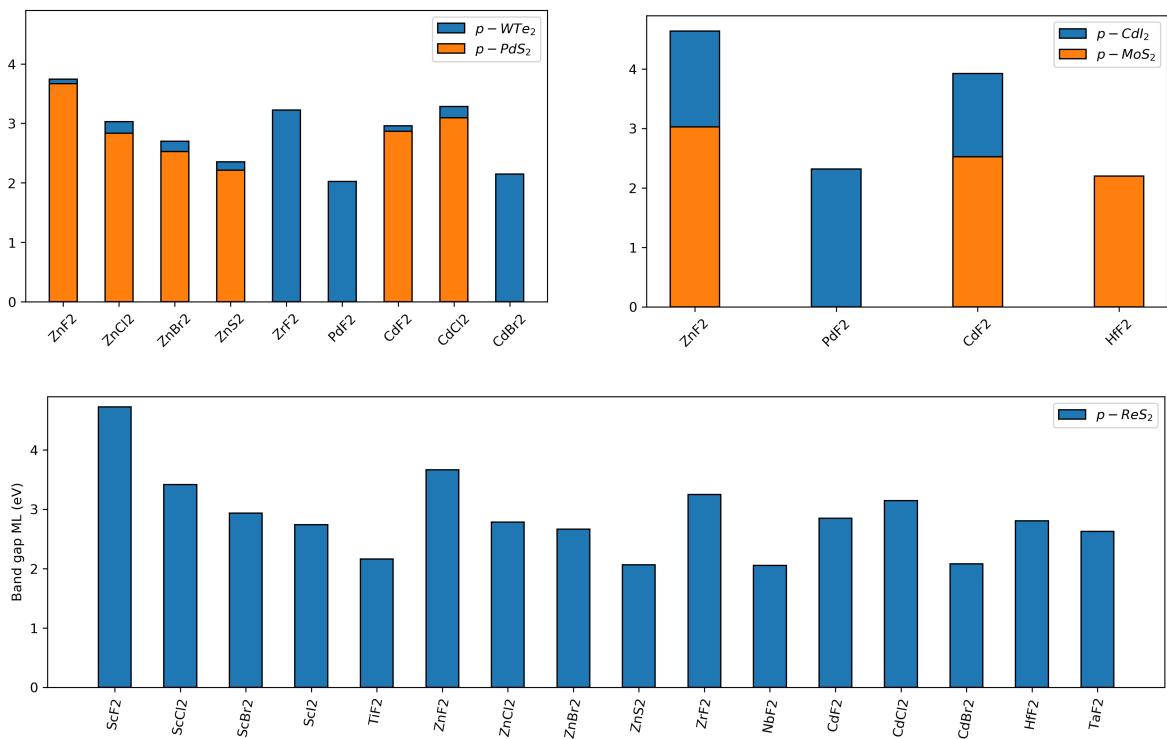


Figura 16: Diagrama de barras do gap previsto nos materiais AB_2 em cada protótipo.

A figura 16 exibe o diagrama de barras do gap para os materiais sugeridos na estequiometria AB_2 . A maioria dos UWBG previstos pertencem ao protótipo $p - \text{ReS}_2$, em que o ScF_2 é o material com o maior gap (4,73 eV). O segundo material com o maior gap é $\text{ZnF}_2 - \text{CdI}_2$ com 4,64 eV. O terceiro sistema com maior gap (CdF_2) também pertence ao protótipo CdI_2 , sendo igual a 3,92 eV. A próxima sessão apresenta a estrutura de bandas do material $\text{CdF}_2 - \text{CdI}_2$.

4.5 Cálculos da Estrutura Eletrônica

A estrutura cristalina do Fluoreto de Cádmio (CdF_2) no protótipo CdI_2 está representada na figura 17. Os cálculos das propriedades estruturais e eletrônicas foram realizadas com o código Quantum Espresso. O funcional de troca e correlação empregado é GGA-PBE e os orbitais de Kohn-Sham foram expandidos em uma base de ondas planas de acordo com o método PAW. A energia cinética máxima foi definida conforme o protocolo SSSP (PRANDINI et al., 2018), sendo igual a 50 Ry . Em relação aos pontos k , escolheu-se uma malha $11 \times 11 \times 1$.

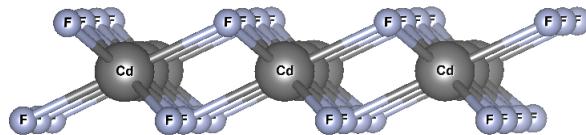


Figura 17: Estrutura do CdF_2 com protótipo CdI_2 .

As posições atômicas iniciais são as mesmas posições atômicas do protótipo. Na etapa de otimização estrutural é permitido variar essas posições em diferentes volumes fixos. O vácuo permanece em 20 Å. A figura 18 mostra a energia total da estrutura em cada volume, mas apresentada em função do parâmetro de rede. O mínimo ocorre em 3,58 Å.

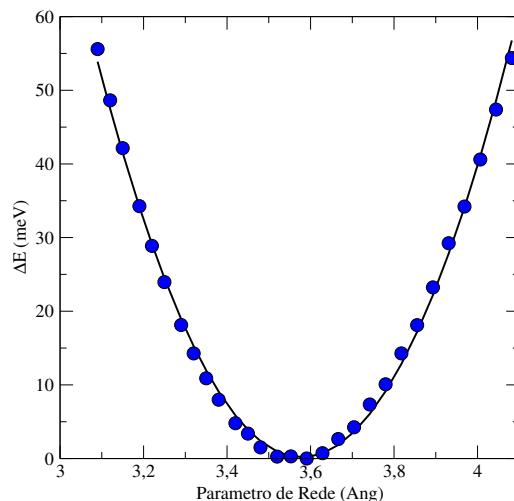


Figura 18: Otimização estrutural do CdF_2 na estrutura do protótipo CdI_2 .

De posse da estrutura otimizada, calculou-se a estrutura de bandas exibida na figura 19. O mínimo da banda de condução e o topo da banda de valência se encontram no ponto Γ , promovendo um gap direto e igual a 3,81 eV.

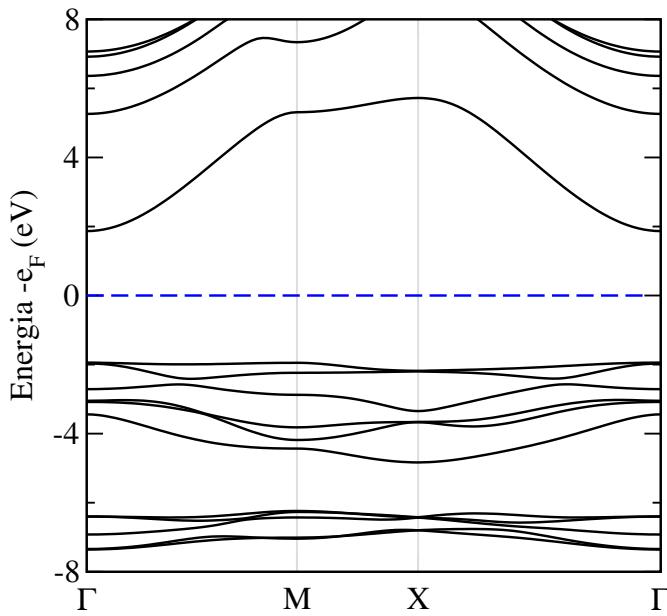


Figura 19: Estrutura de bandas do CdF₂ com protótipo CdI₂.

A comparação com as previsões feitas pelo modelo *Gradient Boosting* treinado no *dataset* geral e o *dataset* com apenas estequiometria AB₂ aparece na tabela 6. É possível notar que o resultado da simulação é próximo dos gaps previstos considerando as margens de erro do algoritmo.

Material	band gap (eV) - <i>datasetAB₂</i>	band gap (eV) <i>dataset</i> geral	DFT
CdF ₂ -CdI ₂	4,12	3,92	3,81

Tabela 6: Gap decorrente do modelo *Gradient Boosting* e DFT para o CdF₂–CdI₂.

5 Conclusões e Perspectivas

As ferramentas de aprendizado de máquina estão sendo empregadas em diversos campos da ciência de materiais, principalmente prevendo propriedades físicas. Para além da previsão de propriedades inexploradas de sistemas já conhecidos, seu poder preditivo pode ser extendido para propriedades de materiais novos. Diante da escassez no estudo de semicondutores bidimensionais de gap ultra largo, utilizou-se o aprendizado de máquina para prever os gaps dos compostos propostos por Schleder et. al. ([SCHLEDER; ACOSTA; FAZZIO, 2020](#)), cujo trabalho analisou a estabilidade termodinâmica deles. Com o modelo de classificação treinado e otimizado, é possível classificá-los em metais e isolantes. A separação desses materiais se mostra adequada no conjunto de teste, exibindo uma precisão média de 80% para o modelo *Gradient Boosting*. Na regressão, elaborada para prever o valor do gap, a menor raiz do erro quadrático médio (0,25 eV) também é alcançada no algoritmo *Gradient Boosting*. Entre os 800 novos materiais com estequiometrias ABC_2 e ABC_4 , os algoritmos treinados com os dados da base C2db ([HAASTRUP, 2018](#)) são capazes de identificar 266 isolantes, dos quais 134 são considerados materiais de gap ultra largo. A maioria dos semicondutores previstos são óxidos e apresentam maior gap quando pertencentes ao protótipo CdI_2 . Ademais, são sugeridos 479 materiais de estequiometria AB_2 que não estão presentes no banco de dados C2db (versão 2018-2). Entre eles, 37 sistemas são previstos com gap maior que 2,0 eV. A maior parte desses compostos UWBG pertencem à classe do protótipo ReS_2 , o qual também exibe o material ScF_2-ReS_2 com o maior gap esperado (4,73 eV). Da mesma forma, o CdF_2-CdI_2 também dispõe de um gap ultra largo, igual a 4,12 eV conforme o modelo *Gradient Boosting*. Além disso, esse gap é confirmado por cálculos computacionais utilizando a Teoria do Funcional da Densidade (DFT), sendo igual a 3,81 eV.

A utilização de *features* atômicas e dos protótipos permite continuar propondo novos materiais de diferentes estequiometrias, e que não demandam outros conhecimentos sobre a estrutura cristalina. Uma das propostas futuras na continuação deste trabalho é sugerir novos compostos de forma consistente com os princípios físicos necessários. Além disso,

é fundamental uma maior investigação sobre a eficiência desses preditores na previsão do gap. Sendo assim, outra perspectiva é realizar mais cálculos de estrutura de bandas através da Teoria do Funcional da Densidade (DFT) para comparar com o gap previsto pelo algoritmo de aprendizado de máquina. Com o mesmo intuito, também é possível sugerir novas features. Após as simulações de estrutura eletrônica, será igualmente necessário analisar a estabilidade termodinâmica e dinâmica desses semicondutores.

REFERÊNCIAS

- ALMEIDA, Joseane Santos. **Estudo Ab-initio da Estrutura Eletrônica do Nitreto de Tálio sob Pressão.** [S. l.: s. n.], 2020. Monografia (Bacharel em Física), UEFS (Universidade Estadual de Feira de Santana), Feira de Santana, Brasil.
- ALTMANN, S.L. **Band Theory of Solids: An introduction from the point of view of symmetry.** Nova York, USA: Oxford University Press, 1991.
- ALVES, Á. S. **Estudos de Magnetos Moleculares Através de Cálculos de Primeiros Princípios.** 2011. Tese (Doutorado) – Universidade Federal Fluminense, Rio de Janeiro, Brasil.
- BACHELET G. B.; HAMMAN, D.R.; SCHLUTER. Pseudopotentials that work: From H to Pu. **Phys. Rev. B**, v. 26, 1982.
- BALIGA, B. J. Semiconductors for high-voltage, vertical channel field-effect transistors. **Journal of Applied Physics**, v. 53, 3 1982. DOI: [10.1063/1.331646](https://doi.org/10.1063/1.331646).
- BALLESTÍN-FUERTES, Javier et al. Role of Wide Bandgap Materials in Power Electronics for Smart Grids Applications. **Electronics**, v. 10, n. 6, 2021. ISSN 2079-9292. DOI: [10.3390/electronics10060677](https://doi.org/10.3390/electronics10060677). Disponível em: <<https://www.mdpi.com/2079-9292/10/6/677>>.
- BARDEEN, J.; BRATTAIN, W. Three-Electrode Circuit Element Utilizing Semiconductive Materials. **United States Patent 2524035**, v. 4, n. 72, 1950.
- BASKURT, M. et al. Stable single-layers of calcium halides (CaX_2 , X = F, Cl, Br, I). **J. Chem. Phys.**, American Institute of Physics, v. 152, 16 2020. DOI: [10.1063/5.0006011](https://doi.org/10.1063/5.0006011). Disponível em: <<https://aip.scitation.org/doi/10.1063/5.0006011>>.
- BLÖCHL, P. E. Projector augmented-wave method. **Phys. Rev. B**, 1994.
- BUONANNO, Manuela; WELCH, David; BRENNER, Igor Shuryak David J. Far-UVC light (222 nm) efficiently and safely inactivates airborne human coronaviruses. **Nature Scientific Report**, 2020.

- CAPAZ, Rodrigo. **Elétrons em Sólidos: Elétrons em um Potencial Cristalino: Teorema de Bloch.** [S. l.: s. n.], 2019. Notas de Aula.
- CERPELEY D. M. ALDER, B. J. Ground State of the Electron Gas by a Stochastic Method. **Phys. Rev. Lett.**, v. 45, n. 566, 1980.
- CHAE, Sieun et al. Toward the predictive discovery of ambipolarly dopable ultra-wide-band-gap semiconductors: The case of rutile GeO_2 . **Appl. Phys. Lett.**, American Institute of Physics, v. 118, 2021. DOI: [10.1063/5.0056674](https://doi.org/10.1063/5.0056674). Disponível em: <<https://aip.scitation.org/doi/citedby/10.1063/5.0056674>>.
- CURTAROLO, S. et al. AFLOWLIB.ORG: A distributed materials properties repository from high-throughput ab initio calculations. **Computational Materials Science**, v. 58, p. 227–235, 2012.
- DARSHANA WICKRAMARATNE, Leigh Weston; WALLE, Chris G. Van de. Monolayer to Bulk Properties of Hexagonal Boron Nitride. **Appl. Phys. Lett.**, American Chemical Society, v. 122, 44 2018. DOI: [10.1021/acs.jpcc.8b09087](https://doi.org/10.1021/acs.jpcc.8b09087). Disponível em: <<https://pubs.acs.org/doi/10.1021/acs.jpcc.8b09087>>.
- DAVIS, Jesse; GOADRICH, Mark. The Relationship between Precision-Recall and ROC Curves. In: PROCEEDINGS of the 23rd International Conference on Machine Learning. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 2006. (ICML '06), p. 233–240. ISBN 1595933832. DOI: [10.1145/1143844.1143874](https://doi.org/10.1145/1143844.1143874). Disponível em: <<https://doi.org/10.1145/1143844.1143874>>.
- DIRAC, P. A. M. Note on Exchange Phenomena in the Thomas Atom. **Proc. Cambridge Phil. Roy. Soc.**, v. 26, p. 376–385, 1930.
- DRAXL C.; SCHEFFLER, M. The NOMAD laboratory: from data sharing to artificial intelligence. **J. Phys.: Mater.**, v. 2, n. 3, p. 036001, 2019.
- FELICIANO, G. **Materials Modelling Using Density Functional Theory: Properties and Predictions**. [S. l.]: Oxford University Press, 2014.
- FERMI, E. Un metodo statistico per la determinazione di alcune priorietà dell'atome. **Rend. Accad. Naz. Lincei**, v. 6, p. 602, 1927.
- FLACH, Peter; KULL, Meelis. Precision-Recall-Gain Curves: PR Analysis Done Right. In _____. **Advances in Neural Information Processing Systems**. [S. l.]: Curran Associates, Inc., 2015. v. 28. Disponível em: <<https://proceedings.neurips.cc/paper/2015/file/33e8075e9970de0cfeca955afd4644bb2-Paper.pdf>>.

- GALAZKA, Zbigniew et al. *MgGa₂O₄* as a new wide bandgap transparent semiconducting oxide: Growth and properties of bulk single crystals. **Phys. Status Solidi A**, WILEY VCH, v. 212, 7 2004. DOI: [10.1002/pssa.201431835](https://doi.org/10.1002/pssa.201431835).
- GE, Yanfeng et al. Electronic Transport of Two-Dimensional Ultrawide Bandgap Material h-BeO, 2020. Disponível em: <<https://arxiv.org/abs/2007.01090>>.
- GÉRON, Aurélien. **Hands-on Machine Learning with Scikit-Learn, Keras TensorFlow**. 2. ed. Sebastopol: O'Reilly Media, 2019.
- GIANNOZZI, P et al. QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials. **J. Phys.: Condens. Matter.**, v. 21, 2009.
- GROSSO G.; PARRAVICINI, G.P. **Solid State Physics**. [S. l.]: Academic Press, 2000.
- HAASTRUP, S. et al. The Computational 2D Materials Database: high-throughput modeling and discovery of atomically thin crystals. **2D Materials**, IOP Publishing, v. 5, n. 4, p. 042002, 2018.
- HAMANN, D. R.; M., SCHULÜTER; CHIANG, C. Norm-Conserving Pseudopotentials. **Phys. Rev. Lett.**, v. 43, 1979.
- HASTIE, Trevor; TIBSHIRANI, Robert; FRIEDMAN, Jerome. **The Elements of Statistical Learning**. 2. ed. Stanford, California: Springer, 2017.
- HIGASHIWAKI, Masataka; KAPLAR, Robert; PERNOT, Julien. Ultrawidebandgap semiconductors. **Appl. Phys. Lett.**, v. 118, n. 200401, 2021.
- HOHENBERG, P. KOHN. Inhomogeneous electron gas. **Phys. Rev.**, v. 136, 3B, 1964.
- JAIN, A. et al. Commentary: The Materials Project: A materials genome approach to accelerating materials innovation. **APL Materials**, v. 1, p. 011002, 2013.
- JIN, Haiwei et al. Review of wide band-gap semiconductors technology. **MATEC Web of Conferences**, v. 40, n. 01006, 2016.
- JOHNSON, E. Physical limitations on frequency and power parameters of transistors. In: 1958 IRE International Convention Record. [S. l.: s. n.], 1965. v. 13, p. 27–34. DOI: [10.1109/IRECON.1965.1147520](https://doi.org/10.1109/IRECON.1965.1147520).
- KEYES, R.W. Figure of merit for semiconductors for high-speed switches. **Proceedings of the IEEE**, v. 60, n. 2, p. 225–225, 1972. DOI: [10.1109/PROC.1972.8593](https://doi.org/10.1109/PROC.1972.8593).

- KHAN, Karim et al. Recent developments in emerging two-dimensional materials and their applications. **J. Mater. Chem. C**, The Royal Society of Chemistry, v. 8, p. 387–440, 2 2020. DOI: [10.1039/C9TC04187G](https://doi.org/10.1039/C9TC04187G). Disponível em: <<http://dx.doi.org/10.1039/C9TC04187G>>.
- KOHN W.; SHAM, L. J. Sham. Self-Consistent Equations Including Exchange and Correlation Effects. **Phys. Rev.**, v. 140, 4A, 1965.
- KUDRAWIEC, Robert; HOMMEL, Detlef. Bandgap engineering in III-nitrides with boron and group V elements: Toward applications in ultraviolet emitters. **Appl. Phys. Lett.**, American Institute of Physics, v. 7, 4 2020. DOI: [10.1063/5.0025371](https://doi.org/10.1063/5.0025371). Disponível em: <<https://aip.scitation.org/doi/10.1063/5.0025371>>.
- L. LINDSAY, D. A. Broido; REINECKE, T. L. First-Principles Determination of Ultrahigh Thermal Conductivity of Boron Arsenide: A Competitor for Diamond? **Appl. Phys. Lett.**, American Institute of Physics, v. 111, 2013. DOI: [10.1103/PhysRevLett.111.025901](https://doi.org/10.1103/PhysRevLett.111.025901). Disponível em: <<https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.111.025901>>.
- LARSEN, Ask Hjorth et al. The atomic simulation environment—a Python library for working with atoms. **Journal of Physics: Condensed Matter**, IOP Publishing, v. 29, n. 27, p. 273002, jun. 2017. DOI: [10.1088/1361-648x/aa680e](https://doi.org/10.1088/1361-648x/aa680e). Disponível em: <<https://doi.org/10.1088/1361-648x/aa680e>>.
- LIU, Xuejie et al. Optical and mechanical properties of C, Si, Ge, and 3C–SiC determined by first-principles theory using Heyd–Scuseria–Ernzerhof functional. **Materials Science in Semiconductor Processing**, v. 16, n. 6, p. 1369–1376, 2013. ISSN 1369-8001. DOI: <https://doi.org/10.1016/j.mssp.2013.04.017>. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S1369800113001212>>.
- MCCREARY, Amber et al. An outlook into the flat land of 2D materials beyond graphene: synthesis, properties and device applications. **2D Materials**, IOP Publishing, v. 8, n. 1, p. 013001, dez. 2020. DOI: [10.1088/2053-1583/abc13d](https://doi.org/10.1088/2053-1583/abc13d). Disponível em: <<https://doi.org/10.1088/2053-1583/abc13d>>.
- MENGLE, Kelsey. **First-Principles Calculations on the Electronic, Optical, and Vibrational Properties of Ultrawide-Band-Gap Semiconductor Materials**. 2020. Tese (Doutorado) – Universidade de Michigan, The address of the publisher.

- METRICS and scoring: quantifying the quality of predictions. [S. l.: s. n.], c2007-2022. Acessado: 07 de julho, 2022. Disponível em: <https://scikit-learn.org/stable/modules/model%5C_evaluation.html%5C#davis2006>.
- MISRA, Prasanta K. **Physics of Condensed Matter**. [S. l.]: Academic Press, 2012.
- MOSESA, Poul Georg et al. Hybrid functional investigations of band gaps and band alignments for AlN, GaN, InN, and InGaN. **J. Chem. Phys.**, v. 134, 8 2011. DOI: <https://doi.org/10.1063/1.3548872>. Disponível em: <<https://aip.scitation.org/doi/abs/10.1063/1.3548872?ver=pdfcov&journalCode=jcp>>.
- NEIL W. A, Mermin D. **Solid State Physics**. [S. l.]: Thomson Learning, 1976.
- ONG, Shyue Ping et al. Python Materials Genomics (pymatgen): A robust, open-source python library for materials analysis. **Computational Materials Science**, v. 68, p. 314–319, 2013. ISSN 0927-0256. DOI: <https://doi.org/10.1016/j.commatsci.2012.10.028>. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0927025612006295>>.
- PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.
- PERDEW J. P.; ERNZERHOF M.; BURKE, K. Generalized Gradient Approximation Made Simple. **Phys. Rev. Lett.**, v. 77, n. 18, 1996.
- PHILLIPS, J. C.; KLEINMAN., L. New Method for Calculating Wave Functions in Crystals and Molecules. **Phys. Rev.**, v. 116, 1959.
- PRANDINI, Gianluca et al. Precision and efficiency in solid-state pseudopotential calculations. **npj Comput Mater.**, v. 4, n. 72, 2018. DOI: [10.1038/s41524-018-0127-2](https://doi.org/10.1038/s41524-018-0127-2). Disponível em: <<https://www.nature.com/articles/s41524-018-0127-2>>.
- RAVICHANDRAN, Ram; WANG, Alan X.; WAGER, John F. Solid state dielectric screening versus band gap trends and implications. **Optical Materials**, v. 60, p. 181–187, 2016. ISSN 0925-3467. DOI: <https://doi.org/10.1016/j.optmat.2016.07.027>. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0925346716303834>>.
- RICHARD M.RICHARD, M. **Electronic Structure Basic Theory and Practical Methods**. Reino Unido: Cambridge University Press, 2004.
- ROSTGAARD, C. **The Projector Augmented-wave Method**. [S. l.: s. n.], 2009. Apostila.

- SANTOS, R. N. **Propriedades Estruturais e Magnéticas do $KNaCuSi_4O_{10}$ Utilizando Cálculo de Primeiros Princípios.** [S. l.: s. n.], 2017. Monografia (Bacharel em Física), UEFS (Universidade Estadual de Feira de Santana), Feira de Santana, Brasil.
- SCHLEDER, Gabriel R. **Machine Learning for Materials Science.** Agosto 2021. Tese (Doutorado) – Universidade Federal do ABC, São Paulo, Brasil.
- SCHLEDER, Gabriel R.; ACOSTA, Carlos Mera; FAZZIO, Adalberto. Exploring Two-Dimensional Materials Thermodynamic Stability via Machine Learning. **ACS Appl. Mater. Interfaces**, American Chemical Society, v. 12, 18 2020. DOI: [10.1021/acsami.9b14530](https://doi.org/10.1021/acsami.9b14530). Disponível em: <<https://pubs.acs.org/doi/10.1021/acsami.9b14530>>.
- SCHMIDT, Jonathan et al. Recent advances and applications of machine learning in solid-state materials science. **npj Comput Mater**, v. 5, n. 83, 2019. DOI: [10.1038/s41524-019-0221-0](https://doi.org/10.1038/s41524-019-0221-0).
- SHEVITSKI, Brian et al. Blue-light-emitting color centers in high-quality hexagonal boron nitride. **Phys. Rev. B**, American Physical Society, v. 100, 44 2019. DOI: [10.1103/PhysRevB.100.155419](https://doi.org/10.1103/PhysRevB.100.155419). Disponível em: <<https://journals.aps.org/prb/abstract/10.1103/PhysRevB.100.155419>>.
- SHOLL D.S.; STECKEL, J. **Density functional theory : a practical introduction.** New Jersey: John Wiley & Sons, Inc., Hoboken, 2009.
- STOLL, Anke; BENNER, Peter. Machine learning for material characterization with an application for predicting mechanical properties. **Special Issue: Scientific Machine Learning - Part I**, Wiley-VCH GmbH, v. 44, 1 2021. DOI: [10.1002/gamm.202100003](https://doi.org/10.1002/gamm.202100003).
- THOMAS, L.H. The calculation of atomic fields. **Proc. Cambridge Phil. Roy. Soc.**, v. 23, n. 542, 1927.
- TROULLIER N.; MARTINS, J.L. Efficient pseudopotentials for plane-wave calculations. **Phys. Rev.**, v. 43, 1993.
- TSAO, J. Y. et al. Ultrawide-Bandgap Semiconductors: Research Opportunities and Challenges. **Advanced Electronic Materials**, v. 4, 2018.
- TUMELERO, Milton Andre. **Desenvolvimento de um Sistema de Medidas de Transporte de Carga em Função da Temperatura em Semicondutores.** 2010. Diss. (Mestrado) – Universidade Federal de Santa Catarina, Florianópolis - SC.

- VANDERBILT, David. Soft self-consistent pseudopotentials in a generalized eigenvalue formalism. **Phys. Rev. B**, American Physical Society, v. 41, p. 7892–7895, 11 abr. 1990. DOI: [10.1103/PhysRevB.41.7892](https://doi.org/10.1103/PhysRevB.41.7892). Disponível em: <<https://link.aps.org/doi/10.1103/PhysRevB.41.7892>>.
- VARLEY, J. B. et al. Role of self-trapping in luminescence and *p*-type conductivity of wide-band-gap oxides. **Phys. Rev. B**, American Physical Society, v. 85, p. 081109, 8 fev. 2012. DOI: [10.1103/PhysRevB.85.081109](https://doi.org/10.1103/PhysRevB.85.081109). Disponível em: <<https://link.aps.org/doi/10.1103/PhysRevB.85.081109>>.
- VIANNA, J. D. M.; CANUTO S.; FAZZIO, A. **Teoria Quântica de Moléculas e Sólidos**. [S. l.]: Livraria da Física, 2004.
- WALLE, Chris G. Van de; NEUGEBAUER, Jörg. First-principles calculations for defects and impurities: Applications to III-nitrides. **J. Appl. Phys.**, American Institute of Physics, v. 95, 8 2004. DOI: [10.1063/1.1682673](https://doi.org/10.1063/1.1682673). Disponível em: <<https://aip.scitation.org/doi/10.1063/1.1682673>>.
- YAN, Yanfa; WEI, Su-Huai. Doping asymmetry in wide-bandgap semiconductors: Origins and solutions. **Phys. Stat. Sol.**, WILEY-VCH, v. 245, 4 2008. DOI: [10.1002/pssb.200743334](https://doi.org/10.1002/pssb.200743334). Disponível em: <<https://onlinelibrary.wiley.com/doi/abs/10.1002/pssb.200743334>>.
- YE, Xiao-Juan et al. Two-dimensional CaFCl: ultra-wide bandgap, strong interlayer quantum confinement, and n-type doping. **Phys. Chem. Chem. Phys.**, The Royal Society of Chemistry, v. 22, p. 17213–17220, 30 2020. DOI: [10.1039/DOPC02804E](https://doi.org/10.1039/DOPC02804E). Disponível em: <<http://dx.doi.org/10.1039/DOPC02804E>>.
- ZEMAN, Charles J. et al. Investigation of p-type doping in - and -Ga₂O₃. **Journal of Alloys and Compounds**, v. 877, p. 160227, 2021. ISSN 0925-8388. DOI: <https://doi.org/10.1016/j.jallcom.2021.160227>. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0925838821016364>>.
- ZHANG, Ying; LING, Chen. A strategy to apply machine learning to small datasets in materials science. **npj Comput Mater**, v. 4, n. 25, 2018. DOI: [10.1038/s41524-018-0081-z](https://doi.org/10.1038/s41524-018-0081-z).
- ZHU, Zhenxue. Two-dimensional BaFCl monolayer: tunable bandgap and pronounced deep ultraviolet absorption. **IOP Conference Series: Earth and Environmental Science**, IOP Publishing, v. 526, n. 1, p. 012016, jun. 2020. DOI: [10.1088/1755-1315/526/1/012016](https://doi.org/10.1088/1755-1315/526/1/012016). Disponível em: <<https://doi.org/10.1088/1755-1315/526/1/012016>>.

ZUNGER A.; COHEN, M. First-principles nonlocal-pseudopotential approach in the density-functional formalism: Development and application to atoms. **Phys. Rev. B**, v. 18, 1978.

APÊNDICE A - Algoritmos

A tabela com as propriedades atômicas de cada átomo estão disponíveis no material suplementar do artigo de Schleider *et. al.*, disponível em <<https://pubs.acs.org/doi/10.1021/acsami.9b14530>>. Ela é disponibilizada em formato *csv* e possui o nome Schleider2019_AtomicTable.csv. De posse dessa tabela, construímos o *dataset* com estequiometria AB₂ e o *dataset* geral.

A.1 *Dataset* AB₂

```

1 # importando pacotes
2 from pymatgen.core.composition import *
3 import numpy as np
4 import pandas as pd
5 import ase.db
6
7 # Lendo o arquivo com as propriedades atômicas
8 df_atoms = pd.read_csv('Schleider2019_AtomicTable.csv')
9 print(df_atoms)
10
11 # conectando à base de dados
12 data = ase.db.connect('/home/joseane/Dropbox/Joseane/ML_initial_tests/
13                         c2db/old/c2db.db')
14
15 #selecionando materiais não magnéticos
16 rows = data.select(is_magnetic=False)
17
18 df_materials = pd.DataFrame(columns=['Material', 'Band gap', 'Atom1',
19                               'Atom2', 'prototype'])
20
21 for row in rows:
22     try:
23         formula_information = Composition(row.formula).as_dict()

```

```
22     lista = list(formula_information.items())
23     if row.stoichiometry=='AB2': # garante estequiometria AB2
24         nova_entrada = [row.formula, row.gap, lista[0][0], lista
25 [1][0], row.prototype]
26         df_materials.loc[len(df_materials)] = nova_entrada
27     except:
28         pass
29
30 print(df_materials)
31
32 ### ver número de metais e isolantes
33 metal = insulator = 0
34 for gap in df_materials['Band gap']:
35     if (gap < 0.000001):
36         metal += 1
37     else:
38         insulator += 1
39
40 print('Num. Metals : ',metal)
41 print('Num. Insulators: ', insulator)
42
43 merge1 = pd.merge(df_materials, df_atoms, how='inner', left_on='Atom1',
44                   right_on='Element')
45 print(merge1)
46
47 merge2 = pd.merge(merge1, df_atoms, how='inner', left_on='Atom2',
48                   right_on='Element')
49 print(merge2)
50
51 # Organizando o dataframe retirando as colunas desnecessárias
52 df = merge2
53 df.drop(columns=['Atom1', 'Atom2', 'Element_x', 'Element_y'], inplace=
54           True)
55 df.head(10)
56
57 # exporta para um arquivo csv
58 df.to_csv('AB2_prototype_atomic.csv')
59
60 ## separando os isolantes ##
61 df = df[df['Band gap'] != 0.00000000]
62
63 # exporta para um arquivo csv
```

```
61 df.to_csv('AB2_prototype_atomic_insulators.csv')
```

A.2 *Dataset* geral

```
1 # importando pacotes
2 from pymatgen.core.composition import *
3 import matplotlib.pyplot as plt
4 import numpy as np
5 import pandas as pd
6 import ase.db
7
8 # Lendo o arquivo com as propriedades atômicas
9 df_atoms = pd.read_csv('Schleder2019_AtomicTable.csv')
10 print(df_atoms)
11
12 # cria uma lista com as estequiometrias de todos os materiais. Nesse
13 # caso, elas vão se repetir.
14 data = ase.db.connect('/home/joseane/Dropbox/Joseane/ML_initial_tests/
15   c2db/old/c2db.db')
16 rows = data.select(is_magnetic = False)
17
18 lista = []
19
20 # verifica quais estequiometrias estão envolvidas (sem repetição)
21 print(set(lista))
22
23 #verifica a quantidade de vezes que ocorre cada estequiometria
24 print(lista.count('AB3C8'))
25 print(lista.count('AB2C2D2E24F8'))
26 print(lista.count('AB2'))
27 print(lista.count('ABC3'))
28 print(lista.count('A2B3C4'))
29 print(lista.count('A'))
30 print(lista.count('AB2C2D2'))
31 print(lista.count('ABC'))
32 print(lista.count('A3B4'))
33 print(lista.count('AB2C2'))
34 print(lista.count('A2B2C3'))
35 print(lista.count('A2B2C3D4'))
36 print(lista.count('ABC4'))
```

```
37 print(lista.count('A2B2C2D3'))
38 print(lista.count('ABC2'))
39 print(lista.count('AB2C24D4E8'))
40 print(lista.count('AB3'))
41 print(lista.count('A2B3'))
42 print(lista.count('AB'))
43
44 # Plota o digrama de barras com as estequiometrias disponíveis
45 plt.title('Estequiometrias C2db', fontsize=15)
46 plt.xlabel('Estequiometrias')
47 plt.ylabel('Frequência Absoluta')
48 x = ['AB2', 'AB', 'ABC', 'AB3', 'outras']
49 plt.bar(x, height=[876, 705, 535, 179, 390],
50          label='Data', color='blue', edgecolor='black')
51 plt.savefig('bar_stoi', dpi=300)
52
53 ## Transformando o dataframe em dicionário em que os elementos são as
54 ## chaves.
54 df_atoms.set_index('Element', inplace = True)
55 dicio = df_atoms.to_dict('index')
56 print(dicio)
57
58 prop = ['Z',
59          'Electronegativity',
60          'IonizationPotential',
61          'ElectronAffinity',
62          'HOMO',
63          'LUMO',
64          'r_s_orbital',
65          'r_p_orbital',
66          'r_d_orbital',
67          'r_atomic_nonbonded',
68          'r_valence_lastorbital',
69          'r_covalent',
70          'Valence',
71          'PeriodicColumn',
72          'PeriodicColumn_uppto18',
73          'NumberUnfilledOrbitals',
74          'Polarizability']
75
76 ## Listas que guardarão cada propriedade de cada elemento no composto
77 ## por vez. ##
77 lista = []
```

```
78 pesos = []
79
80 ## Dicionário com as features estatísticas de todas as propriedades para
     cada material##
81 media_interm = {}
82
83 ## Lista que guarda cada dicionário de cada material para levar para um
     dataframe ##
84 lista_completa = []
85
86
87 for row in rows:
88
89     try:
90         comp = Composition(row.formula).as_dict()
91         elem = list(comp.items())
92
93         ## Acrescentando a fórmula química ##
94         media_interm['Material'] = row.formula
95
96         ## Acrescentando o grupo espacial ##
97         media_interm['prototype'] = row.prototype
98
99         ## Acrescentando o gap ##
100        media_interm['Band gap'] = row.gap
101
102        for i in prop:
103            ## Lista com a propriedade de cada átomo ##
104            for m in range(0, len(elem)):
105                lista.append(dicio[elem[m][0]][i])
106                pesos.append(elem[m][1])
107
108            ## Valor médio ##
109            media_interm[f'media_{i}'] = np.mean(lista)
110
111
112            ## Média ponderada ##
113            avg = np.average(lista, weights=pesos)
114            media_interm[f'media_pon_{i}'] = avg
115
116            ## Valor máximo e mínimo ##
117            max_prop = max(lista)
118            min_prop = min(lista)
```

```
119     media_interm[f'max_{i}'] = max_prop
120     media_interm[f'min_{i}'] = min_prop
121
122     ## Desvio padrão em relação a média ##
123     media_interm[f'desvio_{i}'] = np.std(lista)
124
125     ## Desvio padrão em relação a média ponderada ##
126     sum_prop = 0
127     for j in lista:
128         sub2 = (j - avg)**2
129         sum_prop = sum_prop + sub2
130     media_interm[f'desvio_pon_{i}'] = np.sqrt(sum_prop/len(lista))
131
132     lista.clear()
133     pesos.clear()
134
135     lista_completa.append(media_interm.copy())
136 except:
137     pass
138
139 lista_completa
140
141 #print(lista, pesos)
142
143
144 df = pd.DataFrame(lista_completa)
145 print(df.sample(20, random_state=100))
146
147 ## vendendo número de metais e isolantes ##
148 n_metal = n_insulator = 0
149
150 for i in df['Band gap']:
151     if i > 0.0000000001:
152         n_insulator += 1
153     else:
154         n_metal += 1
155
156 print(f' No de metais: {n_metal} \n No de isolantes: {n_insulator}')
157
158 # esportando para um arquivo csv
159 df.to_csv('dataset_pbe_prototype_com_metais.csv')
160
```

```
161 # Separar os isolantes ##
162
163 df = df[df['Band gap'] != 0.0000000]
164
165 # Exportando o dataframe como csv
166 df.to_csv('dataset_pbe_prototype_isolantes.csv')
```

A.3 Dataset com materiais novos de estequiometrias ABC₂ e ABC₄

Primeiro extrai-se a tabela S3 (originalmente em formato pdf), disponível no material suplementar do artigo de Schleder *et. al.*, acessível em <<https://pubs.acs.org/doi/10.1021/acsami.9b14530>>.

```
1
2 ## extraindo tabela do paper do artigo do Gabriel ##
3 import tabula
4 import pandas as pd
5
6 ## Cada tabela identificada vira um dataframe e read_pdf retorna uma
    lista desses dataframes ##
7 ## necessário guess = False para ele pegar todas as tabelas corretamente
    . ##
8 list_of_dfs = tabula.read_pdf('suporte_inform.pdf', pages='all', guess=
    False)
9 len(list_of_dfs)
10
11 # Configura o número de linhas e colunas que serão mostrados, e o
    comprimento das colunas nos dataframes
12 pd.set_option('display.max_columns', None)
13 pd.set_option('display.max_rows', None)
14 pd.set_option('max_colwidth', None)
15
16 ## Todas estão vindo com o título 'Supporting Information' que é o cabeç
    alho do pdf. ##
17 ## A última linha dos dataframes tem sido a página do pdf ##
18 print(list_of_dfs[9])
19
20 ## retirando as três primeiras linhas do dataframe 9. Os outros não
    precisam, pois as páginas não tem título. ##
21 ## Excluindo a última linha também, pois é o número da página do pdf. ##
```

```
22 list_of_dfs[9].drop([0,1,2, 27], inplace=True)
23
24 ## mudando o nome da coluna dos dfs e excluindo a última linha ##
25 list_of_dfs[9].columns = ['Teste']
26 for i in range(10,47):
27     list_of_dfs[i].columns = ['Teste']
28     indice = len(list_of_dfs[i]) - 1
29     list_of_dfs[i].drop(indice, inplace=True)
30
31 ## fazendo isso apenas para não ter que escrever cada um no próximo
32 ## passo. Copia a saída e cola no próximo passo.##
33 for i in range(9,47):
34     print(f'list_of_dfs[{i}]')
35
36 ## Juntando dataframes ##
37 df = pd.concat([list_of_dfs[9],
38 list_of_dfs[10],
39 list_of_dfs[11],
40 list_of_dfs[12],
41 list_of_dfs[13],
42 list_of_dfs[14],
43 list_of_dfs[15],
44 list_of_dfs[16],
45 list_of_dfs[17],
46 list_of_dfs[18],
47 list_of_dfs[19],
48 list_of_dfs[20],
49 list_of_dfs[21],
50 list_of_dfs[22],
51 list_of_dfs[23],
52 list_of_dfs[24],
53 list_of_dfs[25],
54 list_of_dfs[26],
55 list_of_dfs[27],
56 list_of_dfs[28],
57 list_of_dfs[29],
58 list_of_dfs[30],
59 list_of_dfs[31],
60 list_of_dfs[32],
61 list_of_dfs[33],
62 list_of_dfs[34],
63 list_of_dfs[35],
64 list_of_dfs[36],
```

```
64 list_of_dfs[37] ,
65 list_of_dfs[38] ,
66 list_of_dfs[39] ,
67 list_of_dfs[40] ,
68 list_of_dfs[41] ,
69 list_of_dfs[42] ,
70 list_of_dfs[43] ,
71 list_of_dfs[44] ,
72 list_of_dfs[45] ,
73 list_of_dfs[46]], axis=0)
74 print(df)

75

76 ## Separando as colunas ##
77 divisao = df['Teste'].str.split(' ')
78 divisao.head()

79

80 material = divisao.str.get(0)
81 prototipo = divisao.str.get(1)
82 estabilidade = divisao.str.get(2)

83

84 df['Material'] = material
85 df['Prototipo'] = prototipo
86 df['Estabilidade'] = estabilidade
87 df.drop('Teste', axis=1, inplace=True)
88 print(df)

89

90 print(df['Prototipo'].value_counts())

91

92 ## Excluindo os protótipos que não aparecem em isolantes no c2db ##
93

94 ## excluindo o protótipo Rh0 ##
95 df2 = df[df['Prototipo'] != 'Rh0']
96 print(df2.shape)

97

98 ## excluindo o protótipo NiSe ##
99 df3 = df2[df2['Prototipo'] != 'NiSe']
100 print(df3.shape)

101

102 ## vendo se a exclusão funcionou ##
103 print(df3['Prototipo'].value_counts())

104

105 ## excluindo a coluna da estabilidade ##
106 df3.drop(columns='Estabilidade', inplace=True)
```

```
107 print(df3)
```

Agora no mesmo código, construímos as features estatísticas e acrescentamos o protótipo.

```
1
2 from pymatgen.core.composition import *
3 import numpy as np
4 import pandas as pd
5 import ase.db
6 import json
7 import re
8
9 # Lendo o arquivo com as propriedades atômicas
10 df_atoms = pd.read_csv('/home/joseane/Dropbox/Joseane/exercícios/
11   construindo_dataset_statistical_features/Schleider2019_AtomicTable.csv
12   ')
13 df_atoms
14
15 ## Transformando o dataframe em dicionário em que os elementos são as
16   chaves.
17 df_atoms.set_index('Element', inplace = True)
18 dicio = df_atoms.to_dict('index')
19 dicio
20
21 prop = ['Z',
22   'Electronegativity',
23   'IonizationPotential',
24   'ElectronAffinity',
25   'HOMO',
26   'LUMO',
27   'r_s_orbital',
28   'r_p_orbital',
29   'r_d_orbital',
30   'r_atomic_nonbonded',
31   'r_valence_lastorbital',
32   'r_covalent',
33   'Valence',
34   'PeriodicColumn',
35   'PeriodicColumn_uppto18',
36   'NumberUnfilledOrbitals',
37   'Polarizability']
```

```
36 ## Listas que guardarão cada propriedade de cada elemento no composto
37 ## por vez. ##
38 lista = []
39 pesos = []
40
41 ## Dicionário com as features estatísticas de todas as propriedades para
42 ## cada material##
43 media_interm = {}
44
45
46
47 for i in range(0,1000):
48     try:
49         formula = df3.iloc[i]['Material']
50         comp = Composition(formula).as_dict()
51         elem = list(comp.items())
52
53         ## Acrescentando a fórmula química ##
54         media_interm['Material'] = formula
55
56         ## Acrescentando o grupo espacial ##
57         media_interm['Prototipo'] = df3.iloc[i]['Prototipo']
58
59         for i in prop:
60             ## Lista com a propriedade de cada átomo ##
61             for m in range(0, len(elem)):
62                 lista.append(dicio[elem[m][0]][i])
63                 pesos.append(elem[m][1])
64
65             ## Valor médio ##
66             media_interm[f'media_{i}'] = np.mean(lista)
67
68
69             ## Média ponderada ##
70             avg = np.average(lista, weights=pesos)
71             media_interm[f'media_pon_{i}'] = avg
72
73             ## Valor máximo e mínimo ##
74             max_prop = max(lista)
75             min_prop = min(lista)
```

```

76     media_interm[f'max_{i}'] = max_prop
77     media_interm[f'min_{i}'] = min_prop
78
79     ## Desvio padrão em relação a média ##
80     media_interm[f'desvio_{i}'] = np.std(lista)
81
82     ## Desvio padrão em relação a média ponderada ##
83     sum_prop = 0
84     for j in lista:
85         sub2 = (j - avg)**2
86         sum_prop = sum_prop + sub2
87     media_interm[f'desvio_pon_{i}'] = np.sqrt(sum_prop/len(lista))
88
89     lista.clear()
90     pesos.clear()
91
92     lista_completa.append(media_interm.copy())
93 except:
94     pass
95
96 lista_completa
97 #print(lista, pesos)
98
99 df = pd.DataFrame(lista_completa)
100 print(df.sample(20, random_state=100))
101
102 print(df.shape)
103
104 print(df['Prototipo'].value_counts())
105
106 df = df.rename(columns={'Prototipo': 'prototype'})
107 print(df.head())
108
109 ## exportando para csv ##
110 df.to_csv('dataset_new_prototipo.csv')

```

A.4 Dataset de materiais novos com estequiometria AB₂ - features estatísticas

```

1 from pymatgen.core.composition import *

```

```

2 import numpy as np
3 import pandas as pd
4 import ase.db
5 import json
6 import re
7
8 TM=['Sc','Ti','Cu','Zn','Zr','Nb','Mo','Ru','Rh','Pd','Ag','Cd','Hf','Ta'
     ','W','Re','Os','Ir','Pt','Au']
9 HL=['F','Cl','Br','I','S','Se','Te']
10 STCH=['AB2']
11 PROT=['MoS2','CdI2','GeS2', 'WTe2', 'ReS2', 'PdS2']
12
13
14 elem=list(Composition(STCH[0]).as_dict().items())
15 n=0
16 new = {}
17 lista = []
18
19 for i in range(len(STCH)):
20     elem=list(Composition(STCH[i]).as_dict().items())
21     for j in range(len(TM)):
22         for k in range(len(HL)):
23             for l in range(len(PROT)):
24
25                 if(int(elem[0][1])==1):
26                     if(int(elem[1][1])==1):
27                         new['Material']=("%s%s"%(TM[j],HL[k]))
28                     else:
29                         new['Material']=("%s%s%s"%(TM[j],HL[k],str(int(
29
    elem[1][1]))))
30
31                         #new['Material']=("%s%s%s"%(TM[j],HL[k],str(int(elem
31
    [1][1]))))
32
33                     else:
34                         if(int(elem[1][1])==1):
35                             new['Material']=("%s%s%s%s"%(TM[j],str(int(elem
35
    [0][1])),HL[k]))
36                     else:
37                         new['Material']=("%s%s%s%s"%(TM[j],str(int(elem
37
    [0][1])),HL[k],str(int(elem[1][1]))))
38
39

```

```
40             new['Prototype']=(PROT[1])
41             lista.append(new.copy())
42             #print(STCH[i],TM[j],HL[k],new)
43             n+=1
44 print(len(lista))
45 df2 = pd.DataFrame(lista)
46 print(df2.sample(20, random_state=100))
47
48 ## excluindo os sistemas que já estão na base de dados ##
49 data = ase.db.connect('/home/joseane/Dropbox/Joseane/ML_initial_tests/
50   c2db/old/c2db.db')
51 rows = data.select()
52
53 for row in rows:
54     for i in df2.index:
55         if row.formula == df2['Material'][i] and row.prototype ==
56             df2['Prototype'][i]:
57             print(row.formula, row.prototype)
58             df2.drop(i, inplace=True)
59
60 print(df2)
61
62 ## construindo o \textit{dataset} com as \textit{features} estatísticas.
63 ## 
64
65 # Lendo o arquivo com as propriedades atômicas
66 df_atoms = pd.read_csv('./Schleder2019_AtomicTable.csv')
67 print(df_atoms)
68
69 ## Transformando o dataframe em dicionário em que os elementos são as
70 ## chaves.
71 df_atoms.set_index('Element', inplace = True)
72 dicio = df_atoms.to_dict('index')
73 print(dicio)
74
75 prop = ['Z',
76          'Electronegativity',
77          'IonizationPotential',
78          'ElectronAffinity',
79          'HOMO',
80          'LUMO',
81          'r_s_orbital',
82          'r_p_orbital',
```

```
79     'r_d_orbital',
80     'r_atomic_nonbonded',
81     'r_valence_lastorbital',
82     'r_covalent',
83     'Valence',
84     'PeriodicColumn',
85     'PeriodicColumn_uppto18',
86     'NumberUnfilledOrbitals',
87     'Polarizability']

88

89 ## Listas que guardarão cada propriedade de cada elemento no composto
  por vez. ##
90 lista = []
91 pesos = []

92

93 ## Dicionário com as features estatísticas de todas as propriedades para
  cada material##
94 media_interm = {}

95

96 ## Lista que guarda cada dicionário de cada material para levar para um
  dataframe ##
97 lista_completa = []

98

99

100 for i in range(0,100000):
101     try:
102         formula = df2.iloc[i]['Material']
103         comp = Composition(formula).as_dict()
104         elem = list(comp.items())

105

106         ## Acrescentando a fórmula química ##
107         media_interm['Material'] = formula

108

109         ## Acrescentando o grupo espacial ##
110         media_interm['Prototype'] = df2.iloc[i]['Prototype']

111

112         for i in prop:
113             ## Lista com a propriedade de cada átomo ##
114             for m in range(0, len(elem)):
115                 lista.append(dicio[elem[m][0]][i])
116                 pesos.append(elem[m][1])

117             ## Valor médio ##
```

```
119     media_interm[f'media_{i}'] = np.mean(lista)
120
121
122     ## Média ponderada ##
123     avg = np.average(lista, weights=pesos)
124     media_interm[f'media_pon_{i}'] = avg
125
126     ## Valor máximo e mínimo ##
127     max_prop = max(lista)
128     min_prop = min(lista)
129     media_interm[f'max_{i}'] = max_prop
130     media_interm[f'min_{i}'] = min_prop
131
132     ## Desvio padrão em relação a média ##
133     media_interm[f'desvio_{i}'] = np.std(lista)
134
135     ## Desvio padrão em relação a média ponderada ##
136     sum_prop = 0
137     for j in lista:
138         sub2 = (j - avg)**2
139         sum_prop = sum_prop + sub2
140     media_interm[f'desvio_pon_{i}'] = np.sqrt(sum_prop/len(lista))
141
142     lista.clear()
143     pesos.clear()
144
145     lista_completa.append(media_interm.copy())
146 except:
147     pass
148
149 lista_completa
150 #print(lista, pesos)
151
152 df = pd.DataFrame(lista_completa)
153 print(df.sample(20, random_state=100))
154
155 ## exportando para csv ##
156 df.to_csv('dataset_new_prototype.csv')
```

A.5 Dataset de materiais novos com estequiometria AB₂ - features atômicas

O dataset criado anteriormente com estequiometria AB₂ e features estatísticas será utilizado aqui apenas para aproveitar as colunas que já possuem o nome dos materiais e os protótipos. Antes disso ele foi renomeado de ‘dataset_new_prototype.csv’ para ‘dataset_new_prototipo.csv’.

```

1 from pymatgen.core.composition import *
2 import numpy as np
3 import pandas as pd
4 import ase.db
5
6 # Lendo o arquivo com as propriedades atômicas
7 df_atoms = pd.read_csv('./Schleder2019_AtomicTable.csv')
8 print(df_atoms)
9
10 seed=50
11 df_new = pd.read_csv('dataset_new_prototipo.csv')
12 df_new.drop('Unnamed: 0', axis='columns', inplace=True)
13 print(df_new.sample(10, random_state=seed))
14
15 ## Permanecendo apenas com as colunas dos materiais e protótipo ##
16 for i in df_new.columns:
17     print(i)
18     if i != 'Material' and i != 'prototype':
19         df_new.drop(i, axis=1, inplace=True)
20
21 print(df_new.sample(10))
22
23 df_materials = pd.DataFrame(columns=['Material', 'Atom1', 'Atom2', 'prototype'])
24
25 for i in df_new.index:
26     try:
27         formula_information = Composition(df_new['Material'][i]).as_dict()
28         lista = list(formula_information.items())
29         nova_entrada = [df_new['Material'][i], lista[0][0], lista[1][0],
30                         df_new['prototype'][i]]
31         df_materials.loc[len(df_materials)] = nova_entrada

```

```

32     except:
33         pass
34
35 print(df_materials.sample(10))
36
37 merge1 = pd.merge(df_materials, df_atoms, how='inner', left_on='Atom1',
38                    right_on='Element')
39 print(merge1)
40
41 merge2 = pd.merge(merge1, df_atoms, how='inner', left_on='Atom2',
42                    right_on='Element')
43 print(merge2)
44
45 # excluindo as colunas desnecessárias ##
46 df = merge2
47 df.drop(columns=['Atom1', 'Atom2', 'Element_x', 'Element_y'], inplace=
48 True)
49 print(df.head(10))
50
51 # exporta para um arquivo csv #
52 df.to_csv('AB2_new_atomic.csv')

```

A.6 Classificação geral

Abaixo consta o algoritmo para o treinamento da classificação com *dataset* geral e previsão das classes para os materiais novos com estequiometrias ABC_2 e ABC_4 . Os dois processos foram feitos no mesmo código. No entanto, é mais produtivo salvar o modelo final (em formato *pickle*, por exemplo) e utilizá-lo para previsões separadamente. Na próxima seção há o exemplo em que a previsão é feita à parte.

```

1 # coding=utf-8
2 def cabecalho(msg):
3     print(100*'=')
4     print(f'{msg:^100}')
5     print(100*'=')
6
7 # importar pacotes
8 from sklearn.model_selection import train_test_split
9 import matplotlib.pyplot as plt
10 import pandas as pd
11 import numpy as np

```

```
12
13 # Configura o número de linhas e colunas que serão mostrados, e o
14 # comprimento das colunas nos dataframes
15 #pd.set_option('display.max_columns', None)
16 #pd.set_option('display.max_rows', None)
17 #pd.set_option('max_colwidth', None)
18
19 # para conseguir repetir o experimento
20 seed = 50
21 np.random.seed(seed)
22
23 #importando o dataset como um dataframe
24 df = pd.read_csv('dataset_pbe_prototype_com_metais.csv')
25 df.drop('Unnamed: 0', axis='columns', inplace=True)
26 cabecalho('Dataset')
27 print(df.sample(10, random_state=seed))
28
29 ## Diz quantos grupos espaciais distintos existem ##
30 cabecalho('Quantidade de protótipos')
31 print(len(df['prototype'].unique()))
32
33 ## Há muitos grupos espaciais com poucos materiais. Vou trabalhar sem
34 # tirar eles por enquanto ##
35 cabecalho('Quantidade de cada protótipo')
36 print(df['prototype'].value_counts())
37
38 ## Excluindo linhas onde o protótipo é "Undefined" ##
39 df.drop(df.index[df['prototype'] == 'Undefined'], axis=0, inplace=True)
40
41 ## Atribuindo valores inteiros aos protótipos ##
42 df['prototype'] = df['prototype'].apply(lambda x : 2 if x == 'FeOCl'
43                                         else x)
44 df['prototype'] = df['prototype'].apply(lambda x : 3 if x == 'CdI2' else
45                                         x)
46 df['prototype'] = df['prototype'].apply(lambda x : 4 if x == 'MoS2' else
47                                         x)
48 df['prototype'] = df['prototype'].apply(lambda x : 5 if x == 'GeS2' else
49                                         x)
50 df['prototype'] = df['prototype'].apply(lambda x : 6 if x == 'CH' else x
51                                         )
52 df['prototype'] = df['prototype'].apply(lambda x : 7 if x == 'FeSe' else
53                                         x)
```

```
47 df['prototype'] = df['prototype'].apply(lambda x : 8 if x == 'GaS' else
48                                         x)
49 df['prototype'] = df['prototype'].apply(lambda x : 9 if x == 'BiTeI'
50                                         else x)
51 df['prototype'] = df['prototype'].apply(lambda x : 10 if x == 'MoSSe'
52                                         else x)
53 df['prototype'] = df['prototype'].apply(lambda x : 11 if x == 'WTe2'
54                                         else x)
55 df['prototype'] = df['prototype'].apply(lambda x : 12 if x == 'GaSe'
56                                         else x)
57 df['prototype'] = df['prototype'].apply(lambda x : 13 if x == 'MnPSe3'
58                                         else x)
59 df['prototype'] = df['prototype'].apply(lambda x : 14 if x == 'BiI3'
60                                         else x)
61 df['prototype'] = df['prototype'].apply(lambda x : 15 if x == 'PbSe'
62                                         else x)
63 df['prototype'] = df['prototype'].apply(lambda x : 16 if x == 'CrW3S8'
64                                         else x)
65 df['prototype'] = df['prototype'].apply(lambda x : 17 if x == 'Ti4C3O2'
66                                         else x)
67 df['prototype'] = df['prototype'].apply(lambda x : 18 if x == 'Ti3C2O2'
68                                         else x)
69 df['prototype'] = df['prototype'].apply(lambda x : 19 if x == 'Ti2CO2'
70                                         else x)
71 df['prototype'] = df['prototype'].apply(lambda x : 20 if x == 'PdS2'
72                                         else x)
73 df['prototype'] = df['prototype'].apply(lambda x : 21 if x == 'TiCl3'
74                                         else x)
75 df['prototype'] = df['prototype'].apply(lambda x : 22 if x == 'PbA2I4'
76                                         else x)
77 df['prototype'] = df['prototype'].apply(lambda x : 23 if x == 'NiSe'
78                                         else x)
79 df['prototype'] = df['prototype'].apply(lambda x : 24 if x == 'TiS3'
80                                         else x)
81 df['prototype'] = df['prototype'].apply(lambda x : 25 if x == 'AuSe'
82                                         else x)
83 df['prototype'] = df['prototype'].apply(lambda x : 26 if x == 'C3N' else
84                                         x)
85 df['prototype'] = df['prototype'].apply(lambda x : 27 if x == 'AgBr3'
86                                         else x)
87 df['prototype'] = df['prototype'].apply(lambda x : 28 if x == 'GeSe'
88                                         else x)
```

```
68 df['prototype'] = df['prototype'].apply(lambda x : 29 if x == 'Ti3C2H2O2'
69   ' else x)
70 df['prototype'] = df['prototype'].apply(lambda x : 30 if x == 'VTe3'
71   ' else x)
72 df['prototype'] = df['prototype'].apply(lambda x : 31 if x == 'Ti2CH2O2'
73   ' else x)
74 df['prototype'] = df['prototype'].apply(lambda x : 32 if x == 'HfBrS'
75   ' else x)
76 df['prototype'] = df['prototype'].apply(lambda x : 33 if x == 'Ti4C3H2O2'
77   ' else x)
78 df['prototype'] = df['prototype'].apply(lambda x : 34 if x == 'CrWS4'
79   ' else x)
80 df['prototype'] = df['prototype'].apply(lambda x : 35 if x == 'Ti4C3'
81   ' else x)
82 df['prototype'] = df['prototype'].apply(lambda x : 36 if x == 'Ti3C2'
83   ' else x)
84 df['prototype'] = df['prototype'].apply(lambda x : 37 if x == 'RhO' else
85   x)
86 df['prototype'] = df['prototype'].apply(lambda x : 38 if x == 'MnS2'
87   ' else x)
88 df['prototype'] = df['prototype'].apply(lambda x : 39 if x == 'BN' else
89   x)
90 df['prototype'] = df['prototype'].apply(lambda x : 40 if x == 'ReS2'
91   ' else x)
92 df['prototype'] = df['prototype'].apply(lambda x : 41 if x == 'ScPSe3'
93   ' else x)
94 df['prototype'] = df['prototype'].apply(lambda x : 42 if x == 'SnS' else
95   x)
96 df['prototype'] = df['prototype'].apply(lambda x : 43 if x == 'C' else x
97   )
98 df['prototype'] = df['prototype'].apply(lambda x : 44 if x == 'PbS' else
99   x)
100 df['prototype'] = df['prototype'].apply(lambda x : 45 if x == 'ISb' else
101   x)
102 df['prototype'] = df['prototype'].apply(lambda x : 46 if x == 'P' else x
103   )
104 df['prototype'] = df['prototype'].apply(lambda x : 47 if x == 'CH2Si'
105   ' else x)
106 df['prototype'] = df['prototype'].apply(lambda x : 48 if x == 'VPSe3'
107   ' else x)
108
109 cabecalho('Atribuindo números inteiros aos protótipos')
110 print(df['prototype'].value_counts())
```

```
91
92 ## exclusão dos protótipos com poucos representantes ##
93
94 for group in df['prototype']:
95     if (df['prototype'] == group).sum() < 10:
96         df.drop(df.index[df['prototype'] == group], axis=0, inplace=True
97 )
98
99 df['prototype'].value_counts()
100 cabecalho('Excluindo protótipos com poucos representantes')
101 print(df['prototype'].value_counts())
102
103 ## Contabilizando metais e isolantes antes de transformar o gap em números inteiros ##
104 n_metal = n_insulator = 0
105
106 for i in df['Band gap']:
107     if i > 0.000000001:
108         n_insulator += 1
109     else:
110         n_metal += 1
111 cabecalho('Número de metais e isolantes')
112 print(f' No de metais: {n_metal} \n No de isolantes: {n_insulator}')
113
114
115 cabecalho(' Atribuindo valores inteiros para o band gap')
116 ## Atribuindo valores inteiros para o band gap ##
117 df['Band gap'] = df['Band gap'].apply(lambda x : -1 if x < 0.000001 else
118 1)
119
120 print(df['Band gap'].value_counts())
121
122 cabecalho('Excluindo dados faltantes')
123 df.dropna(inplace=True)
124 print(df.isnull().sum())
125
126 cabecalho('Conjunto de treino e teste')
127 # separando em conjunto de treino e teste de forma estratificada depois
128 # de embaralhar pseudo aleatoriamente
129 train_set, test_set = train_test_split(df, stratify=df['Band gap'],
130                                         test_size=0.2, shuffle=True, random_state=seed)
131
132 print('treino \n ', train_set['Band gap'].value_counts())
133 print('teste \n ', test_set['Band gap'].value_counts())
```

```
129
130 ## Treinando modelos ##
131 X = train_set.drop(['Band gap', 'Material'], axis='columns')
132 y = train_set['Band gap']
133
134 cabecalho('Gradiente Boosting')
135 from sklearn.ensemble import GradientBoostingClassifier
136 from sklearn.model_selection import cross_val_predict
137 from sklearn import metrics
138
139 clf_gradboost = GradientBoostingClassifier(loss='deviance',
140                                              max_depth=3,
141                                              learning_rate=0.1,
142                                              n_estimators=100,
143                                              max_leaf_nodes=None)
144 y_pred = cross_val_predict(clf_gradboost, X, y, cv=5)
145
146 print(metrics.classification_report(y,y_pred, digits=3))
147
148 from sklearn.metrics import precision_recall_curve
149 ## calcular em termos de probabilidade ##
150 y_pred_prob = cross_val_predict(clf_gradboost, X, y, cv=5, method="predict_proba")
151
152 ## probabilidade de cada instância ser isolante (bandgap=1) ##
153 prob_insulator = y_pred_prob[:,1]
154
155 ## precision_recall ##
156 pre_grad, rec_grad, thr_grad = precision_recall_curve(y, prob_insulator)
157
158 cabecalho('RandomForestClassifier')
159 # Treinando com Random Forest
160 from sklearn.ensemble import RandomForestClassifier
161 y_pred = cross_val_predict(RandomForestClassifier(), X, y, cv=5)
162 print(metrics.classification_report(y, y_pred, digits=3))
163
164 ## calcular em termos de probabilidade ##
165 y_pred_prob = cross_val_predict(RandomForestClassifier(), X, y, cv=5,
166                                 method="predict_proba")
167 ## probabilidade de cada instância ser isolante (bandgap=1) ##
168 prob_insulator = y_pred_prob[:,1]
169
```

```
170 ## precision_recall ##
171 pre_forest, rec_forest, thr_forest = precision_recall_curve(y,
172     prob_insulator)
173
174 cabecalho('DecisionTreeClassifier')
175 from sklearn.tree import DecisionTreeClassifier
176 tree_clf = DecisionTreeClassifier(max_depth=None,
177                                     criterion='gini',
178                                     min_samples_split=2,
179                                     min_samples_leaf=1,
180                                     random_state=None)
181 y_pred = cross_val_predict(tree_clf, X, y, cv=5)
182 print(metrics.classification_report(y,y_pred, digits=3))
183
184 ## calcular em termos de probabilidade ##
185 y_pred_prob = cross_val_predict(tree_clf, X, y, cv=5, method="
186     predict_proba")
187
188 ## probabilidade de cada instância ser isolante (bandgap=1) ##
189 prob_insulator = y_pred_prob[:,1]
190
191 ## precision_recall ##
192 pre_tree, rec_tree, thr_tree = precision_recall_curve(y, prob_insulator)
193
194 cabecalho('AdaBoostClassifier')
195 from sklearn.ensemble import AdaBoostClassifier
196 clf_ada = AdaBoostClassifier(base_estimator=None,
197                               n_estimators=50,
198                               learning_rate=1.0,
199                               algorithm='SAMME.R',
200                               random_state=None)
201 y_pred = cross_val_predict(clf_ada, X.values, y, cv=5)
202 print(metrics.classification_report(y, y_pred, digits=3))
203
204 ## calcular em termos de probabilidade ##
205 y_pred_prob = cross_val_predict(clf_ada, X, y, cv=5, method="
206     predict_proba")
207
208 ## probabilidade de cada instância ser isolante (bandgap=1) ##
209 prob_insulator = y_pred_prob[:,1]
```

```
210 ## precision_recall ##
211 pre_ada, rec_ada, thr_ada = precision_recall_curve(y, prob_insulator)
212
213 ## plot da curva PR ##
214 cabecalho('Salvando curva precisionxrecall')
215
216 plt.plot(pre_grad, rec_grad, label="GradientBoosting", color='black')
217 plt.plot(pre_forest, rec_forest, label="RandomForest")
218 plt.plot(pre_tree, rec_tree, label="DecisionTree")
219 plt.plot(pre_ada, rec_ada, label="AdaBoost")
220 plt.legend(bbox_to_anchor=(1.05, 1.0), loc='upper left')
221 plt.ylabel('Precision')
222 plt.xlabel('Recall')
223 #plt.show()
224 plt.savefig("PR_curve.png", dpi=300, bbox_inches='tight')
225
226 cabecalho('Melhores parâmetrosRandomizedSearch')
227 from sklearn.model_selection import RandomizedSearchCV
228 from sklearn.metrics import average_precision_score, make_scorer
229 avg_prec = make_scorer(average_precision_score)
230
231 param_grid = {
232     "loss": ["deviance"],
233     "learning_rate": [0.02, 0.05, 0.1, 0.2, 0.3],
234     "min_samples_split": [2, 4, 8, 12, 16, 20],
235     "min_samples_leaf": [1, 2, 4, 8, 12, 16, 20],
236     "max_depth": [3, 5, 7, 8, 10],
237     "max_features": [None, 'sqrt'],
238     "criterion": ["friedman_mse"],
239     "subsample": [1.0],
240     "max_leaf_nodes": [8, 12, 16, 20, 24, 28, 32],
241     "n_estimators": [50, 60, 70, 80, 90, 100, 120, 140, 160, 165,
242                     170, 175, 180, 185, 190, 195, 200, 220, 240, 260, 270, 280, 300]
243     }
244 random_search = RandomizedSearchCV(GradientBoostingClassifier(),
245                                     param_grid,
246                                     cv=5,
247                                     scoring= avg_prec, # \C3rea sobre a
248                                     cuva precision x recall
249                                     refit=True,
250                                     n_iter=20,
251                                     n_jobs=None,
252                                     verbose=0,
```

```
251                                         pre_dispatch='2*n_jobs',
252                                         random_state=seed,
253                                         return_train_score=False)
254 random_search.fit(X,y)
255
256 print(random_search.best_params_)
257
258 cabecalho('Resultado de cada interacao precisao')
259 modelo = random_search.best_estimator_
260 print(random_search.cv_results_['mean_test_score'])
261
262 #cabecalho('Resultado de cada interacao recall')
263 #print(random_search.cv_results_['mean_test_recall'])
264
265 cabecalho('Performance no conjunto de teste')
266 testX = test_set.drop(['Band gap', 'Material'], axis='columns')
267 testY = test_set['Band gap']
268 result = modelo.predict(testX)
269 print(metrics.precision_score(testY, result))
270 print(metrics.recall_score(testY, result))
271 print(metrics.average_precision_score(testY, result))
272
273 ## new_data ##
274 df = pd.read_csv('dataset_new_prototipo.csv')
275 df.drop('Unnamed: 0', axis='columns', inplace=True)
276
277 # excluindo dados faltantes se houver #
278 df.dropna(inplace=True)
279
280 cabecalho('Novos materiais: atribuindo num aos prototipos')
281 ## atribuindo n\umeros inteiros aos prototipos ##
282
283 df['prototype'] = df['prototype'].apply(lambda x : 2 if x == 'FeOCl'
284                                         else x)
284 df['prototype'] = df['prototype'].apply(lambda x : 3 if x == 'CdI2' else
285                                         x)
285 df['prototype'] = df['prototype'].apply(lambda x : 4 if x == 'MoS2' else
286                                         x)
286 df['prototype'] = df['prototype'].apply(lambda x : 5 if x == 'GeS2' else
287                                         x)
287 df['prototype'] = df['prototype'].apply(lambda x : 6 if x == 'CH' else x
288                                         )
```

```
288 df['prototype'] = df['prototype'].apply(lambda x : 7 if x == 'FeSe' else
289                                         x)
290 df['prototype'] = df['prototype'].apply(lambda x : 8 if x == 'GaS' else
291                                         x)
292 df['prototype'] = df['prototype'].apply(lambda x : 9 if x == 'BiTeI'
293                                         else x)
294 df['prototype'] = df['prototype'].apply(lambda x : 10 if x == 'MoSSe'
295                                         else x)
296 df['prototype'] = df['prototype'].apply(lambda x : 11 if x == 'WTe2'
297                                         else x)
298 df['prototype'] = df['prototype'].apply(lambda x : 12 if x == 'GaSe'
299                                         else x)
300 df['prototype'] = df['prototype'].apply(lambda x : 13 if x == 'MnPSe3'
301                                         else x)
302 df['prototype'] = df['prototype'].apply(lambda x : 14 if x == 'BiI3'
303                                         else x)
304 df['prototype'] = df['prototype'].apply(lambda x : 15 if x == 'PbSe'
305                                         else x)
306 df['prototype'] = df['prototype'].apply(lambda x : 16 if x == 'CrW3S8'
307                                         else x)
308 df['prototype'] = df['prototype'].apply(lambda x : 17 if x == 'Ti4C3O2'
309                                         else x)
310 df['prototype'] = df['prototype'].apply(lambda x : 18 if x == 'Ti3C2O2'
311                                         else x)
312 df['prototype'] = df['prototype'].apply(lambda x : 19 if x == 'Ti2CO2'
313                                         else x)
314 df['prototype'] = df['prototype'].apply(lambda x : 20 if x == 'PdS2'
315                                         else x)
316 df['prototype'] = df['prototype'].apply(lambda x : 21 if x == 'TiCl3'
317                                         else x)
318 df['prototype'] = df['prototype'].apply(lambda x : 22 if x == 'PbA2I4'
319                                         else x)
320 df['prototype'] = df['prototype'].apply(lambda x : 23 if x == 'NiSe'
321                                         else x)
322 df['prototype'] = df['prototype'].apply(lambda x : 24 if x == 'TiS3'
323                                         else x)
324 df['prototype'] = df['prototype'].apply(lambda x : 25 if x == 'AuSe'
325                                         else x)
326 df['prototype'] = df['prototype'].apply(lambda x : 26 if x == 'C3N' else
327                                         x)
328 df['prototype'] = df['prototype'].apply(lambda x : 27 if x == 'AgBr3'
329                                         else x)
```

```
309 df['prototype'] = df['prototype'].apply(lambda x : 28 if x == 'GeSe'
310                                         else x)
310 df['prototype'] = df['prototype'].apply(lambda x : 29 if x == 'Ti3C2H2O2'
311                                         else x)
311 df['prototype'] = df['prototype'].apply(lambda x : 30 if x == 'VTe3'
312                                         else x)
312 df['prototype'] = df['prototype'].apply(lambda x : 31 if x == 'Ti2CH2O2'
313                                         else x)
313 df['prototype'] = df['prototype'].apply(lambda x : 32 if x == 'HfBrS'
314                                         else x)
314 df['prototype'] = df['prototype'].apply(lambda x : 33 if x == 'Ti4C3H2O2'
315                                         else x)
315 df['prototype'] = df['prototype'].apply(lambda x : 34 if x == 'CrWS4'
316                                         else x)
316 df['prototype'] = df['prototype'].apply(lambda x : 35 if x == 'Ti4C3'
317                                         else x)
317 df['prototype'] = df['prototype'].apply(lambda x : 36 if x == 'Ti3C2'
318                                         else x)
318 df['prototype'] = df['prototype'].apply(lambda x : 37 if x == 'RhO' else
319                                         x)
319 df['prototype'] = df['prototype'].apply(lambda x : 38 if x == 'MnS2'
320                                         else x)
320 df['prototype'] = df['prototype'].apply(lambda x : 39 if x == 'BN' else
321                                         x)
321 df['prototype'] = df['prototype'].apply(lambda x : 40 if x == 'ReS2'
322                                         else x)
322 df['prototype'] = df['prototype'].apply(lambda x : 41 if x == 'ScPSe3'
323                                         else x)
323 df['prototype'] = df['prototype'].apply(lambda x : 42 if x == 'SnS' else
324                                         x)
324 df['prototype'] = df['prototype'].apply(lambda x : 43 if x == 'C' else x
325                                         )
325 df['prototype'] = df['prototype'].apply(lambda x : 44 if x == 'PbS' else
326                                         x)
326 df['prototype'] = df['prototype'].apply(lambda x : 45 if x == 'ISb' else
327                                         x)
327 df['prototype'] = df['prototype'].apply(lambda x : 46 if x == 'P' else x
328                                         )
328 df['prototype'] = df['prototype'].apply(lambda x : 47 if x == 'CH2Si'
329                                         else x)
329 df['prototype'] = df['prototype'].apply(lambda x : 48 if x == 'VPSe3'
330                                         else x)
```

```

331 print(df['prototype'].value_counts())
332
333 ## informando quem são as features ##
334 X_new = df.drop(['Material'], axis='columns')
335 y_new = modelo.predict(X_new)
336 print(y_new)
337
338 ## Salvando data frame com os isolantes ##
339 cabecalho('Salvando o dataframe com os isolantes')
340
341 df_insulators = pd.DataFrame(columns=df.columns)
342 for i in range(0,800):
343     if y_new[i] == 1:
344         nova_entrada = list(df.loc[i])
345         df_insulators.loc[len(df_insulators)] = nova_entrada
346 print(df_insulators)
347
348
349 df_insulators.to_csv('dataset_pred_insulators.csv')

```

Para prever as classes dos novos materiais com estequiometria AB₂ e *features* estatísticas, basta modificar as últimas linhas do código anterior por:

```

1
2 ## new_data ##
3 df = pd.read_csv('dataset_new_prototipo.csv')
4 df.drop('Unnamed: 0', axis='columns', inplace=True)
5
6 # excluindo dados faltantes se houver #
7 df.dropna(inplace=True)
8
9 cabecalho('Novos materiais: atribuindo num aos protótipos')
10 ## atribuindo números inteiros aos protótipos ##
11
12 df['prototype'] = df['prototype'].apply(lambda x : 2 if x == 'FeOCl'
13                                         else x)
13 df['prototype'] = df['prototype'].apply(lambda x : 3 if x == 'CdI2' else
14                                         x)
14 df['prototype'] = df['prototype'].apply(lambda x : 4 if x == 'MoS2' else
15                                         x)
15 df['prototype'] = df['prototype'].apply(lambda x : 5 if x == 'GeS2' else
16                                         x)
16 df['prototype'] = df['prototype'].apply(lambda x : 6 if x == 'CH' else x)

```

```
    )
17 df['prototype'] = df['prototype'].apply(lambda x : 7 if x == 'FeSe' else
18                                         x)
18 df['prototype'] = df['prototype'].apply(lambda x : 8 if x == 'GaS' else
19                                         x)
19 df['prototype'] = df['prototype'].apply(lambda x : 9 if x == 'BiTeI'
20                                         else x)
20 df['prototype'] = df['prototype'].apply(lambda x : 10 if x == 'MoSSe'
21                                         else x)
21 df['prototype'] = df['prototype'].apply(lambda x : 11 if x == 'WTe2'
22                                         else x)
22 df['prototype'] = df['prototype'].apply(lambda x : 12 if x == 'GaSe'
23                                         else x)
23 df['prototype'] = df['prototype'].apply(lambda x : 13 if x == 'MnPSe3'
24                                         else x)
24 df['prototype'] = df['prototype'].apply(lambda x : 14 if x == 'BiI3'
25                                         else x)
25 df['prototype'] = df['prototype'].apply(lambda x : 15 if x == 'PbSe'
26                                         else x)
26 df['prototype'] = df['prototype'].apply(lambda x : 16 if x == 'CrW3S8'
27                                         else x)
27 df['prototype'] = df['prototype'].apply(lambda x : 17 if x == 'Ti4C3O2'
28                                         else x)
28 df['prototype'] = df['prototype'].apply(lambda x : 18 if x == 'Ti3C2O2'
29                                         else x)
29 df['prototype'] = df['prototype'].apply(lambda x : 19 if x == 'Ti2C02'
30                                         else x)
30 df['prototype'] = df['prototype'].apply(lambda x : 20 if x == 'PdS2'
31                                         else x)
31 df['prototype'] = df['prototype'].apply(lambda x : 21 if x == 'TiCl3'
32                                         else x)
32 df['prototype'] = df['prototype'].apply(lambda x : 22 if x == 'PbA2I4'
33                                         else x)
33 df['prototype'] = df['prototype'].apply(lambda x : 23 if x == 'NiSe'
34                                         else x)
34 df['prototype'] = df['prototype'].apply(lambda x : 24 if x == 'TiS3'
35                                         else x)
35 df['prototype'] = df['prototype'].apply(lambda x : 25 if x == 'AuSe'
36                                         else x)
36 df['prototype'] = df['prototype'].apply(lambda x : 26 if x == 'C3N' else
37                                         x)
37 df['prototype'] = df['prototype'].apply(lambda x : 27 if x == 'AgBr3'
                                         else x)
```

```
38 df['prototype'] = df['prototype'].apply(lambda x : 28 if x == 'GeSe' 
    else x)
39 df['prototype'] = df['prototype'].apply(lambda x : 29 if x == 'Ti3C2H2O2' 
    else x)
40 df['prototype'] = df['prototype'].apply(lambda x : 30 if x == 'VTe3' 
    else x)
41 df['prototype'] = df['prototype'].apply(lambda x : 31 if x == 'Ti2CH2O2' 
    else x)
42 df['prototype'] = df['prototype'].apply(lambda x : 32 if x == 'HfBrS' 
    else x)
43 df['prototype'] = df['prototype'].apply(lambda x : 33 if x == 'Ti4C3H2O2' 
    else x)
44 df['prototype'] = df['prototype'].apply(lambda x : 34 if x == 'CrWS4' 
    else x)
45 df['prototype'] = df['prototype'].apply(lambda x : 35 if x == 'Ti4C3' 
    else x)
46 df['prototype'] = df['prototype'].apply(lambda x : 36 if x == 'Ti3C2' 
    else x)
47 df['prototype'] = df['prototype'].apply(lambda x : 37 if x == 'RhO' else 
    x)
48 df['prototype'] = df['prototype'].apply(lambda x : 38 if x == 'MnS2' 
    else x)
49 df['prototype'] = df['prototype'].apply(lambda x : 39 if x == 'BN' else 
    x)
50 df['prototype'] = df['prototype'].apply(lambda x : 40 if x == 'ReS2' 
    else x)
51 df['prototype'] = df['prototype'].apply(lambda x : 41 if x == 'ScPSe3' 
    else x)
52 df['prototype'] = df['prototype'].apply(lambda x : 42 if x == 'SnS' else 
    x)
53 df['prototype'] = df['prototype'].apply(lambda x : 43 if x == 'C' else x 
    )
54 df['prototype'] = df['prototype'].apply(lambda x : 44 if x == 'PbS' else 
    x)
55 df['prototype'] = df['prototype'].apply(lambda x : 45 if x == 'ISb' else 
    x)
56 df['prototype'] = df['prototype'].apply(lambda x : 46 if x == 'P' else x 
    )
57 df['prototype'] = df['prototype'].apply(lambda x : 47 if x == 'CH2Si' 
    else x)
58 df['prototype'] = df['prototype'].apply(lambda x : 48 if x == 'VPSe3' 
    else x)
```

```

60 print(df['prototype'].value_counts())
61
62 ## informando quem são as features ##
63 X_new = df.drop(['Material'], axis='columns')
64 y_new = modelo.predict(X_new)
65 print(y_new)
66
67 ## Salvando data frame com os isolantes ##
68 cabecalho('Salvando o dataframe com os isolantes')
69
70 df_insulators = pd.DataFrame(columns=df.columns)
71 for i in range(0,840):
72     if y_new[i] == 1:
73         nova_entrada = list(df.loc[i])
74         df_insulators.loc[len(df_insulators)] = nova_entrada
75 print(df_insulators)
76
77
78 df_insulators.to_csv('dataset_pred_insulators.csv')

```

A.7 Classificação AB₂

Segue o algoritmo para treinamento da classificação com materiais AB₂ da base de dados C2DB e *features* atômicas.

```

1 # coding=utf-8
2 def cabecalho(msg):
3     print(100*'=')
4     print(f'{msg:^100}')
5     print(100*'=')
6
7 # importar pacotes
8 from sklearn.model_selection import train_test_split
9 import matplotlib.pyplot as plt
10 import pandas as pd
11 import numpy as np
12
13 # Configura o número de linhas e colunas que serão mostrados, e o
14 # comprimento das colunas nos dataframes
14 #pd.set_option('display.max_columns', None)
15 #pd.set_option('display.max_rows', None)
16 #pd.set_option('max_colwidth', None)

```

```
17
18 # para conseguir repetir o experimento
19 seed = 50
20 np.random.seed(seed)
21
22 #importando o dataset como um dataframe
23 df = pd.read_csv('AB2_prototype_atomic.csv')
24 df.drop('Unnamed: 0', axis='columns', inplace=True)
25 cabecalho('Dataset')
26 print(df.sample(10, random_state=seed))
27
28 ## Diz quantos grupos espaciais distintos existem ##
29 cabecalho('Quantidade de protótipos')
30 print(len(df['prototype'].unique()))
31
32 ## Há muitos grupos espaciais com poucos materiais. Vou trabalhar sem
33     tirar eles por enquanto ##
34 cabecalho('Quantidade de cada protótipo')
35 print(df['prototype'].value_counts())
36
37 ## Excluindo linhas onde o protótipo é "Undefined" ##
38 df.drop(df.index[df['prototype'] == 'Undefined'], axis=0, inplace=True)
39
40 ## Atribuindo valores inteiros aos protótipos ##
41 df['prototype'] = df['prototype'].apply(lambda x : 2 if x == 'FeOCl'
42                                         else x)
43 df['prototype'] = df['prototype'].apply(lambda x : 3 if x == 'CdI2' else
44                                         x)
45 df['prototype'] = df['prototype'].apply(lambda x : 4 if x == 'MoS2' else
46                                         x)
47 df['prototype'] = df['prototype'].apply(lambda x : 5 if x == 'GeS2' else
48                                         x)
49 df['prototype'] = df['prototype'].apply(lambda x : 6 if x == 'CH' else x)
```

```
50 df['prototype'] = df['prototype'].apply(lambda x : 11 if x == 'WTe2'
51   else x)
52 df['prototype'] = df['prototype'].apply(lambda x : 12 if x == 'GaSe'
53   else x)
54 df['prototype'] = df['prototype'].apply(lambda x : 13 if x == 'MnPSe3'
55   else x)
56 df['prototype'] = df['prototype'].apply(lambda x : 14 if x == 'BiI3'
57   else x)
58 df['prototype'] = df['prototype'].apply(lambda x : 15 if x == 'PbSe'
59   else x)
60 df['prototype'] = df['prototype'].apply(lambda x : 16 if x == 'CrW3S8'
61   else x)
62 df['prototype'] = df['prototype'].apply(lambda x : 17 if x == 'Ti4C3O2'
63   else x)
64 df['prototype'] = df['prototype'].apply(lambda x : 18 if x == 'Ti3C2O2'
65   else x)
66 df['prototype'] = df['prototype'].apply(lambda x : 19 if x == 'Ti2CO2'
67   else x)
68 df['prototype'] = df['prototype'].apply(lambda x : 20 if x == 'PdS2'
69   else x)
70 df['prototype'] = df['prototype'].apply(lambda x : 21 if x == 'TiCl3'
71   else x)
72 df['prototype'] = df['prototype'].apply(lambda x : 22 if x == 'PbA2I4'
73   else x)
74 df['prototype'] = df['prototype'].apply(lambda x : 23 if x == 'NiSe'
75   else x)
76 df['prototype'] = df['prototype'].apply(lambda x : 24 if x == 'TiS3'
77   else x)
78 df['prototype'] = df['prototype'].apply(lambda x : 25 if x == 'AuSe'
79   else x)
80 df['prototype'] = df['prototype'].apply(lambda x : 26 if x == 'C3N' else
81   x)
82 df['prototype'] = df['prototype'].apply(lambda x : 27 if x == 'AgBr3'
83   else x)
84 df['prototype'] = df['prototype'].apply(lambda x : 28 if x == 'GeSe'
85   else x)
86 df['prototype'] = df['prototype'].apply(lambda x : 29 if x == 'Ti3C2H2O2'
87   else x)
88 df['prototype'] = df['prototype'].apply(lambda x : 30 if x == 'VTe3'
89   else x)
90 df['prototype'] = df['prototype'].apply(lambda x : 31 if x == 'Ti2CH2O2'
91   else x)
```

```
71 df['prototype'] = df['prototype'].apply(lambda x : 32 if x == 'HfBrS' else x)
72 df['prototype'] = df['prototype'].apply(lambda x : 33 if x == 'Ti4C3H2O2' else x)
73 df['prototype'] = df['prototype'].apply(lambda x : 34 if x == 'CrWS4' else x)
74 df['prototype'] = df['prototype'].apply(lambda x : 35 if x == 'Ti4C3' else x)
75 df['prototype'] = df['prototype'].apply(lambda x : 36 if x == 'Ti3C2' else x)
76 df['prototype'] = df['prototype'].apply(lambda x : 37 if x == 'RhO' else x)
77 df['prototype'] = df['prototype'].apply(lambda x : 38 if x == 'MnS2' else x)
78 df['prototype'] = df['prototype'].apply(lambda x : 39 if x == 'BN' else x)
79 df['prototype'] = df['prototype'].apply(lambda x : 40 if x == 'ReS2' else x)
80 df['prototype'] = df['prototype'].apply(lambda x : 41 if x == 'ScPSe3' else x)
81 df['prototype'] = df['prototype'].apply(lambda x : 42 if x == 'SnS' else x)
82 df['prototype'] = df['prototype'].apply(lambda x : 43 if x == 'C' else x)
83 df['prototype'] = df['prototype'].apply(lambda x : 44 if x == 'PbS' else x)
84 df['prototype'] = df['prototype'].apply(lambda x : 45 if x == 'ISb' else x)
85 df['prototype'] = df['prototype'].apply(lambda x : 46 if x == 'P' else x)
86 df['prototype'] = df['prototype'].apply(lambda x : 47 if x == 'CH2Si' else x)
87 df['prototype'] = df['prototype'].apply(lambda x : 48 if x == 'VPSe3' else x)
88
89 cabecalho('Atribuindo números inteiros aos protótipos')
90 print(df['prototype'].value_counts())
91
92 ## exclusão dos protótipos com poucos representantes ##
93
94 for group in df['prototype']:
95     if (df['prototype'] == group).sum() < 10:
```

```
96         df.drop(df.index[df['prototype'] == group], axis=0, inplace=True
97     )
98
99 df['prototype'].value_counts()
100 cabecalho('Excluindo protótipos com poucos representantes')
101 print(df['prototype'].value_counts())
102
103 ## Contabilizando metais e isolantes antes de transformar o gap em números inteiros ##
104 n_metal = n_insulator = 0
105
106 for i in df['Band gap']:
107     if i > 0.000000001:
108         n_insulator += 1
109     else:
110         n_metal += 1
111 cabecalho('Número de metais e isolantes')
112 print(f' No de metais: {n_metal} \n No de isolantes: {n_insulator}')
113
114
115 cabecalho(' Atribuindo valores inteiros para o band gap')
116 ## Atribuindo valores inteiros para o band gap ##
117 df['Band gap'] = df['Band gap'].apply(lambda x : -1 if x < 0.000001 else
118                                         1)
119 print(df['Band gap'].value_counts())
120
121 cabecalho('Excluindo dados faltantes')
122 df.dropna(inplace=True)
123 print(df.isnull().sum())
124
125 cabecalho('Conjunto de treino e teste')
126 # separando em conjunto de treino e teste de forma estratificada depois
127 # de embaralhar pseudo aleatoriamente
128 train_set, test_set = train_test_split(df, stratify=df['Band gap'],
129                                         test_size=0.2, shuffle=True, random_state=seed)
130 print('treino \n ', train_set['Band gap'].value_counts())
131 print('teste \n ', test_set['Band gap'].value_counts())
132
133 ## Treinando modelos ##
134 X = train_set.drop(['Band gap', 'Material'], axis='columns')
135 y = train_set['Band gap']
```

```
134 cabecalho('Gradiente Boosting')
135 from sklearn.ensemble import GradientBoostingClassifier
136 from sklearn.model_selection import cross_val_predict
137 from sklearn import metrics
138
139 clf_gradboost = GradientBoostingClassifier(loss='deviance',
140                                              max_depth=3,
141                                              learning_rate=0.1,
142                                              n_estimators=100,
143                                              max_leaf_nodes=None)
144 y_pred = cross_val_predict(clf_gradboost, X, y, cv=5)
145
146 print(metrics.classification_report(y,y_pred, digits=3))
147
148 from sklearn.metrics import precision_recall_curve
149 ## calcular em termos de probabilidade ##
150 y_pred_prob = cross_val_predict(clf_gradboost, X, y, cv=5, method="
151     predict_proba")
152 ## probabilidade de cada instância ser isolante (bandgap=1) ##
153 prob_insulator = y_pred_prob[:,1]
154
155 ## precision_recall ##
156 pre_grad, rec_grad, thr_grad = precision_recall_curve(y, prob_insulator)
157
158
159 cabecalho('RandomForestClassifier')
160 # Treinando com Random Forest
161 from sklearn.ensemble import RandomForestClassifier
162 y_pred = cross_val_predict(RandomForestClassifier(), X, y, cv=5)
163 print(metrics.classification_report(y, y_pred, digits=3))
164
165 ## calcular em termos de probabilidade ##
166 y_pred_prob = cross_val_predict(RandomForestClassifier(), X, y, cv=5,
167                                 method="predict_proba")
168 ## probabilidade de cada instância ser isolante (bandgap=1) ##
169 prob_insulator = y_pred_prob[:,1]
170
171 ## precision_recall ##
172 pre_forest, rec_forest, thr_forest = precision_recall_curve(y,
173                  prob_insulator)
174
```

```
174
175 cabecalho('DecisionTreeClassifier')
176 from sklearn.tree import DecisionTreeClassifier
177 tree_clf = DecisionTreeClassifier(max_depth=None,
178                                     criterion='gini',
179                                     min_samples_split=2,
180                                     min_samples_leaf=1,
181                                     random_state=None)
182 y_pred = cross_val_predict(tree_clf, X, y, cv=5)
183 print(metrics.classification_report(y,y_pred, digits=3))
184
185 ## calcular em termos de probabilidade ##
186 y_pred_prob = cross_val_predict(tree_clf, X, y, cv=5, method="
187     predict_proba")
188
189 ## probabilidade de cada instância ser isolante (bandgap=1) ##
190 prob_insulator = y_pred_prob[:,1]
191
192 ## precision_recall ##
193 pre_tree, rec_tree, thr_tree = precision_recall_curve(y, prob_insulator)
194
195
196 cabecalho('AdaBoostClassifier')
197 from sklearn.ensemble import AdaBoostClassifier
198 clf_ada = AdaBoostClassifier(base_estimator=None,
199                               n_estimators=50,
200                               learning_rate=1.0,
201                               algorithm='SAMME.R',
202                               random_state=None)
203 y_pred = cross_val_predict(clf_ada, X.values, y, cv=5)
204 print(metrics.classification_report(y, y_pred, digits=3))
205
206 ## calcular em termos de probabilidade ##
207 y_pred_prob = cross_val_predict(clf_ada, X, y, cv=5, method="
208     predict_proba")
209
210 ## probabilidade de cada instância ser isolante (bandgap=1) ##
211 prob_insulator = y_pred_prob[:,1]
212
213 ## precision_recall ##
214 pre_ada, rec_ada, thr_ada = precision_recall_curve(y, prob_insulator)
```



```

255                                     random_state=seed,
256                                     return_train_score=False)
257 random_search.fit(X,y)
258
259 print(random_search.best_params_)
260
261 cabecalho('Resultado de cada interacao precisao')
262 modelo = random_search.best_estimator_
263 print(random_search.cv_results_['mean_test_score'])
264
265 #cabecalho('Resultado de cada interacao recall')
266 #print(random_search.cv_results_['mean_test_recall'])
267
268 cabecalho('Performance no conjunto de teste')
269 testX = test_set.drop(['Band gap', 'Material'], axis='columns')
270 testY = test_set['Band gap']
271 result = modelo.predict(testX)
272 print(metrics.precision_score(testY, result))
273 print(metrics.recall_score(testY, result))
274 print(metrics.average_precision_score(testY, result))
275
276 cabecalho('Salvando modelo')
277
278 import pickle
279 with open('gradboost_class_ab2_atomic.pkl', 'wb') as file:
280     pickle.dump(modelo, file)

```

No código abaixo encontra-se a previsão das classes para os novos materiais de estequiometria AB₂ e *features* atômicas.

```

1 # coding=utf-8
2 def cabecalho(msg):
3     print(100*'=')
4     print(f'{msg:^100}')
5     print(100*'=')
6
7 import pickle
8 import pandas as pd
9
10 with open('gradboost_class_ab2_atomic.pkl', 'rb') as f:
11     model_class = pickle.load(f)
12
13 ## new_data ##

```

```
14 df = pd.read_csv('AB2_new_atomic.csv')
15 df.drop('Unnamed: 0', axis='columns', inplace=True)
16
17 # excluindo dados faltantes se houver #
18 df.dropna(inplace=True)
19
20 cabecalho('Novos materiais: atribuindo num aos protótipos')
21 ## atribuindo n\úmeros inteiros aos protótipos ##
22
23 df['prototype'] = df['prototype'].apply(lambda x : 2 if x == 'FeOCl'
24                                         else x)
24 df['prototype'] = df['prototype'].apply(lambda x : 3 if x == 'CdI2' else
25                                         x)
25 df['prototype'] = df['prototype'].apply(lambda x : 4 if x == 'MoS2' else
26                                         x)
26 df['prototype'] = df['prototype'].apply(lambda x : 5 if x == 'GeS2' else
27                                         x)
27 df['prototype'] = df['prototype'].apply(lambda x : 6 if x == 'CH' else x
28                                         )
28 df['prototype'] = df['prototype'].apply(lambda x : 7 if x == 'FeSe' else
29                                         x)
29 df['prototype'] = df['prototype'].apply(lambda x : 8 if x == 'GaS' else
30                                         x)
30 df['prototype'] = df['prototype'].apply(lambda x : 9 if x == 'BiTeI'
31                                         else x)
31 df['prototype'] = df['prototype'].apply(lambda x : 10 if x == 'MoSSe'
32                                         else x)
32 df['prototype'] = df['prototype'].apply(lambda x : 11 if x == 'WTe2'
33                                         else x)
33 df['prototype'] = df['prototype'].apply(lambda x : 12 if x == 'GaSe'
34                                         else x)
34 df['prototype'] = df['prototype'].apply(lambda x : 13 if x == 'MnPSe3'
35                                         else x)
35 df['prototype'] = df['prototype'].apply(lambda x : 14 if x == 'BiI3'
36                                         else x)
36 df['prototype'] = df['prototype'].apply(lambda x : 15 if x == 'PbSe'
37                                         else x)
37 df['prototype'] = df['prototype'].apply(lambda x : 16 if x == 'CrW3S8'
38                                         else x)
38 df['prototype'] = df['prototype'].apply(lambda x : 17 if x == 'Ti4C3O2'
39                                         else x)
39 df['prototype'] = df['prototype'].apply(lambda x : 18 if x == 'Ti3C2O2'
40                                         else x)
```

```
40 df['prototype'] = df['prototype'].apply(lambda x : 19 if x == 'Ti2CO2' 
   else x)
41 df['prototype'] = df['prototype'].apply(lambda x : 20 if x == 'PdS2' 
   else x)
42 df['prototype'] = df['prototype'].apply(lambda x : 21 if x == 'TiCl3' 
   else x)
43 df['prototype'] = df['prototype'].apply(lambda x : 22 if x == 'PbA2I4' 
   else x)
44 df['prototype'] = df['prototype'].apply(lambda x : 23 if x == 'NiSe' 
   else x)
45 df['prototype'] = df['prototype'].apply(lambda x : 24 if x == 'TiS3' 
   else x)
46 df['prototype'] = df['prototype'].apply(lambda x : 25 if x == 'AuSe' 
   else x)
47 df['prototype'] = df['prototype'].apply(lambda x : 26 if x == 'C3N' else 
   x)
48 df['prototype'] = df['prototype'].apply(lambda x : 27 if x == 'AgBr3' 
   else x)
49 df['prototype'] = df['prototype'].apply(lambda x : 28 if x == 'GeSe' 
   else x)
50 df['prototype'] = df['prototype'].apply(lambda x : 29 if x == 'Ti3C2H2O2' 
   else x)
51 df['prototype'] = df['prototype'].apply(lambda x : 30 if x == 'VTe3' 
   else x)
52 df['prototype'] = df['prototype'].apply(lambda x : 31 if x == 'Ti2CH2O2' 
   else x)
53 df['prototype'] = df['prototype'].apply(lambda x : 32 if x == 'HfBrS' 
   else x)
54 df['prototype'] = df['prototype'].apply(lambda x : 33 if x == 'Ti4C3H2O2' 
   else x)
55 df['prototype'] = df['prototype'].apply(lambda x : 34 if x == 'CrWS4' 
   else x)
56 df['prototype'] = df['prototype'].apply(lambda x : 35 if x == 'Ti4C3' 
   else x)
57 df['prototype'] = df['prototype'].apply(lambda x : 36 if x == 'Ti3C2' 
   else x)
58 df['prototype'] = df['prototype'].apply(lambda x : 37 if x == 'RhO' else 
   x)
59 df['prototype'] = df['prototype'].apply(lambda x : 38 if x == 'MnS2' 
   else x)
60 df['prototype'] = df['prototype'].apply(lambda x : 39 if x == 'BN' else 
   x)
61 df['prototype'] = df['prototype'].apply(lambda x : 40 if x == 'ReS2'
```

```
    else x)
62 df['prototype'] = df['prototype'].apply(lambda x : 41 if x == 'ScPSe3'
    else x)
63 df['prototype'] = df['prototype'].apply(lambda x : 42 if x == 'SnS' else
    x)
64 df['prototype'] = df['prototype'].apply(lambda x : 43 if x == 'C' else x
    )
65 df['prototype'] = df['prototype'].apply(lambda x : 44 if x == 'PbS' else
    x)
66 df['prototype'] = df['prototype'].apply(lambda x : 45 if x == 'ISb' else
    x)
67 df['prototype'] = df['prototype'].apply(lambda x : 46 if x == 'P' else x
    )
68 df['prototype'] = df['prototype'].apply(lambda x : 47 if x == 'CH2Si'
    else x)
69 df['prototype'] = df['prototype'].apply(lambda x : 48 if x == 'VPSe3'
    else x)
70
71 print(df['prototype'].value_counts())
72
73 ## informando quem são as features ##
74 X_new = df.drop(['Material'], axis='columns')
75 y_new = model_class.predict(X_new)
76 print(y_new)
77
78 ## Salvando data frame com os isolantes ##
79 cabecalho('Salvando o dataframe com os isolantes')
80
81 df_insulators = pd.DataFrame(columns=df.columns)
82 for i in range(0,840):
83     if y_new[i] == 1:
84         nova_entrada = list(df.loc[i])
85         df_insulators.loc[len(df_insulators)] = nova_entrada
86 print(df_insulators)
87
88
89 df_insulators.to_csv('dataset_pred_insulators.csv')
```

A.8 Regressão geral

A seguir está disponível o algoritmo para treinamento da regressão com o *dataset* geral e previsão do gap para os materiais novos com estequiometria ABC₂ e ABC₄.

```
1 # coding=utf-8
2
3 def cabecalho(msg):
4     print(100*'=')
5     print(f'{msg:^100}')
6     print(100*'=')
7
8 # importar pacotes
9 from sklearn.model_selection import train_test_split
10 import matplotlib.pyplot as plt
11 import pandas as pd
12 import numpy as np
13
14 # Configura o número de linhas e colunas que serão mostrados, e o
15 # comprimento das colunas nos dataframes
16 pd.set_option('display.max_columns', None)
17 pd.set_option('display.max_rows', None)
18 pd.set_option('max_colwidth', None)
19
20 # para conseguir repetir o experimento
21 seed = 50
22 np.random.seed(seed)
23
24 # carregando dataset
25 cabecalho('Dataset')
26 df = pd.read_csv('dataset_pbe_prototype_isolantes.csv')
27 df.drop('Unnamed: 0', axis='columns', inplace=True)
28 print(df.head())
29
30 cabecalho('Tamanho do dataset')
31 print(df.shape)
32
33 cabecalho('Quantidade de protótipos')
34 print(len(df['prototype'].unique()))
35
36 cabecalho('Quantidade de cada protótipo')
37 print(df['prototype'].value_counts())
```

```
38 ## Excluindo linhas onde o protótipo é "Unfined" ##
39 df.drop(df.index[df['prototype'] == 'Undefined'], axis=0, inplace=True)
40
41 ## Atribuindo valores inteiros aos protótipos ##
42 cabecalho('Atribuindo numeros inteiros aos protótipos')
43 df['prototype'] = df['prototype'].apply(lambda x : 2 if x == 'FeOCl'
44                                         else x)
44 df['prototype'] = df['prototype'].apply(lambda x : 3 if x == 'CdI2' else
45                                         x)
45 df['prototype'] = df['prototype'].apply(lambda x : 4 if x == 'MoS2' else
46                                         x)
46 df['prototype'] = df['prototype'].apply(lambda x : 5 if x == 'GeS2' else
47                                         x)
47 df['prototype'] = df['prototype'].apply(lambda x : 6 if x == 'CH' else x
48                                         )
48 df['prototype'] = df['prototype'].apply(lambda x : 7 if x == 'FeSe' else
49                                         x)
49 df['prototype'] = df['prototype'].apply(lambda x : 8 if x == 'GaS' else
50                                         x)
50 df['prototype'] = df['prototype'].apply(lambda x : 9 if x == 'BiTeI'
51                                         else x)
51 df['prototype'] = df['prototype'].apply(lambda x : 10 if x == 'MoSSe'
52                                         else x)
52 df['prototype'] = df['prototype'].apply(lambda x : 11 if x == 'WTe2'
53                                         else x)
53 df['prototype'] = df['prototype'].apply(lambda x : 12 if x == 'GaSe'
54                                         else x)
54 df['prototype'] = df['prototype'].apply(lambda x : 13 if x == 'MnPSe3'
55                                         else x)
55 df['prototype'] = df['prototype'].apply(lambda x : 14 if x == 'BiI3'
56                                         else x)
56 df['prototype'] = df['prototype'].apply(lambda x : 15 if x == 'PbSe'
57                                         else x)
57 df['prototype'] = df['prototype'].apply(lambda x : 16 if x == 'CrW3S8'
58                                         else x)
58 df['prototype'] = df['prototype'].apply(lambda x : 17 if x == 'Ti4C3O2'
59                                         else x)
59 df['prototype'] = df['prototype'].apply(lambda x : 18 if x == 'Ti3C2O2'
60                                         else x)
60 df['prototype'] = df['prototype'].apply(lambda x : 19 if x == 'Ti2C02'
61                                         else x)
61 df['prototype'] = df['prototype'].apply(lambda x : 20 if x == 'PdS2'
62                                         else x)
```

```
62 df['prototype'] = df['prototype'].apply(lambda x : 21 if x == 'TiCl3'  
63     else x)  
64 df['prototype'] = df['prototype'].apply(lambda x : 22 if x == 'PbA2I4'  
65     else x)  
66 df['prototype'] = df['prototype'].apply(lambda x : 23 if x == 'NiSe'  
67     else x)  
68 df['prototype'] = df['prototype'].apply(lambda x : 24 if x == 'TiS3'  
69     else x)  
70 df['prototype'] = df['prototype'].apply(lambda x : 25 if x == 'AuSe'  
71     else x)  
72 df['prototype'] = df['prototype'].apply(lambda x : 26 if x == 'C3N' else  
73     x)  
74 df['prototype'] = df['prototype'].apply(lambda x : 27 if x == 'AgBr3'  
75     else x)  
76 df['prototype'] = df['prototype'].apply(lambda x : 28 if x == 'GeSe'  
77     else x)  
78 df['prototype'] = df['prototype'].apply(lambda x : 29 if x == 'Ti3C2H2O2'  
79     else x)  
80 df['prototype'] = df['prototype'].apply(lambda x : 30 if x == 'VTe3'  
81     else x)  
82 df['prototype'] = df['prototype'].apply(lambda x : 31 if x == 'Ti2CH2O2'  
83     else x)  
84 df['prototype'] = df['prototype'].apply(lambda x : 32 if x == 'HfBrS'  
85     else x)  
86 df['prototype'] = df['prototype'].apply(lambda x : 33 if x == 'Ti4C3H2O2'  
87     else x)  
88 df['prototype'] = df['prototype'].apply(lambda x : 34 if x == 'CrWS4'  
89     else x)  
90 df['prototype'] = df['prototype'].apply(lambda x : 35 if x == 'Ti4C3'  
91     else x)  
92 df['prototype'] = df['prototype'].apply(lambda x : 36 if x == 'Ti3C2'  
93     else x)  
94 df['prototype'] = df['prototype'].apply(lambda x : 37 if x == 'RhO' else  
95     x)  
96 df['prototype'] = df['prototype'].apply(lambda x : 38 if x == 'MnS2'  
97     else x)  
98 df['prototype'] = df['prototype'].apply(lambda x : 39 if x == 'BN' else  
99     x)  
100 df['prototype'] = df['prototype'].apply(lambda x : 40 if x == 'ReS2'  
101     else x)  
102 df['prototype'] = df['prototype'].apply(lambda x : 41 if x == 'ScPSe3'  
103     else x)
```

```
83 df['prototype'] = df['prototype'].apply(lambda x : 42 if x == 'SnS' else
84     x)
85 df['prototype'] = df['prototype'].apply(lambda x : 43 if x == 'C' else x
86 )
87 df['prototype'] = df['prototype'].apply(lambda x : 44 if x == 'PbS' else
88     x)
89 df['prototype'] = df['prototype'].apply(lambda x : 45 if x == 'ISb' else
90     x)
91 df['prototype'] = df['prototype'].apply(lambda x : 46 if x == 'P' else x
92 )
93 df['prototype'] = df['prototype'].apply(lambda x : 47 if x == 'CH2Si'
94     else x)
95 df['prototype'] = df['prototype'].apply(lambda x : 48 if x == 'VPSe3'
96     else x)
97
98 print(df['prototype'].value_counts())
99
100 cabecalho('Excluindo protótipos com poucos representantes')
101 for group in df['prototype']:
102     if (df['prototype'] == group).sum() < 6:
103         df.drop(df.index[df['prototype'] == group], axis=0, inplace=True
104     )
105
106 print(df['prototype'].value_counts())
107
108 cabecalho('Número de metais e isolantes com o funcionalpbe')
109 n_metal = n_insulator = 0
110
111 for i in df['Band gap']:
112     if i > 0.0000000001:
113         n_insulator += 1
114     else:
115         n_metal += 1
116
117 print(f' No de metais: {n_metal} \n No de isolantes: {n_insulator}')
118
119 cabecalho('Min e max do gap')
120 print('gap min', df['Band gap'].min())
121 print('gap max', df['Band gap'].max())
122
123 cabecalho('Intervalos de gap')
124 pri = seg = ter = qua = qui = sex = seti = oit = non = 0
125 for i in df['Band gap']:
```

```
118     if i > 0.00001 and i <= 1.0:
119         pri = pri + 1
120     if i > 1.0 and i <= 2.0:
121         seg = seg + 1
122     if i > 2.0 and i <= 3.0:
123         ter = ter + 1
124     if i > 3.0 and i <= 4.0:
125         qua = qua + 1
126     if i > 4.0 and i <= 5.0:
127         qui = qui + 1
128     if i > 5.0 and i <= 6.0:
129         sex = sex + 1
130     if i > 6.0 and i <= 7.0:
131         seti = seti + 1
132     if i > 7.0 and i <= 8.0:
133         oit = oit + 1
134     if i > 8.0 and i <= 9.0:
135         non = non + 1
136
137 print(f'''Quant. de materiais com gap entre 0 e 1: {pri}
138 Quant. de materiais com gap entre 1 e 2: {seg}
139 Quant. de materiais com gap entre 2 e 3: {ter}
140 Quant. de materiais com gap entre 3 e 4: {qua}
141 Quant. de materiais com gap entre 4 e 5: {qui}
142 Quant. de materiais com gap entre 5 e 6: {sex}
143 Quant. de materiais com gap entre 6 e 7: {seti}
144 Quant. de materiais com gap entre 7 e 8: {oit}
145 Quant. de materiais com gap entre 8 e 9: {non}''')
146 print('Soma dos materiais com todos os gaps:', np.sum([pri, seg, ter, qua
    , qui, sex, seti, oit, non]))
147
148 cabecalho('Salvando histograma ...')
149 gap = df['Band gap'].copy()
150 plt.hist(gap, bins=[0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0],
    density=False, label='Data', rwidth=0.9)
151 plt.savefig('hist_gap_pbe.png', dpi=300)
152
153 cabecalho('Procurando dados faltantes')
154 print(df.isnull().sum())
155
156 cabecalho('Excluindo dados faltantes')
157 df.dropna(inplace=True)
158 print(df.isnull().sum())
```

```
159
160 cabecalho('Tamanho do treino e teste')
161 # separando em conjunto de treino e teste de forma estratificada depois
162 # de embaralhar pseudo aleatoriamente
163 train_set, test_set = train_test_split(df, test_size=0.2, shuffle=True,
164                                         random_state=seed)
165
166 print('treino: ', train_set.shape)
167 print('teste: ', test_set.shape)
168
169
170 ## Treinando modelos ##
171 X = train_set.drop(['Band gap', 'Material'], axis='columns')
172 y = train_set['Band gap']
173
174 from sklearn import metrics
175 from sklearn.model_selection import cross_val_predict, cross_val_score,
176                                         cross_validate
177
178 cabecalho('Random Forest')
179 from sklearn.ensemble import RandomForestRegressor
180 y_pred = cross_val_predict(RandomForestRegressor(), X, y, cv=5)
181 print('RMSE:', metrics.mean_squared_error(y, y_pred, squared=True))
182 print('MAE:', metrics.mean_absolute_error(y, y_pred))
183 print('R2:', metrics.r2_score(y, y_pred))
184
185 cabecalho('Gradient Boosting')
186 from sklearn.ensemble import GradientBoostingRegressor
187 clf_gradboost = GradientBoostingRegressor(
188                                         max_depth=3,
189                                         learning_rate=0.1,
190                                         n_estimators=100,
191                                         max_leaf_nodes=None)
192 y_pred = cross_val_predict(clf_gradboost, X, y, cv=5)
193 print('RMSE:', metrics.mean_squared_error(y, y_pred, squared=True))
194 print('MAE:', metrics.mean_absolute_error(y, y_pred))
195 print('R2:', metrics.r2_score(y, y_pred))
196
197 cabecalho('AdaBoost')
198 from sklearn.ensemble import AdaBoostRegressor
199 y_pred = cross_val_predict(AdaBoostRegressor(), X, y, cv=5)
200 print('RMSE:', metrics.mean_squared_error(y, y_pred, squared=True))
201 print('MAE:', metrics.mean_absolute_error(y, y_pred))
202 print('R2:', metrics.r2_score(y, y_pred))
```

```
199 cabecalho('Decision Tree')
200 from sklearn.tree import DecisionTreeRegressor
201 y_pred = cross_val_predict(DecisionTreeRegressor(), X, y, cv=5)
202 print('RMSE:', metrics.mean_squared_error(y, y_pred, squared=True))
203 print('MAE:', metrics.mean_absolute_error(y, y_pred))
204 print('R2:', metrics.r2_score(y, y_pred))

205
206
207 cabecalho('RandomizedSearchCV')
208 from sklearn.model_selection import RandomizedSearchCV
209 param_grid = {
210     "loss": ["squared_error"],
211     "learning_rate": [0.02, 0.05, 0.1, 0.2, 0.3],
212     "min_samples_split": [2, 4, 8, 12, 16, 20],
213     "min_samples_leaf": [1, 2, 4, 8, 12, 16, 20],
214     "max_depth": [3, 5, 7, 8, 10],
215     "max_features": [None, 'sqrt'],
216     "criterion": ["friedman_mse"],
217     "subsample": [1.0],
218     "max_leaf_nodes": [8, 12, 16, 20, 24, 28, 32],
219     "n_estimators": [50, 60, 70, 80, 90, 100, 120, 140, 160, 165,
220         170, 175, 180, 185, 190, 195, 200, 220, 240, 260, 270, 280, 300]
221 }
222
223 random_search = RandomizedSearchCV(GradientBoostingRegressor(),
224                                     param_grid,
225                                     cv=5,
226                                     scoring='neg_root_mean_squared_error',
227                                     refit=True,
228                                     n_iter=20,
229                                     n_jobs=None,
230                                     verbose=0,
231                                     pre_dispatch='2*n_jobs',
232                                     random_state=seed,
233                                     return_train_score=True)

234 random_search.fit(X,y)
235 print('Melhores parâmetros', random_search.best_params_)

236
237 cabecalho('Modelo final')
238 modelo = random_search.best_estimator_
239 print(modelo)
```

```
241 cabecalho('Resultado de cada validação cruzada')
242 random_results = pd.DataFrame(random_search.cv_results_)[['params',
243     'rank_test_score', 'mean_test_score']]
244 print(random_results)
245
246 cabecalho('Performance do modelo no conjunto de teste')
247 testX = test_set.drop(['Band gap', 'Material'], axis='columns')
248 testY = test_set['Band gap']
249 result = modelo.predict(testX)
250 print('RMSE:', metrics.mean_squared_error(testY, result, squared=True))
251 print('MAE:', metrics.mean_absolute_error(testY, result))
252 print('R2:', metrics.r2_score(testY, result))
253
254 cabecalho('Predicao')
255 ## new_data ##
256 df = pd.read_csv('dataset_pred_insulators.csv')
257 df.drop('Unnamed: 0', axis='columns', inplace=True)
258
259 ## informando quem são as features ##
260 X_new = df.drop(['Material'], axis='columns')
261 y_new = modelo.predict(X_new)
262 print(y_new)
263
264 name_columns = list(df.columns)
265 name_columns.append('band gap')
266
267 df_insulators = pd.DataFrame(columns=name_columns)
268 for i in range(0,266):
269     nova_entrada = list(df.loc[i])
270     nova_entrada.append(y_new[i])
271     df_insulators.loc[len(df_insulators)] = nova_entrada
272 print(df_insulators.head())
273
274 df_insulators.to_csv('prediction_gap_pbe.csv')
275
276 df_ultra = pd.DataFrame(columns=name_columns)
277 ## separando os ultrawide band gap ##
278 for i in range(0,266):
279     if y_new[i] > 2.0:
280         nova_entrada = list(df.loc[i])
281         nova_entrada.append(y_new[i])
282         df_ultra.loc[len(df_ultra)] = nova_entrada
```

```

283
284 df_ultra.to_csv('insulators_ultra_pbe.csv')

```

Para previsão do gap dos novos materiais com estequiometria AB₂ e *features* estatísticas, basta modificar as últimas linhas do código anterior:

```

1 cabecalho('Predicao')
2 ## new_data ##
3 df = pd.read_csv('dataset_pred_insulators.csv')
4 df.drop('Unnamed: 0', axis='columns', inplace=True)
5
6 ## informando quem são as features ##
7 X_new = df.drop(['Material'], axis='columns')
8 y_new = modelo.predict(X_new)
9 print(y_new)
10
11 name_columns = list(df.columns)
12 name_columns.append('band gap')
13
14 df_insulators = pd.DataFrame(columns=name_columns)
15 for i in range(0,313):
16     nova_entrada = list(df.loc[i])
17     nova_entrada.append(y_new[i])
18     df_insulators.loc[len(df_insulators)] = nova_entrada
19 print(df_insulators.head())
20
21 df_insulators.to_csv('prediction_gap_pbe.csv')
22
23 df_ultra = pd.DataFrame(columns=name_columns)
24 ## separando os ultrawide band gap ##
25 for i in range(0,313):
26     if y_new[i] > 2.0:
27         nova_entrada = list(df.loc[i])
28         nova_entrada.append(y_new[i])
29         df_ultra.loc[len(df_ultra)] = nova_entrada
30
31 df_ultra.to_csv('insulators_ultra_pbe.csv')

```

A.9 Regressão AB₂

Segue o algoritmo com o treinamento para a regressão no *dataset* com materiais de estequiometria AB₂ advindas da base de dados C2DB.

```
1 # coding=utf-8
2
3 def cabecalho(msg):
4     print(100*'=')
5     print(f'{msg:^100}')
6     print(100*'=')
7
8 # importar pacotes
9 from sklearn.model_selection import train_test_split
10 import matplotlib.pyplot as plt
11 import pandas as pd
12 import numpy as np
13
14 # Configura o número de linhas e colunas que serão mostrados, e o
15 # comprimento das colunas nos dataframes
15 pd.set_option('display.max_columns', None)
16 pd.set_option('display.max_rows', None)
17 pd.set_option('max_colwidth', None)
18
19 # para conseguir repetir o experimento
20 seed = 50
21 np.random.seed(seed)
22
23 # carregando dataset
24 cabecalho('Dataset')
25 df = pd.read_csv('AB2_prototype_atomic_insulators.csv')
26 df.drop('Unnamed: 0', axis='columns', inplace=True)
27 print(df.head())
28
29 cabecalho('Tamanho do dataset')
30 print(df.shape)
31
32 cabecalho('Quantidade de protótipos')
33 print(len(df['prototype'].unique()))
34
35 cabecalho('Quantidade de cada protótipo')
36 print(df['prototype'].value_counts())
37
38 ## Excluindo linhas onde o protótipo é "Unfined" ##
39 df.drop(df.index[df['prototype'] == 'Undefined'], axis=0, inplace=True)
40
41 ## Atribuindo valores inteiros aos protótipos ##
```

```
43 cabecalho('Atribuindo numeros inteiros aos prototipos')
44 df['prototype'] = df['prototype'].apply(lambda x : 2 if x == 'FeOCl'
45                                         else x)
46 df['prototype'] = df['prototype'].apply(lambda x : 3 if x == 'CdI2' else
47                                         x)
48 df['prototype'] = df['prototype'].apply(lambda x : 4 if x == 'MoS2' else
49                                         x)
50 df['prototype'] = df['prototype'].apply(lambda x : 5 if x == 'GeS2' else
51                                         x)
52 df['prototype'] = df['prototype'].apply(lambda x : 6 if x == 'CH' else x
53                                         )
54 df['prototype'] = df['prototype'].apply(lambda x : 7 if x == 'FeSe' else
55                                         x)
56 df['prototype'] = df['prototype'].apply(lambda x : 8 if x == 'GaS' else
57                                         x)
58 df['prototype'] = df['prototype'].apply(lambda x : 9 if x == 'BiTeI'
59                                         else x)
60 df['prototype'] = df['prototype'].apply(lambda x : 10 if x == 'MoSSe'
61                                         else x)
62 df['prototype'] = df['prototype'].apply(lambda x : 11 if x == 'WTe2'
63                                         else x)
64 df['prototype'] = df['prototype'].apply(lambda x : 12 if x == 'GaSe'
65                                         else x)
66 df['prototype'] = df['prototype'].apply(lambda x : 13 if x == 'MnPSe3'
67                                         else x)
68 df['prototype'] = df['prototype'].apply(lambda x : 14 if x == 'BiI3'
69                                         else x)
70 df['prototype'] = df['prototype'].apply(lambda x : 15 if x == 'PbSe'
71                                         else x)
72 df['prototype'] = df['prototype'].apply(lambda x : 16 if x == 'CrW3S8'
73                                         else x)
74 df['prototype'] = df['prototype'].apply(lambda x : 17 if x == 'Ti4C3O2'
75                                         else x)
76 df['prototype'] = df['prototype'].apply(lambda x : 18 if x == 'Ti3C2O2'
77                                         else x)
78 df['prototype'] = df['prototype'].apply(lambda x : 19 if x == 'Ti2CO2'
79                                         else x)
80 df['prototype'] = df['prototype'].apply(lambda x : 20 if x == 'PdS2'
81                                         else x)
82 df['prototype'] = df['prototype'].apply(lambda x : 21 if x == 'TiCl3'
83                                         else x)
84 df['prototype'] = df['prototype'].apply(lambda x : 22 if x == 'PbA2I4'
85                                         else x)
```

```
65 df['prototype'] = df['prototype'].apply(lambda x : 23 if x == 'NiSe' 
66     else x)
66 df['prototype'] = df['prototype'].apply(lambda x : 24 if x == 'TiS3' 
67     else x)
67 df['prototype'] = df['prototype'].apply(lambda x : 25 if x == 'AuSe' 
68     else x)
68 df['prototype'] = df['prototype'].apply(lambda x : 26 if x == 'C3N' else 
69     x)
69 df['prototype'] = df['prototype'].apply(lambda x : 27 if x == 'AgBr3' 
70     else x)
70 df['prototype'] = df['prototype'].apply(lambda x : 28 if x == 'GeSe' 
71     else x)
71 df['prototype'] = df['prototype'].apply(lambda x : 29 if x == 'Ti3C2H2O2' 
72     else x)
72 df['prototype'] = df['prototype'].apply(lambda x : 30 if x == 'VTe3' 
73     else x)
73 df['prototype'] = df['prototype'].apply(lambda x : 31 if x == 'Ti2CH2O2' 
74     else x)
74 df['prototype'] = df['prototype'].apply(lambda x : 32 if x == 'HfBrS' 
75     else x)
75 df['prototype'] = df['prototype'].apply(lambda x : 33 if x == 'Ti4C3H2O2' 
76     else x)
76 df['prototype'] = df['prototype'].apply(lambda x : 34 if x == 'CrWS4' 
77     else x)
77 df['prototype'] = df['prototype'].apply(lambda x : 35 if x == 'Ti4C3' 
78     else x)
78 df['prototype'] = df['prototype'].apply(lambda x : 36 if x == 'Ti3C2' 
79     else x)
79 df['prototype'] = df['prototype'].apply(lambda x : 37 if x == 'RhO' else 
80     x)
80 df['prototype'] = df['prototype'].apply(lambda x : 38 if x == 'MnS2' 
81     else x)
81 df['prototype'] = df['prototype'].apply(lambda x : 39 if x == 'BN' else 
82     x)
82 df['prototype'] = df['prototype'].apply(lambda x : 40 if x == 'ReS2' 
83     else x)
83 df['prototype'] = df['prototype'].apply(lambda x : 41 if x == 'ScPSe3' 
84     else x)
84 df['prototype'] = df['prototype'].apply(lambda x : 42 if x == 'SnS' else 
85     x)
85 df['prototype'] = df['prototype'].apply(lambda x : 43 if x == 'C' else x
    )
```

```
86 df['prototype'] = df['prototype'].apply(lambda x : 44 if x == 'PbS' else
87     x)
88 df['prototype'] = df['prototype'].apply(lambda x : 45 if x == 'ISb' else
89     x)
90 df['prototype'] = df['prototype'].apply(lambda x : 46 if x == 'P' else x
91     )
92 df['prototype'] = df['prototype'].apply(lambda x : 47 if x == 'CH2Si'
93     else x)
94 df['prototype'] = df['prototype'].apply(lambda x : 48 if x == 'VPSe3'
95     else x)
96
97 print(df['prototype'].value_counts())
98
99 cabecalho('Excluindo protótipos com poucos representantes')
100
101 for group in df['prototype']:
102     if (df['prototype'] == group).sum() < 10:
103         df.drop(df.index[df['prototype'] == group], axis=0, inplace=True
104     )
105
106 print(df['prototype'].value_counts())
107
108
109 cabecalho('Número de metais e isolantes com o funcionalpbe')
110 n_metal = n_insulator = 0
111
112 for i in df['Band gap']:
113     if i > 0.0000000001:
114         n_insulator += 1
115     else:
116         n_metal += 1
117
118 print(f' No de metais: {n_metal} \n No de isolantes: {n_insulator}')
119
120 cabecalho('Min e max do gap')
121 print('gap min', df['Band gap'].min())
122 print('gap max', df['Band gap'].max())
123
124 cabecalho('Intervalos de gap')
125 pri = seg = ter = qua = qui = sex = seti = oit = non = 0
126 for i in df['Band gap']:
127     if i > 0.00001 and i <= 1.0:
128         pri = pri + 1
129     if i > 1.0 and i <= 2.0:
```

```
123     seg = seg + 1
124     if i > 2.0 and i <= 3.0:
125         ter = ter + 1
126     if i > 3.0 and i <= 4.0:
127         qua = qua + 1
128     if i > 4.0 and i <= 5.0:
129         qui = qui + 1
130     if i > 5.0 and i <= 6.0:
131         sex = sex + 1
132     if i > 6.0 and i <= 7.0:
133         seti = seti + 1
134     if i > 7.0 and i <= 8.0:
135         oit = oit + 1
136     if i > 8.0 and i <= 9.0:
137         non = non + 1
138
139 print(f'''Quant. de materiais com gap entre 0 e 1: {pri}
140 Quant. de materiais com gap entre 1 e 2: {seg}
141 Quant. de materiais com gap entre 2 e 3: {ter}
142 Quant. de materiais com gap entre 3 e 4: {qua}
143 Quant. de materiais com gap entre 4 e 5: {qui}
144 Quant. de materiais com gap entre 5 e 6: {sex}
145 Quant. de materiais com gap entre 6 e 7: {seti}
146 Quant. de materiais com gap entre 7 e 8: {oit}
147 Quant. de materiais com gap entre 8 e 9: {non}''')
148 print('Soma dos materiais com todos os gaps:', np.sum([pri, seg, ter, qua
149             , qui, sex, seti, oit, non]))
150 #cabecalho('Salvando histograma ...')
151 #gap = df['Band gap'].copy()
152 #plt.hist(gap, bins=[0.0, 1.0, 2.0, 3.0, 4.0, 5.0 , 6.0, 7.0, 8.0, 9.0],
153           density=False, label='Data', rwidth=0.9)
154 #plt.savefig('hist_gap_pbe.png', dpi=300)
155
156 cabecalho('Procurando dados faltantes')
157 print(df.isnull().sum())
158
159 cabecalho('Excluindo dados faltantes')
160 df.dropna(inplace=True)
161 print(df.isnull().sum())
162 cabecalho('Tamanho do treino e teste')
```

```
163 # separando em conjunto de treino e teste de forma estratificada depois
164 # de embaralhar pseudo aleatoriamente
165 train_set, test_set = train_test_split(df, test_size=0.1, shuffle=True,
166                                         random_state=seed)
167 print('treino: ',train_set.shape)
168 print('teste: ', test_set.shape)
169
170 ## Treinando modelos ##
171 X = train_set.drop(['Band gap', 'Material'], axis='columns')
172 y = train_set['Band gap']
173
174 from sklearn import metrics
175 from sklearn.model_selection import cross_val_predict, cross_val_score,
176                                         cross_validate
177
178 cabecalho('Random Forest')
179 from sklearn.ensemble import RandomForestRegressor
180 y_pred = cross_val_predict(RandomForestRegressor(), X, y, cv=5)
181 print('RMSE:', metrics.mean_squared_error(y, y_pred, squared=True))
182 print('MAE:', metrics.mean_absolute_error(y, y_pred))
183 print('R2:', metrics.r2_score(y, y_pred))
184
185 cabecalho('Gradient Boosting')
186 from sklearn.ensemble import GradientBoostingRegressor
187 clf_gradboost = GradientBoostingRegressor(
188                                         max_depth=3,
189                                         learning_rate=0.1,
190                                         n_estimators=100,
191                                         max_leaf_nodes=None)
192 y_pred = cross_val_predict(clf_gradboost, X, y, cv=5)
193 print('RMSE:', metrics.mean_squared_error(y, y_pred, squared=True))
194 print('MAE:', metrics.mean_absolute_error(y, y_pred))
195 print('R2:', metrics.r2_score(y, y_pred))
196
197 plt.scatter(y,y_pred)
198 plt.xlabel('C2DB PBE bandgap (eV)')
199 plt.ylabel('ML PBE bandgap (eV)')
200 plt.legend(['Gradient Boosting'])
201 plt.savefig('regression_plot_ab2_atomic', dpi=300)
202
203 cabecalho('AdaBoost')
204 from sklearn.ensemble import AdaBoostRegressor
205 y_pred = cross_val_predict(AdaBoostRegressor(), X, y, cv=5)
```

```
203 print('RMSE:', metrics.mean_squared_error(y, y_pred, squared=True))
204 print('MAE:', metrics.mean_absolute_error(y, y_pred))
205 print('R2:', metrics.r2_score(y, y_pred))

206
207 cabecalho('Decision Tree')
208 from sklearn.tree import DecisionTreeRegressor
209 y_pred = cross_val_predict(DecisionTreeRegressor(), X, y, cv=5)
210 print('RMSE:', metrics.mean_squared_error(y, y_pred, squared=True))
211 print('MAE:', metrics.mean_absolute_error(y, y_pred))
212 print('R2:', metrics.r2_score(y, y_pred))

213
214
215 cabecalho('RandomizedSearchCV')
216 from sklearn.model_selection import RandomizedSearchCV
217 param_grid = {
218     "loss": ["squared_error"],
219     "learning_rate": [0.02, 0.05, 0.1, 0.2, 0.3],
220     "min_samples_split": [2, 4, 8, 12, 16, 20],
221     "min_samples_leaf": [1, 2, 4, 8, 12, 16, 20],
222     "max_depth": [3, 5, 7, 8, 10],
223     "max_features": [None, 'sqrt'],
224     "criterion": ["friedman_mse"],
225     "subsample": [1.0],
226     "max_leaf_nodes": [8, 12, 16, 20, 24, 28, 32],
227     "n_estimators": [50, 60, 70, 80, 90, 100, 120, 140, 160, 165,
228                   170, 175, 180, 185, 190, 195, 200, 220, 240, 260, 270, 280, 300]
229 }
230
231 random_search = RandomizedSearchCV(GradientBoostingRegressor(),
232                                     param_grid,
233                                     cv=5,
234                                     scoring='neg_root_mean_squared_error',
235                                     refit=True,
236                                     n_iter=20,
237                                     n_jobs=None,
238                                     verbose=0,
239                                     pre_dispatch='2*n_jobs',
240                                     random_state=seed,
241                                     return_train_score=True)

242 random_search.fit(X,y)
243 print('Melhores parâmetros', random_search.best_params_)
```

```

245 cabecalho('Modelo final')
246 modelo = random_search.best_estimator_
247 print(modelo)
248
249 cabecalho('Resultado de cada validação cruzada')
250 random_results = pd.DataFrame(random_search.cv_results_)[['params', 'rank_test_score', 'mean_test_score']]
251 print(random_results)
252
253 cabecalho('Performance do modelo no conjunto de teste')
254 testX = test_set.drop(['Band gap', 'Material'], axis='columns')
255 testY = test_set['Band gap']
256 result = modelo.predict(testX)
257 print('RMSE:', metrics.mean_squared_error(testY, result, squared=True))
258 print('MAE:', metrics.mean_absolute_error(testY, result))
259 print('R2:', metrics.r2_score(testY, result))
260
261 import pickle
262 with open('gradboost_reg_ab2_atomic.pkl', 'wb') as file:
263     pickle.dump(modelo, file)

```

E a previsão do gap dos materiais propostos com estequiometria AB₂ e *features* atômicas se encontra abaixo:

```

1 # coding=utf-8
2 def cabecalho(msg):
3     print(100*'=')
4     print(f'{msg:^100}')
5     print(100*'=')
6
7 import pickle
8 import pandas as pd
9
10 with open('gradboost_reg_ab2_atomic.pkl', 'rb') as f:
11     model_reg = pickle.load(f)
12
13 cabecalho('Predicao')
14 ## new_data ##
15 df = pd.read_csv('dataset_pred_insulators.csv')
16 df.drop('Unnamed: 0', axis='columns', inplace=True)
17
18 ## informando quem são as features ##
19 X_new = df.drop(['Material'], axis='columns')

```

```
20 y_new = model_reg.predict(X_new)
21 print(y_new)
22
23 name_columns = list(df.columns)
24 name_columns.append('band gap')
25
26 df_insulators = pd.DataFrame(columns=name_columns)
27 for i in range(0,302):
28     nova_entrada = list(df.loc[i])
29     nova_entrada.append(y_new[i])
30     df_insulators.loc[len(df_insulators)] = nova_entrada
31 print(df_insulators.head())
32
33 df_insulators.to_csv('prediction_gap_pbe.csv')
34
35 df_ultra = pd.DataFrame(columns=name_columns)
36 ## separando os ultrawide band gap ##
37 for i in range(0,302):
38     if y_new[i] > 2.0:
39         nova_entrada = list(df.loc[i])
40         nova_entrada.append(y_new[i])
41         df_ultra.loc[len(df_ultra)] = nova_entrada
42
43 df_ultra.to_csv('insulators_ultra_pbe.csv')
```