UNIVERSIDADE FEDERAL FLUMINENSE
IF - Instituto de Física
PPGF - Programa de Pós-Graduação em Física

Erick Filipe Oliveira Gomes

Accelerating Discovery 2D Orbitronics Materials Via Machine Learning

Niterói - RJ
2025

Erick Filipe Oliveira Gomes

Accelerating Discovery 2D Orbitronics Materials Via Machine Learning

Dissertation presented to the graduate program in physics at the Fluminense Federal University as a requirement for obtaining a master's degree in physics.

Advisor: Prof. Dr. Marcio Jorge Teles da Costa
Co-Advisor: Prof. Dr. Rodrigo Garcia Amorim

Niterói – RJ
2025

Erick Filipe Oliveira Gomes

Accelerating Discovery 2D Orbitronics Materials Via Machine Learning

Dissertation presented to the graduate program in physics at the Fluminense Federal University as a requirement for obtaining a master's degree in physics.

Aprovado em de de .

BANCA EXAMINADORA

Prof. Dr. Marcio Jorge Teles da Costa - Advisor
Fluminense Federal University - UFF

Prof. Dr. Roberto Bechara Muniz
Fluminense Federal University - UFF

Dr. Gabriel Ravanhani Schleder
CNPEM - National Center for Research in Energy and Materials

Niterói – RJ
2025

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

G633a Gomes, Erick Filipe Oliveira
Accelerating Discovery 2D Orbitronics Materials Via Machine Learning / Erick Filipe Oliveira Gomes. - 2025.
102 f.: il.

Orientador: Marcio Jorge Teles da Costa.
Coorientador: Rodrigo Garcia Amorim.
Dissertação (mestrado)-Universidade Federal Fluminense, Instituto de Física, Niterói, 2025.

1. Física da Matéria Condensada. 2. Aprendizado de Máquina. 3. Informática de Materiais. 4. Orbitrônica. 5. Produção intelectual. I. Costa, Marcio Jorge Teles da, orientador. II. Amorim, Rodrigo Garcia, coorientador. III. Universidade Federal Fluminense. Instituto de Física. IV. Título.

CDD - XXX

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

Agradecimentos

Quero agradecer, primeiramente, a Deus, que sempre me deu saúde e força para encarar os desafios e que colocou em meu caminho pessoas excepcionais que me ajudaram a chegar até aqui. Agradeço a toda a minha família, sem exceções, que sempre esteve ao meu lado em todos os momentos, aconselhando-me e incentivando-me. Agradeço à minha namorada, Amanda, que está comigo desde o início dessa caminhada, ainda na graduação, e à sua família.

Quero agradecer também ao professor Rodrigo Amorim, que sempre foi um grande incentivador ao longo dessa trajetória. Meu agradecimento se estende a Marcelo Albuquerque, que me ajudou lendo minha dissertação e sugerindo melhorias. Por fim, sou grato ao meu orientador, Márcio Costa, pela proposição do trabalho e pela orientação.

Resumo

Nesta dissertação, investigamos a aplicação de técnicas de aprendizado de máquina para prever propriedades eletrônicas e de transporte em materiais, com ênfase na condutividade Hall orbital (OHC). A pesquisa foi motivada pela necessidade de acelerar a identificação de materiais com propriedades específicas, como isolantes com alto potencial para aplicações tecnológicas. Para isso, construímos modelos de aprendizado supervisionado utilizando features físico-químicas extraídas de elementos da tabela periódica, como eletronegatividade, orbitais atômicos e número de orbitais não preenchidos, combinadas com descritores estatísticos ponderados.

Os modelos foram avaliados com base em métricas como MSE, RMSE e R^2, e o modelo Extra Trees apresentou o melhor desempenho, com um RMSE de 0.45 e R^2 de 0.81. As predições realizadas foram validadas utilizando cálculos de primeiros princípios baseados na teoria do funcional da densidade (DFT) e o software PAOFLOW, que permitiram calcular tanto as bandas de energia quanto a condutividade Hall orbital. Um dos materiais analisados, o $ZrSe_2$ no protótipo CdI, foi previsto como isolante pelo modelo, o que foi corroborado pelos cálculos DFT, revelando um gap de energia de 0.27 eV. Além disso, o valor predito da OHC (0.72 eV) apresentou uma diferença aceitável em relação ao valor calculado (0.61 eV), com um erro de apenas 0.11 eV.

Os resultados demonstram o potencial das técnicas de aprendizado de máquina como ferramentas complementares aos métodos tradicionais de simulação, reduzindo significativamente o tempo necessário para a triagem e a análise de materiais. Esta abordagem apresenta implicações importantes para o design de novos materiais funcionais e o avanço na compreensão de suas propriedades eletrônicas e topológicas.

Palavras-chaves: Aprendizado de Máquina, Materiais Funcionais ,Condutividade Hall Orbital (OHC), Isolantes, Teoria do Funcional da Densidade (DFT), PAOFLOW, Previsão de Propriedades de Materiais, Inteligência Artificial em Materiais, Física Computacional, Modelagem de Materiais, Bandas de Energia e Design de Materiais.

Abstract

In this dissertation, we investigate the application of machine learning techniques to predict electronic and transport properties in materials, with an emphasis on orbital Hall conductivity (OHC). The research was motivated by the need to accelerate the identification of materials with specific properties, such as insulators with high potential for technological applications. To this end, we built supervised learning models using physicochemical features extracted from elements of the periodic table, such as electronegativity, atomic orbitals and number of unfilled orbitals, combined with weighted statistical descriptors.

The models were evaluated based on metrics such as MSE, RMSE and R^2, and the Extra Trees model presented the best performance, with an RMSE of 0.45 and R^2 of 0.81. The predictions made were validated using first-principles calculations based on density functional theory (DFT) and the PAOFLOW software, which allowed the calculation of both energy bands and orbital Hall conductivity. One of the analyzed materials, $ZrSe_2$ in the CdI prototype, was predicted to be an insulator by the model, which was corroborated by DFT calculations, revealing an energy band gap of 0.27 eV. Furthermore, the predicted OHC value (0.72 eV) showed an acceptable difference in relation to the calculated value (-0.61 eV), with an error of only 0.11 eV.

The results demonstrate the potential of machine learning techniques as complementary tools to traditional simulation methods, significantly reducing the time required for screening and analyzing materials. This approach has important implications for the design of new functional materials and for advancing the understanding of their electronic and topological properties.

Keywords: Machine Learning, Functional Materials, Orbital Hall Conductivity (OHC), Insulators, Density Functional Theory (DFT), PAOFLOW, Materials Property Prediction, Artificial Intelligence in Materials, Computational Physics, Materials Modeling, Energy Bands and Materials Design.

List of Figures

Contents

1 Introduction

The progress of human civilization has been intricately linked to the discovery and utilization of new materials. From the rudimentary tools of the Stone Age to the advanced silicon technologies of today, materials have defined the trajectory of human development. The different eras of history are a testament to this connection, bearing names such as the Stone Age, the Bronze Age, and the Iron Age, reflecting the pivotal role materials played in shaping human life. Beyond these early advances, the discovery of glass, steel, ceramics, concrete, and polymers has transformed societies, enabling innovations from everyday objects to cutting-edge technologies such as rockets and robots[2–6].

As humanity faces pressing global challenges, the need for sustainable materials has never been greater. Materials science will be crucial for achieving breakthroughs in renewable energy, healthcare, agriculture, and even in the arts. The ability to develop materials with minimal environmental impact is pivotal for advancing toward a sustainable future.

Despite these advancements, the traditional approach to material discovery—based on synthesis and experimental trial-and-error—remains a time-consuming and resource-intensive process. To overcome these limitations, computational methods and machine learning are being increasingly employed to accelerate material discovery. These approaches enable scientists to predict and optimize material properties before experimental synthesis, significantly reducing the time and cost associated with traditional methods.

These advancements highlight a shift from traditional experimental methodologies to computational approaches, fundamentally transforming material discovery. By leveraging machine learning and computational modeling, researchers can not only accelerate the identification of promising materials but also gain deeper insights into their underlying electronic and transport properties. This is particularly relevant for the study of insulating materials with orbital Hall conductivity (OHC), where theoretical predictions can guide the selection of candidate materials for experimental validation. As a result, the integration of computational techniques with materials science is paving the way for more efficient exploration of novel materials for spintronics and quantum technologies.

Materials exhibiting OHC, such as transition metal dichalcogenides (TMDs) like MoS_2 and WSe_2, can be utilized for orbital-current injection and orbital torque transfer. This is significant for developing next-generation spintronic devices that leverage orbital degrees of freedom instead of solely relying on spin currents, potentially enhancing device efficiency and functionality[7].

The unique properties of orbital Hall insulators can be harnessed in quantum computing applications. The ability to manipulate orbital angular momentum could lead to new qubit designs that are less susceptible to decoherence, providing a pathway for more robust

quantum information processing systems[8]. Insulating materials with large OHC can improve the performance of magnetic sensors, particularly those that exploit the orbital Hall effect to detect magnetic fields or changes in magnetic states without the need for external magnetic materials[9].

The study of OHC contributes to the understanding of topological insulators, where their unique electronic properties can be exploited for applications in advanced electronic devices. These materials may enable robust edge states that are less affected by impurities, making them suitable for low-power electronics[10].

Orbital Hall conductivity can also play a role in energy harvesting technologies, where the manipulation of orbital currents could lead to a more efficient conversion of energy from various sources into usable electrical energy[10].

This study explores the integration of machine learning techniques with materials science to address two critical challenges:

1. **Classification of materials into metals and insulators:** Using electronic structure data, machine learning models are developed to classify materials into these fundamental categories.

2. **Prediction of orbital Hall conductivity:** For insulators, a regression model is proposed to predict the orbital Hall conductivity (OHC), a key property with implications for spintronics and orbitronics. The study builds upon the fundamental understanding of the Hall effects, including the classical Hall effect, the spin Hall effect, and the emerging orbital Hall effect. These phenomena illustrate how charge, spin, and orbital momentum can be manipulated in materials, paving the way for advanced technologies such as orbital-based memory devices and sensors.

By leveraging data-driven approaches, this research contributes to the ongoing revolution in material discovery, opening new pathways for the design and characterization of materials with tailored properties for technological applications.

1.1 Classical Hall Effect

The classic Hall effect was discovered by Edwin Hall in 1879, this effect consists of the emergence of an electric field resulting from the application of a magnetic field to a material with electric current. This phenomenon is due to an electric force that appears in the material causing the charge carrier to be deflected.

When an electric current (I) is applied to a conductor, charge carriers (usually electrons) flow in a specific direction.

When a magnetic field (B) is applied perpendicular to the current, the charge carriers experience the Lorentz force, described in the equation 1.1,

$$\vec{F} = q(\vec{v} \times \vec{B}). \tag{1.1}$$

This force bends the trajectory of the charge carriers, deflecting them to one side of the material.

Due to the deflection, charges accumulate on one side of the conductor, generating a potential difference (V_H) perpendicular to the current and the magnetic field.

The accumulation of charge creates an internal electric field (E_H) that balances the Lorentz force. When equilibrium is achieved, a stable Hall voltage appears.

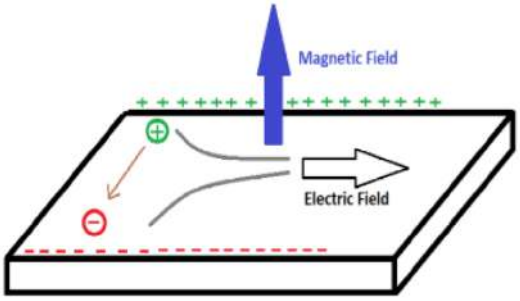

Figure 1 – Classical hall effects.

1.2 Spin Hall Effect

The spin Hall effect is a physical phenomenon in which electrons with opposite spins separate laterally in a non-magnetic material due to spin-orbit interactions. It is analogous to the classical Hall effect, but instead of depending on the electric charge of the carriers, it is related to their spin momentum.

In materials where the spin Hall effect occurs, the spin-orbit interaction creates a coupling between the linear momentum of the electrons and their spin. This means that electrons with spins-up ($\uparrow$) and spin-down ($\downarrow$) experience effective forces in opposite directions.

When an electric current flows through the material, the electrons move and the spin-orbit interaction deflects the electrons with spin-up $\uparrow$ to one side of the material and those with spin-down $\downarrow$ to the opposite side. This separation generates a spin polarization at the edges of the material.

Unlike the classical Hall effect, the spin Hall effect does not require the application of an external magnetic field. It is entirely a consequence of the intrinsic properties of the material and the spin-orbit interaction.

The spin Hall effect is important for spintronics, an area of physics and engineering that seeks to exploit the spin of electrons, in addition to their charge, in electronic devices. Examples of applications include:

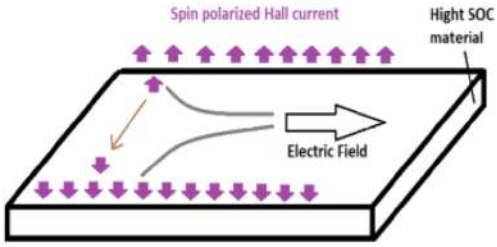

Figure 2 – Spin hall effects.

- Magnetic memories, such as MRAM (Magnetic Random Access Memory);
- Generation and detection of spin currents;
- Advanced magnetic sensors;

1.3 Orbital Hall Effect

The orbital Hall effect is a phenomenon in which the orbital motion of electrons in a material gives rise to an anomalous current, which flows perpendicular to the applied electric current, without the need for an external magnetic field. It is analogous to the spin Hall effect, but instead of involving the spin of the electrons, it is associated with the orbital angular momentum of the electrons[11, 12].

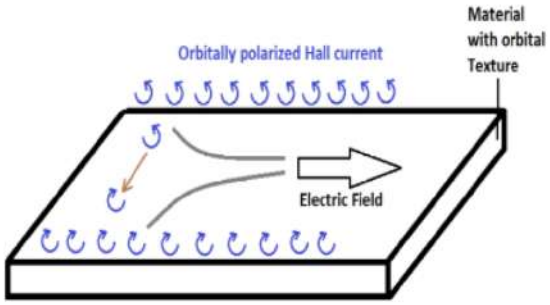

Figure 3 – Orbital hall effects.

The orbital Hall effect is an emerging area in condensed matter physics with implications for orbitronics, a new subfield of spintronics that seeks to manipulate the orbital momentum of electrons for technological applications. Potential applications include[13]:

- Orbital momentum-based memory devices;
- Detection and manipulation of orbital currents for advanced sensors;
- Exploration of new quantum states of matter, such as topological insulators with orbital characteristics.

This effect extends the ability of electronic control beyond spin and electric charge to include orbital momentum as an exploitable degree of freedom.

The equation 1.2 presented describes the orbital Hall conductivity (OHC) and the spin Hall conductivity (SHC) from the Berry curvature modified by an operator X_η.

$$\sigma^{(\eta)}_{\mathrm{OH}(SH)} = \frac{e}{(2\pi)^2} \sum_n \int_{\mathrm{BZ}} d^2k \, f_{nk} \Omega^{X_\eta}_{n,k} \tag{1.2}$$

To understand its physical meaning, we need to analyze the operators involved, the chain operator is defined by equation1.3:

$$j^{X_\eta}_{y,\mathbf{k}} = \frac{(X_\eta v_y(\mathbf{k}) + v_y(\mathbf{k}) X_\eta)}{2} \tag{1.3}$$

where, X_η represents an operator that can be either the orbital angular momentum $\hat{l}_\eta$ or the spin $\hat{s}_\eta$, depending on whether we are calculating the orbital Hall conductivity or the spin Hall conductivity, and $v_y(\mathbf{k})$ is the velocity operator in the y direction.

The conventional Berry curvature is related to the geometric phase acquired by the electronic states in momentum space. In the equation 1.4 we have a modified version given by:

$$\Omega^{X_\eta}_{n,\mathbf{k}} = 2\hbar \sum_{m \neq n} \mathrm{Im} \left[\frac{\langle u_{n,\mathbf{k}} | j^{X_\eta}_{y,\mathbf{k}} | u_{m,\mathbf{k}} \rangle \langle u_{m,\mathbf{k}} | v_x(\mathbf{k}) | u_{n,\mathbf{k}} \rangle}{(E_{n,\mathbf{k}} - E_{m,\mathbf{k}} + i0^+)^2} \right] \tag{1.4}$$

where:

- $\langle u_{n,\mathbf{k}} | j^{X_\eta}_{y,\mathbf{k}} | u_{m,\mathbf{k}} \rangle$ represents the transition matrix element for the current operator.
- $\langle u_{m,\mathbf{k}} | v_x(\mathbf{k}) | u_{n,\mathbf{k}} \rangle$ introduces the **velocity operator** in the x direction.
- The denominator $(E_{n,\mathbf{k}} - E_{m,\mathbf{k}} + i0^+)^2$ ensures the correct weighting of interband transitions.

The selection of X_η determines whether we are computing the orbital or spin Hall conductivity:

$$X_\eta = \hat{l}_\eta \tag{1.5}$$

$$X_\eta = \hat{s}_\eta \tag{1.6}$$

- If $X_\eta = \hat{l}_\eta$, we obtain the orbital Hall conductivity (OHC).
- If $X_\eta = \hat{s}_\eta$, we obtain the spin Hall conductivity (SHC).

1.4 General Objectives

Accelerating the discovery of new materials by exploring material database is the primary aim of this work. To use machine learning algorithms to train models and thus to

make inference properties of interest. After the models are built, we will make inferences and validate orbital hall conductivity values using first-principle calculations.

1.5 Specific Objectives

We will build a workflow with machine learning models in which the main objective will be to make inferences of the orbital hall conductivity value of two-dimensional insulating materials. The calculation of the orbital hall conductivity can have a high computational cost when done using first principle calculations. Therefore, the objective is, through the exploration of the Computational 2D Materials Database (C2DB)[14], to build a workflow that separates the insulating materials from the metallic ones. Then, a machine learning model is used to infer the value of the orbital hall conductivity.

2 Density Functional Theory

This study will employ density functional theory (DFT) to verify the outcomes generated by machine learning models. Consequently, it is essential to establish all the requisite tools for utilizing DFT prior to initiating the model development process. The data we will utilize for training the algorithms is essential, as it originates from electronic structure calculations and will also serve to validate the results following the predictions.

The advancement of quantum mechanics has enabled the characterization of the behavior of entities at the nanoscopic level, including atoms, molecules, and solids. However, the theory is limited in its analytical capabilities, effectively addressing only a select number of problems, such as free particles, harmonic oscillators, the hydrogen atom, among others. To address many-body problems, like the one presented in this study, first-principles methods are employed, specifically utilizing density functional theory (DFT) developed by Pierre Hohenberg, Walter Kohn, and Lu Jeu Sham [15]. DFT has become one of the most widely used theories in materials science and quantum chemistry, and its significance was further recognized when Walter Kohn was awarded the Nobel Prize in Chemistry in 1998 for its development.

The comprehensive formulation of DFT has required the prior advancement of alternative approximations that elucidate phenomena at the nanoscopic scale. The subsequent sections will address the fundamental components of DFT and its practical implications.

2.1 Quantum Description of Atoms, Molecules and Solids

The formulation of quantum mechanics originated at the close of the 19th century and continued into the 20th century. The Schrödinger equation, formulated by Austrian physicist Erwin Schrödinger, a Nobel Prize laureate in 1933, is a fundamental component of quantum theory. This accolade was attributed to his contributions to quantum mechanics through the formulation of the Schrödinger equation. The Schrödinger equation is pivotal in quantum mechanics, as it delineates the behavior of atoms in a solid by computing the interactions among each particle within the system. The equation is consequently expressed in its stationary form:

$$\hat{\mathcal{H}}\Psi(\vec{r},\vec{R}) = E_{tot}\Psi(\vec{r},\vec{R}). \tag{2.1}$$

To describe the energy of a system, it is necessary to write the Hamiltonian of the system, as follows:

$$\hat{\mathcal{H}} = \hat{T} + \hat{V}, \tag{2.2}$$

in other words, the Hamiltonian is equal to the kinetic energy plus the potential energy of a system. Therefore, in order to describe the total energy of the system, it is necessary to calculate the kinetic and potential energy between all the bodies taking part in the interactions. Thus, we can write:

$$\hat{\mathcal{H}} = \hat{T}_e + \hat{T}_n + \hat{V_{ne}} + \hat{V_{nn}} + \hat{V_{ee}}, \tag{2.3}$$

i.e. the kinetic energy of the electrons($\hat{T}_e$) and the kinetic energy of the nuclei(T_n), as well as the nucleus-electron(V_{ne}), nucleus-nucleus(V_{nn}) and electron-electron(V_{ee}) potentials, the latter being the Coulombian potentials between the bodies in the system that depend on the distance between the particles. With this, the Hamiltonian can be written as follows[16]:

$$\hat{\mathcal{H}} = -\frac{1}{2}\sum_{i=1}^{N}\frac{1}{M_\alpha}\nabla_\alpha^2 - \frac{1}{2}\sum_{i=1}^{N}\nabla_i^2 - \sum_{\alpha=1}^{M}\sum_{i=1}^{N}\frac{Z_\alpha}{|R_\alpha - r_i|} + \sum_{i=1}^{M}\sum_{j>1}^{N}\frac{1}{|r_i - r_j|} + \sum_{\alpha=1}^{M}\sum_{\beta>\alpha}^{M}\frac{Z_\alpha Z_\beta}{|R_\alpha - R_\beta|}. \tag{2.4}$$

As mentioned earlier, many-body problems do not have analytical solutions due to their complexity, so it is necessary to take approximations to solve them.

2.2 The Born-Oppenheimer Approximation

The first approximation that must be made is to decouple the movement of electrons and nuclei, i.e:

$$\Psi(\vec{r}, \vec{R}) = \psi_n(\vec{R})\psi_e(\vec{r}; \vec{R}), \tag{2.5}$$

where ψ_n is the wave function of the nucleus and ψ_e is the electronic wave function. It is only possible to make this approximation for two reasons: firstly, the mass of the nuclei is much greater than the mass of the electrons, which results in movement on different time scales, i.e. any movement of the nuclei is accompanied instantaneously by the electrons. Secondly, the speed of the electrons is much greater than the speed of the nuclei, i.e. the nucleus can be considered stationary.

Therefore, the 2.1 equation can be written as follows:

$$[\hat{T}_e + \hat{T}_n + \hat{V_{ne}} + \hat{V_{nn}} + \hat{V_{ee}}]\psi_n(\vec{R})\psi_e(\vec{r}; \vec{R}) = E_{tot}\psi_n(\vec{R})\psi_e(\vec{r}; \vec{R}), \tag{2.6}$$

with this, it is possible to realise the decoupling of the electronic wave function from the wave function of the nuclei. Another consequence of these approximations is that the kinetic energy operator of the nuclei does not act on the electronic wave function[17]. To summarise, the fixed nuclei approximation results in the nucleus-nucleus potential term being constant and the kinetic energy of the nuclei being zero.

By separating the wave functions of the nuclei and electrons, we can write the electronic Hamiltonian independently and cosidering that the nucleus-nucleus potential is constant, i.e. the nuclei are stationary with respect to the electrons. Therefore, $\vec{R}$ will be constant and the electronic Hamiltonian can be written as follows:

$$\hat{\mathcal{H}_e} = \hat{T_e} + \hat{V_{ne}} + \hat{V_{ee}}, \tag{2.7}$$

In other words, it is necessary to calculate only the kinetic and potential energy of the electrons, and calculate the nucleus-electron potential. Therefore, due to the previous approximations, the kinetic energy term of the nucleus is null and the Hamiltonian to be written is only the electronic one.

So far, the theory has been developed using the formalism that depends on the electronic wave function. However, it is known that the electronic wave function depends on the position vector $\vec{r}$ and is a complex function of 3N variables. Thus, $\Psi(\vec{r_1}, \vec{r_2}, ..., \vec{r_N})$[18], is more difficult to work with than, for example, electronic density, so we'll look for a simple way to solve the problem using electronic density. The next section will explain how the Hohenberg and Kohn theorems transform the problem of finding the wave function into a problem of finding the electronic density of the ground state.

2.3 Hohenberg and Kohn's Theorems

To construct the DFT, Hohenberg and Kohn proposed two theorems. These will be announced below:

Theorem 1: The external potential $V(\vec{r})$ is a unique functional of the electronic density $\rho(\vec{r})$.

Theorem 2: The ground state energy $E[\rho_0]$ is minimal for exact density $\rho_0(\vec{r})$.

The first theorem guarantees that from the charge density we can determine the external potential and subsequently the observables of the system. The second theorem states that the energy of the ground state corresponds to the minimum of the energy functional $E_0[\rho_0(r)]$, obtained from the exact density of the ground state $\rho_0(r)$. Any different density will lead to a higher energy state than the fundamental state[19]. So, if you have the charge density given by the first theorem, i.e. the charge density of the ground state, you can then find the energy of the ground state, and the second theorem guarantees that this energy is minimal, so any other energy will be greater.

The Hohenberg and Kohn theorems are fundamental, because it is known that to determine the observables of a system in quantum mechanics the usual form is as follows:

$$v(\vec{r}) \rightarrow \hat{H} \rightarrow \Psi(\vec{r_1}, \vec{r_2}, ..., \vec{r_N}) \rightarrow \langle \Psi | A | \Psi \rangle, \tag{2.8}$$

where A is any operator. However, using theorems, we can ensure that the direction presented is invertible, i.e:

$$\rho_0(\vec{r}) \rightarrow \Psi(\vec{r_1}, \vec{r_2}, ..., \vec{r_N}) \rightarrow v(\vec{r}) \rightarrow \langle \Psi | A | \Psi \rangle \tag{2.9}$$

Thus, the problem that used to be determined by starting from the potential to describe the Hamiltonian and then the wave function and ending by finding the observables. With the theorems, the problem becomes finding the exact density that will describe the potential and finally calculating the observables.

Knowing the implications of the theorems, a proof by absurdity will be established. It will be proved that given two distinct potentials V_{ext} and V'_{ext}, they must differ by more than one additive constant, otherwise they will lead to equal states. Considering the electronic Hamiltonian:

$$\hat{\mathcal{H}}_e = \hat{T}_e + \hat{V_{ee}} + \hat{V_{ext}} \tag{2.10}$$

Writing the Schrödinger equation for each state:

$$(\hat{T}_e + \hat{V_{ee}} + \hat{V_{ext}})\Psi_0 = \epsilon_0 \Psi_0 \tag{2.11}$$

$$(\hat{T}_e + \hat{V_{ee}} + \hat{V'_{ext}})\Psi'_0 = \epsilon'_0 \Psi'_0 \tag{2.12}$$

Knowing that ϵ_0 and ϵ'_0 are energy eigenvalues and assuming that the states are equal, that is, $\Psi_0 = \Psi'_0$. By subtracting the equation 2.12 from 2.11, it follows that:

$$(\hat{V_{ext}} - \hat{V'_{ext}})\Psi_0 = (\epsilon_0 - \epsilon'_0)\Psi_0 \tag{2.13}$$

In other words, for our assumption to be true ($\Psi_0 = \Psi'_0$), the potentials have to differ by at most one additive constant, which violates the proposal. Thus, it can be concluded that if V_{ext} and V'_{ext} are different, then so are the corresponding wave functions.

With this, the second of the theorem's proofs can be made. Given two states, they will not lead to the same electron density. The proof will be established by contradiction, assuming $\rho_0(\vec{r}) = \rho'_0(\vec{r})$, it will be shown that this assumption leads to an absurdity.

Consider $\hat{H}$ and $\hat{H'}$, and the potentials V_{ext} and V'_{ext}, respectively. According to the Rayleigh-Ritz variational principle:

$$E_0 = \langle \Psi | \hat{H} | \Psi \rangle < \langle \Psi' | \hat{H} | \Psi' \rangle. \tag{2.14}$$

Since Ψ' is not an eigenfunction of $\hat{H}$, you can rewrite $\hat{H}$ as a function of $\hat{H'}$:

$$\hat{H} = \hat{H'} + \hat{V}_{ext} + \hat{V'}_{ext}. \tag{2.15}$$

Subtracting the equation 2.15 from 2.14,

$$E_0 = E_0' + \int d\vec{r}\, \rho_0'(\vec{r})[\hat{V_{ext}} - \hat{V_e'}xt]. \tag{2.16}$$

Similarly, you can calculate $\hat{H}'$,

$$E_0' = E_0 + \int d\vec{r}\, \rho_0(\vec{r})[\hat{V_{ext}} - \hat{V_e'}xt]. \tag{2.17}$$

Adding 2.16 and 2.17,

$$E_0 + E_0' < E_0' + E_0 + \int d\vec{r}\, (\rho_0(\vec{r}) - \rho_0'(\vec{r}))[\hat{V_{ext}} - \hat{V_e'}xt]. \tag{2.18}$$

If you replace the initial assumption $\rho_0(\vec{r}) = \rho_0'(\vec{r})$ in 2.18, you get an absurdity:

$$E_0 + E_0' < E_0' + E_0. \tag{2.19}$$

With this, we can calculate:

$$E[\rho] = \langle \Psi_0[\rho] | \hat{T}_e + \hat{V_{ee}} + \hat{V_{ext}'} | \Psi_0[\rho] \rangle, \tag{2.20}$$

as long as the energy of the ground state is determined by the exact density, then the Rayleigh-Ritz principle guarantees that $E_0 < E[\rho]$, i.e. the density value that minimises the energy is just the exact density of the ground state. Therefore, to find the energy of the ground state, the electronic density must be varied until the minimum is found. The equation 2.20 can be rewritten as:

$$E[\rho] = F_{HK} + \int d\vec{r}\, \hat{V}(\vec{r})\rho(\vec{r}). \tag{2.21}$$

The quantity F_{HK} is a universal functional for all N-particle systems. These systems have F_{HK} with the same functional form.

Although the Hohenberg and Kohn theorems are fundamental to density functional theory, they do not provide a practical way of doing the calculations to find the density of the ground state of the systems of interest. To this end, we will show in the next section a practical way to solve the problem using the Kohn-Sham equations.

2.4 Kohn-Sham Equations

The Konh-Sham equations aim to map N interacting bodies to N non-interacting one-body problems[16].

Considering the Honhenberg and Kohn theorem, the energy functional can be written as:

$$E_s[\rho] = T_s[\rho] + \int \hat{v_s}(\vec{r})\rho(\vec{r})\, d\vec{r}\ . \tag{2.22}$$

Thus, considering the system as non-interacting, $T_s[\rho]$ is different from $T[\rho]$.

To write down the electronic density of the non-interacting system, we do the following:

$$\rho_s(\vec{r}) = \sum |\phi_i \vec{r}|^2, \tag{2.23}$$

with this, we can write the Schrödinger equation for the system of N independent particles, i.e. the system we wish to describe, i.e. the non-interacting system, as:

$$\left(\frac{\hbar}{2m} \nabla_i^2 + \hat{V}^{KS}(r) \right) \phi_i = \epsilon_i \phi_i. \tag{2.24}$$

Note that the eigenvalue equation 2.24 is of the Schrödinger type for a single electron. The 2.24 equation has a term known as the Kohn-Sham potential; this term depends on the effective or external potential. The external potential will be the one that reproduces the same density as the fundamental state of the system under study, i.e. it will reproduce the same density as the problem of interacting particles. The effective potential that best reproduces this density is therefore sought in order to solve the 2.24 equation. For this solution, the self-consistent cycle will be needed, which will search for the density that will define the fundamental state of the problem. The Kohn-Sham potential can be written as:

$$\hat{V}^{KS}(r)|\rho(r)| = \hat{V}_{ext}(r) + \hat{V}_H|\rho(r)| + \hat{V}_{XC}|\rho(r)|, \tag{2.25}$$

i.e. the sum of all the potentials involved in the system, which are the potential external to the electronic system, which refers to the potential due to the nuclei of the system and external electric and magnetic fields[18], added to the Hartree potential, and finally the correlation and exchange potential, which refers to everything we don't know in our problem.

To map the non-interacting system, an algebraic manipulation of the system's energy will be carried out, i.e:

$$E = T_e + U_{e-e} + V_{ext} + T_s - T_s + U_s - U_s, \tag{2.26}$$

these quantities are directly related to the quantities in the 2.25 equation, so it can be seen that the correlation and exchange energy arises naturally through the maninulation carried out, i.e. the 2.26 equation can be rearranged,

$$E = T_s + V_{ext} + T_e + U_{e-e} - T_s - U_s, \tag{2.27}$$

and thus the correlation and exchange energy can be identified and written as:

$$E_{xc}[\rho] = F_{HK} - T_s[\rho] - U_s[\rho]. \tag{2.28}$$

.

As already mentioned, the whole unknown of our problem lies in this term, the correlation and exchange term, which holds all the information that cannot be quantified

immediately. If it were possible to determine this term exactly, density functional theory would be an exact theory, but this is not possible precisely because of the difficulty in determining this term.

It is known that the electronic density ρ that minimises the functional $E[\rho]$ is that of the fundamental state[18]. Thus, the minimum condition is given by $\delta E[\rho] = 0$. However, this condition must be restricted to a bond, which will be the number of electrons in the system that will be given correctly, so you can write the bond for the number of electrons as follows:

$$\int d\vec{r}\rho(\vec{r}) - N = 0, \tag{2.29}$$

this link must be included in the equation of the functional to be minimised, i.e. by including the link via a Lagrangian multiplier, which will be the chemical potential μ. Therefore:

$$E[\rho(\vec{r})] - \mu\left[\int d\vec{r}\rho(\vec{r}) - N\right], \tag{2.30}$$

You can then search for the minimum of the equation 2.30 by doing:

$$\delta\left\{E[\rho] - \mu\left[\int d\vec{r}\rho(\vec{r}) - N\right]\right\} = 0, \tag{2.31}$$

As the value of N is constant, the equation can be written as,

$$\delta\left\{E[\rho] - \mu\left[\int d\vec{r}\rho(\vec{r})\right]\right\} = 0, \tag{2.32}$$

with this, the functional derivative with respect to density can be performed, thus finding the following terms:

$$\frac{\delta E[\rho]}{\delta\rho(\vec{r})} - \mu = \frac{\delta[T_s]}{\delta\rho(\vec{r})} + e^2\int d\vec{r}\frac{\rho(r)}{|r-r'|}V_{ext} + V_{xc} = 0, \tag{2.33}$$

therefore,

$$\mu = \frac{\delta E[\rho]}{\delta\rho(\vec{r})} = \frac{\delta T_s}{\delta\rho(\vec{r})} + V_s(\rho(\vec{r})), \tag{2.34}$$

and the value V_{xc} is exactly the correlation and exchange potential which is related to the correlation and exchange energy through the equation:

$$V_{xc} = \frac{\delta E[\rho]}{\delta\rho(\vec{r})}, \tag{2.35}$$

with this, you can rewrite the 2.25 equation for the Kohn-Sham potential (effective potential) where $V_{ks} = V_s$,

$$V_s(\vec{r}) = V_{ext} + e^2 \int d\vec{r} \frac{\rho(r)}{|r - r'|} + V_{xc} \tag{2.36}$$

The previous equation is called the Kohn-Sham equation and we seek to solve it in self-consistent form as will be discussed in the next section.

2.5 The Self-Consistent Cycle

The DFT is implemented using the self-consistent cycle as can be seen in the figure **??**:

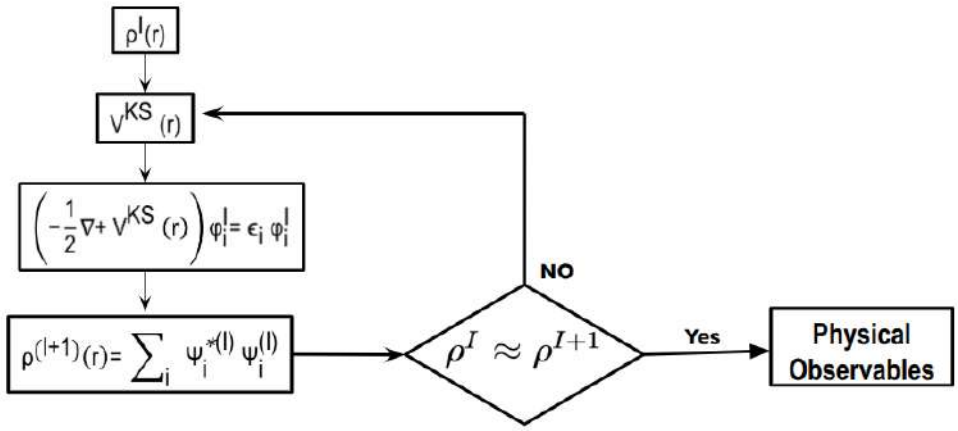

Figure 4 – Representative flowchart of the self-consistent cycle.

In the figure 4 we present the self-consistent cycle, we have an initial electronic density, we calculate the potential from the equation 2.25 and then we find the eigenvalues using the Kohn-Sham equation. The associated energy eigenvalues are found, then the wave function is calculated, then the electron density, and finally the input and output densities are compared; if the difference is less than a certain tolerance, we move on to calculate the physical observables. If not, this density will be used for the next step in the cycle until the condition is met.

2.6 Approximations for the Correlation and Exchange Potential

Throughout this section, the formalism that makes it possible to study many-body systems was developed and discussed. Using the Bohr-Oppenhaimer approximation, it was possible to decouple the nuclear Hamiltonian from the electronic Hamiltonian and decouple the wave functions, making it possible to solve the electronic problem separately. Using the Hohenberg and Kohn theorems, it was possible to verify that once the electronic density of the ground state is found, it is possible to describe the properties of the system.

Finally, the Khon-Sham equations make it possible to map a system of N interangent particles onto a system of N non-interangent particles and how the two systems can be related. However, although the Khon-Sham formalism is exact, the term for the correlation and exchange potential needs to be approximated. This section will discuss how this potential can be approximated, always remembering that this potential is related to the correlation and exchange energy through the 2.35 equation.

In the literature, it is possible to verify the use of two main approximations, the Local Density Approximation (LDA) and the Generalised Gradient Approximation (GGA)[16].

2.6.1 Local Density Approximation - LDA

The LDA is one of the simplest approximations used. The energy density of correlation and exchange per electron is assumed to be equal to the density of a homogeneous electron gas. Each point has its own density and this density is the same as that of the dot[16]. The correlation and exchange energy depends only on the r point, so it doesn't take non-local terms into account. Therefore, the equation below refers to the approximation for the correlation and exchange energy with local dependence only on r.

$$E_{XC}[\rho(\vec{r})] = \int \rho(\vec{r}) e_{XC}[\rho(\vec{r})]\, d\vec{r} \tag{2.37}$$

the term, ϵ_{xc} is exactly the correlation and exchange density, which in this case of LDA is considered homogeneous and can be written as:

$$\epsilon_{xc} = \epsilon_{xc}^{hom}(\vec{r})[\rho(\vec{r})]. \tag{2.38}$$

The correlation and exchange density is considered to be homogeneous, so LDA is expected to reproduce well systems with uniform or smoothly varying electronic density. In other words, the LDA is capable of describing various solids, including metallic or covalent systems. The next section presents an approximation for non-local terms.

2.6.2 Generalised Gradient Approximation - GGA

For materials in which the electronic density does not vary smoothly, the LDA is no longer a good approximation, in which case we will use the GGA. There are two types of generalised gradient approximation (GGA): semi-empirical, which are based on experimental data, and non-empirical, which are theoretical. The GGA correlation and exchange functional used is the PBE (Perdew, Burke and Ernzerhof)[?], which belongs to the class of theoretical functionals.

The approximation for the GGA correlation and exchange energy depends on the r point and the gradient, taking into account non-local aspects of the electronic density and contemplating the non-locality of E_{xc}. The equation can therefore be written as:

$$E_{XC}[\rho(r), \nabla\rho(r)] = \int e_{XC}[\rho(r), \nabla\rho(r)]\, dr \ , \tag{2.39}$$

The 2.39 equation shows the dependence of the e_{xc} density on both the point in space and the gradient, indicating the aforementioned dependence.

2.6.3 Projector Augmented Wave - PAW

The augmented wave projector method is used to overcome the problem that valence electron wave functions can have rapid oscillations close to the atomic nuclei due to the need to be orthogonal to the nuclear states. This can be problematic, as describing these wave functions accurately requires high spatial resolution, which significantly increases the computational cost.

However, the PAW method solves this problem by transforming these rapidly oscillating wave functions into smooth, computationally convenient forms. It does this by introducing pseudopotentials that capture the interaction of electrons with atomic nuclei effectively, allowing it to represent the properties of all electrons more efficiently. In this way, the PAW method simplifies calculations while maintaining the precision necessary to study the electronic properties of materials[20].

To solve this problem, it is necessary to construct a linear operator $\hat{T}$ that when applied to an auxiliary and smooth wave function $\tilde{\psi}_n$ will lead to the true khon-sham wave function ψ_n, this operation can be observed in the equation 2.40:

$$\psi_n = \hat{T}\tilde{\psi}_n, \tag{2.40}$$

Due to this transformation, we will have new kohn-sham equations to solve, as can be seen in the equation 2.41

$$\hat{T}^\dagger \hat{H}^\dagger \hat{T}\tilde{\psi}_n = \varepsilon \hat{T}^\dagger \hat{T}\tilde{\psi}_n. \tag{2.41}$$

Let's construct the operator so that it acts only within a certain sphere centered on each atom a, without overlapping:

$$\hat{T} = \hat{1} + \sum_a \hat{T}^a, \tag{2.42}$$

therefore, let's consider that $\hat{T}^a$ does not act outside the cutting radius r_c^a. With this, inside the sphere it is possible to expand the true wave function into other auxiliary partial wave functions ϕ_i^a each wave associated with a smooth partial wave function $\tilde{\phi}_i^a$, thus obeying the following relationship:

$$\phi_i^a = (\hat{1} + \hat{T}^a)\tilde{\phi}_i^a \Leftrightarrow \hat{T}^a\tilde{\phi}_i^a = \phi_i^a - \tilde{\phi}_i^a, \tag{2.43}$$

Knowing that T^a does not act outside the sphere, the following equality can be established, but with restriction on the radius, that is:

$$\forall a; \phi_i^a(\mathbf{r}) = \tilde{\phi}_i^a(\mathbf{r}), r > r_c^a, \tag{2.44}$$

Taking the partial wave functions as forming a basis, we can expand them as:

$$\tilde{\psi}_n = \sum_i P_{ni}^a \tilde{\phi}_i^a, \quad r < r_c^a, \tag{2.45}$$

with this, we will have:

$$\psi_n = \hat{T}\tilde{\psi}_n = \hat{T}\left(\sum_i P_{ni}^a \tilde{\phi}_i^a\right) = \sum_i P_{ni}^a \phi_i^a, \quad r < r_c^a, \tag{2.46}$$

the expansion coefficients are given by the projection on each sphere, that is:

$$P_{ni}^a = \tilde{p}_a^i || \tilde{\psi}\rangle, \tag{2.47}$$

therefore, $\tilde{p}_i^a$ are called projector functions, such functions must satisfy the completeness relation, that is:

$$\sum_i \tilde{\phi}_i^a \tilde{p}_i^a = 1 \Rightarrow \tilde{p}_i^a || \tilde{\phi}_j^a\rangle = \delta_{ij}, \tag{2.48}$$

Therefore, using this relationship:

$$\hat{T}^a = \sum_i \hat{T}^a \phi_i^a \tilde{p}_i^a = \sum_i (\phi_i^a - \tilde{\phi}_i^a)\tilde{p}_i^a \Rightarrow \hat{T} = \hat{1} + \sum_a \sum_i (\phi_i^a - \tilde{\phi}_i^a)\tilde{p}_i^a. \tag{2.49}$$

Therefore, the new Kohn-Sham wave function will be given by:

$$\psi_n(\mathbf{r}) = \tilde{\psi}_n(\mathbf{r}) + \sum_a \sum_i \left(\phi_i^a(\mathbf{r}) - \tilde{\phi}_i^a(\mathbf{r})\right) \tilde{p}_i^a || \tilde{\psi}_n\rangle \tag{2.50}$$

Such decomposition allows the Kohn-Sham functions to be handled by smooth auxiliary functions throughout the space and oscillating functions close to the nucleus[20].

We will use the PAW pseudopotential method for DFT calculations used through the Quantum ESPRESSO software, which we will talk about in the next section.

Density functional theory describes the behaviour of many atoms with some accuracy. The many-body problem is unsolvable analytically, but the theory has been implemented computationally in order to obtain a numerical solution to the problem. The programme QUANTUMEXPRESSO has an implementation of DFT, and aims mainly to solve the Konh-Sham equations described in the theory in a self-consistent way, this process is called a self-consistent cycle[15].

2.7 Quantum ESPRESSO

Quantum ESPRESSO is an open source software suite used to perform electronic structure calculations of materials using density functional theory (DFT). It relies on plane waves, pseudopotentials, and first-principles methods to calculate the electronic properties of solid materials, molecules, and surfaces. The name "ESPRESSO" is an acronym for "opEn Source Package for Research in Electronic Structure, Simulation, and Optimization". Quantum ESPRESSO is widely used in materials research, condensed matter physics and theoretical chemistry due to its precision and flexibility[21].

In this work, we use quantum espresso to perform the self-consistent cycle, but we will not use it to calculate material properties. We will use PAOFLOW for this purpose, whereas espresso uses plane waves and would have a high computational cost to calculate the orbital hall conductivity property. In this case, we will use the espresso quantum Halmitonian, but to calculate the properties we will use the PAOFLOW that we discuss in the next section.

2.8 PAOFLOW

PAOFLOW is a software tool designed to help with modeling and predicting properties of quantum materials. It is used to construct tight-binding like Hamiltonians from self-consistent electronic wave functions, via projections onto a set of atomic orbitals. This approach allows you to obtain information on the electronic structure of materials, as well as to predict a variety of material properties[22].

This software is particularly useful for investigating quantum materials and devices, where properties cannot be easily accessed using standard first-principles packages. PAOFLOW offers a way to calculate these properties efficiently using more detailed models based on electronic wave functions.

In the previous section, we discussed the use of quantum espresso to carry out the self-consistent cycle, PAOFLOW comes in to calculate the properties of the materials. In the case of this work, we will use it to calculate the orbital hall conductivity, which is calculated by solving the equation 1.2.

In figure 5 a workflow is shown on how to use quantum espresso and PAOFLOW. So far, it has been discussed how the calculation of material properties can be carried out using DFT and PAOFLOW. However, as previously discussed, the main interest is to explore phase space with many combinations of chemical elements to form possible insulating materials and predict the value of orbital hall conductivity. Starting in the next chapter, we will begin to discuss the techniques used to build models capable of making predictions. In the following two chapters, we discuss algorithms capable of exploring data and how we use these algorithms within a process capable of building efficient models.

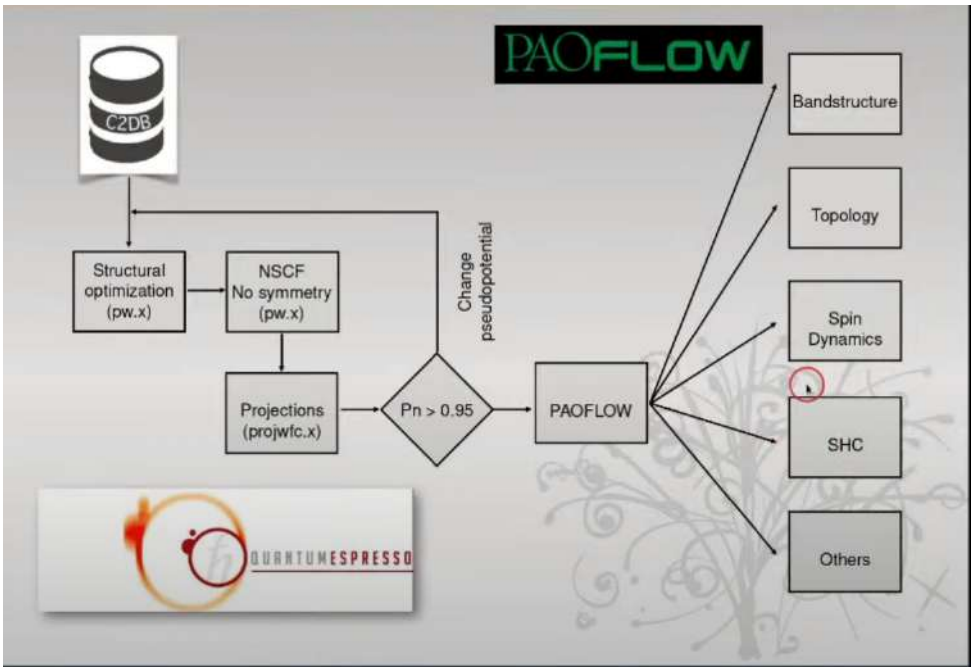

Figure 5 – Workflow depicting the calculation of OHC with PAOFLOW.

3 Artificial Intelligence

In this chapter, we discuss the concept of learning algorithms and outline the key differences between Artificial Intelligence, Machine Learning, and Deep Learning. Additionally, we clarify the main distinctions between different types of machine learning tasks.

Although we use these terms interchangeably, Artificial Intelligence (AI) is any system that seeks to reproduce human intelligence in some way, regardless of how it happens. Therefore, we can visualize AI as a large area that encompasses all the others, it is possible to visualize these areas with the diagram in figure 6.

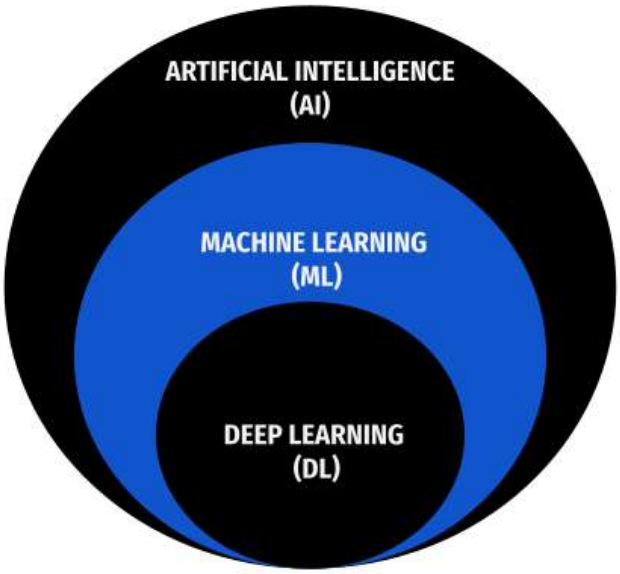

Figure 6 – Hierarchy of Artificial Inteligence, Machine Learning and Deep Learning.

Note that, Artificial Intelligence (AI) is a broader field of study that involves both other areas and aims to create machines and systems capable of performing tasks that normally require human intelligence. These tasks can include speech recognition, computer vision, automated decision making, language translation, robots, and many other systems.

The first frontier of AI is Machine Learning (ML), which is a sub-field that focuses on teaching machines to learn from data. Instead of programming specific rules for each task, ML algorithms are used that use the data to learn patterns and relationships. Some of these algorithms and how they work will be explored in the following sections.

The next frontier of AI, Deep Learning (DL) is a subfield of ML that uses deep artificial neural networks (ANN) to learn representations of data. ANNs are composed of layers of interconnected artificial neurons that process input information and generate the corresponding outputs. Deep neural networks have many layers and many neurons, allowing them to learn complex and abstract representations of data. ANNs are trained with large amounts of data and, in general, require a lot of computational resources for training. ANNs are used in many AI applications, including computer vision, natural language processing, and speech recognition.

There are a few main types of machine learning tasks: supervised, unsupervised, and reinforcement. In the next section, we'll explore the main differences between each type of learning and the main algorithms used for each task of machine learning.

3.1 Machine Learning Algorithms

Machine learning algorithms are built to perform specific tasks, so we can synthesize into two main types, algorithms that are built to work with predictive or descriptive tasks. Algorithms that work with predictive tasks are used when looking to make predictions from new data whose target attribute value is unknown. Algorithms built for descriptive tasks are used when we just want to describe the relationship between data. Learning tasks can be further subdivided into supervised, unsupervised and reinforcement learning, these divisions end up being related to the previous divisions, supervised learning tasks are generally related to predictive tasks and unsupervised is related to descriptive tasks.

Supervised learning involves feeding some algorithm labeled examples so that it can learn to match the input data with the correct outputs. Unsupervised learning involves feeding the algorithm unlabeled data and allowing it to find patterns and relationships without prior guidance. Reinforcement learning involves training the algorithm to make decisions in a reward/punishment environment, with the goal of maximizing rewards and minimizing punishments. We can visualize the types of learning according to the tree in figure 7.

For each type of learning, we still have some tasks that can be performed, for the supervised type, we can perform classification and regression tasks, for the unsupervised type, we have several types of tasks, for example, grouping tasks and association rules.

In the next sections, we will dedicate ourselves to better understand the types of learnig through some examples and understend the operation of some of the main machine learning algorithms for each type of learning.

3.1.1 Supervised Learning

In supervised learning, we can have classification or regression tasks. In classification we are interested in predicting a categorical attribute, for example, predicting whether

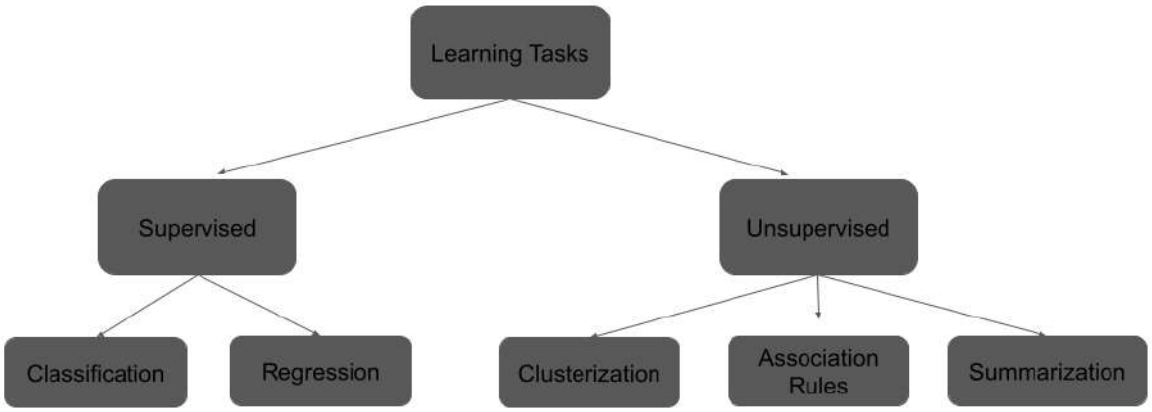

Figure 7 – Types of machine learning tasks.

a material is a conductor or an insulator. In the case of the regression task, we want to predict the value of an attribute, for instance, the band gap(distance betwen conduction band and valence band) value of a certain material.

In figure 8 we can see how we structure a supervised learning problem. For example, trying to predict whether a material is a conductor or an insulator.

1. We have a material and its classification (conductor or insulator);
2. We look for some attributes that represent these materials.
3. We use some algorithms that will learn the relationships between the attributes and the output, that is, what are the values of the features that are related to the output.
4. After the trained algorithm, we will have a model which we can use to make new predictions, that is, from new materials we can predict whether it will be insulating or conducting.

Building a machine learning model is not a trivial process, it involves different phases to ensure that we build a model that really makes sense and that is capable of making correct predictions, however, in this chapter we will dedicate ourselves to understanding how models work and in the next chapter we will dedicate ourselves to understanding how to build a model.

Machine learning algorithms can be divided into four major groups, distance-based algorithms, search-based algorithms, probabilistic algorithms, and optimization-based

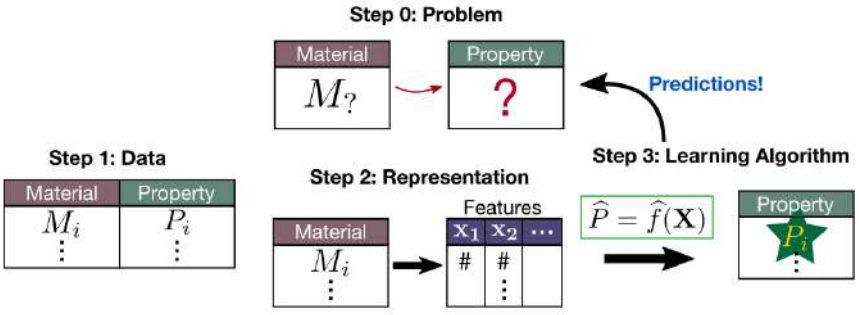

Figure 8 – Structure of supervised learning task. Source: Article "From DFT to machine learning: recent approaches to materials science–a review".(1)

algorithms. This division refers to how the algorithms seek to find the relationships between the data.

3.1.1.1 Distance Based Methods

Distance-based methods consider the proximity between data to make new predictions, that is, they use the hypothesis that similar data tend to be close in the data space. This is also the basic premise for one of the most basic machine learning algorithms we will discuss next, K-nearest neighbors (KNN.)

3.1.1.1.1 K-Nearest Neighbors(KNN)

The K-Nearest Neighbors algorithm was one of the first algorithms to be developed, its initial ideas began in 1951 with the article by Evelyn Fix and Joseph Hodges(23) however the ideas expanded with the article by Thomas Cover(24).

The KNN Algorithm is part of a class of algorithms called "lazy algorithms", as this algorithm does not directly build a model, but uses a strategy to query the training data(25).

To understand the KNN algorithm let's observe the figure 9, in this figure notice that we have points from two different classes(Metal/Insulator). What the KNN algorithm does is use the nearest neighbors to determine which class this new data belongs to. The K parameter is the value that needs to be determined in this algorithm, as it is based on your choice that the algorithm will classify the new data. For example, considering K=1, we will classify the new data according to the first nearest neighbor, that is, through a chosen distance measure,we will use the point that is closest to the new data to classify to which class it will belong.

The basic hypothesis of the KNN algorithm is that similar data tend to be closer to each other in the data space (25). Note in figure 9 which we use two variables to observe the materials that are metal and insulators, using the KNN algorithm we will classify a new data from the number of first neighbors that we will consider in our model and we

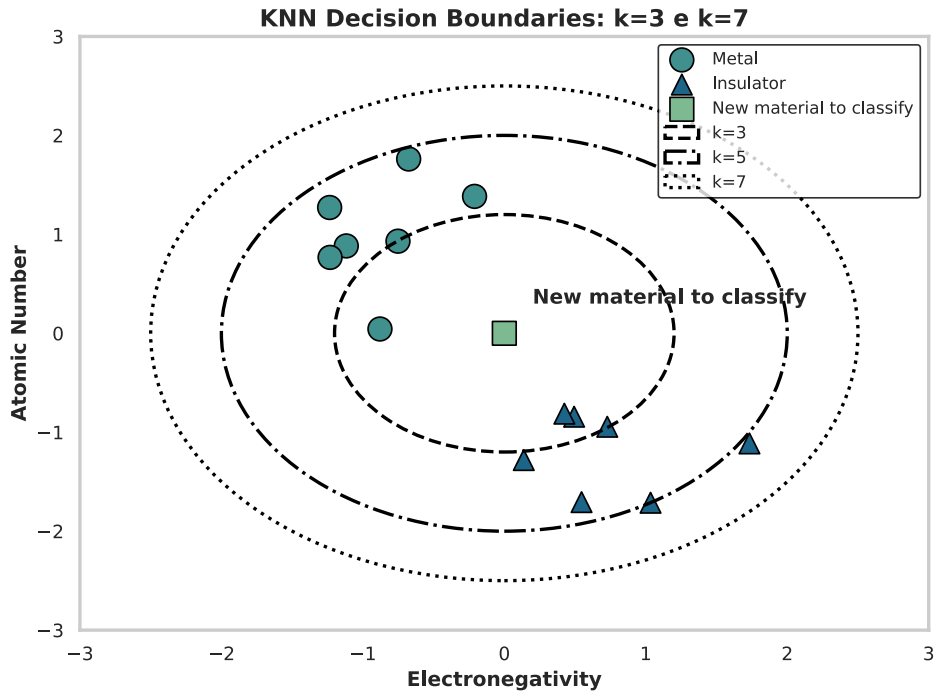

Figure 9 – Representation KNN algorithm decision boudaries.

will use one of the distance measures described in the equations 3.1 what is the euclidean distance, 3.2 what is the manhattan distance, 3.4 what is the minkovisk distance and 3.3 what is the chebyshev distance.

$$d_{ij} = \sqrt{\sum_{k=1}^{p}(x_{ik} - x_{jk})^2} \tag{3.1}$$

$$d_{ij} = \sum_{k=1}^{p} |x_{ik} - x_{jk}| \tag{3.2}$$

$$d_{ij} = \max_{k=1}^{p} |x_{ik} - x_{jk}| \tag{3.3}$$

$$d_p(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}} \tag{3.4}$$

To determine the best distance to use, it is necessary to carry out experiments with the data and analyze which distance will perform best in the algorithm's metrics. We will understand better in the machine learning life cycle chapter how to calculate these performance metrics.

3.1.1.2 Search Based Methods

The machine learning problem can be formulated as a search problem in a space of possible solutions [25]. That is, from a space of hypotheses and a function of evaluation of

hypotheses it is possible to search in such space the hypothesis that best fits the solution of the problem. In this section, we will dedicate ourselves to understanding how decision tree-based methods work, decision tree-based models use the divide-and-conquer method, that is, a complex problem is divided into simpler problems and thus the solution of the problems simpler ones can be combined to solve the more complex problem.

When we refer to data space, we are looking at the set of attributes we have available to perform a given task. That is, if we want to understand how a material is classified between metal and insulator, let's look at the data we have available to make the classification and train an algorithm to make this prediction through the construction of a decision tree, at the end of training algorithms, we will have a decision tree built that can make new predictions.

3.1.1.2.1 Decision Tree

Assuming we want to predict whether a given material is an insulator or a metal, in this case, unlike the distance-based approach, we seek to maximize the 3.7 equation, which we call the information gain equation, and minimize the 3.5 equation.

Through the equation 3.5 we can calculate the total entropy of my data set where p_i is the probability of the target attribute, for our example we would have to calculate p_1, as the probability of being an insulator and p_2 as the probability of being a metal, this would be done by contacting the total of insulators and the total of metals in our dataset. After that we would calculate the E(S) making the total sum which would give us the entropy of the data set,

$$E(S) = -\sum_{i=1}^{n} p_i \log(p_i). \tag{3.5}$$

After calculating the total entropy of the target attribute, we would now have to calculate the entropy of each attribute, that is, calculate the value of $E(S, Atr)$ which is given by dividing $|S_\nu|$ which is the number of observations of a given attribute by $|S|$ which is the total number of observations of the dataset and $E(S_\nu)$ which is calculated using the equation 3.5. Once this is done, we have the calculation of the entropy of an attribute through the following equation:

$$E(S, Atr) = \sum_{\nu \in Domínio(Atr)} \frac{|S_\nu|}{|S|} E(S_\nu) \tag{3.6}$$

After calculating the entropy of each attribute, we can calculate the information gain for a given attribute which is given by the equation 3.7,

$$Gain(Atr) = E(S) - E(S, Atr), \tag{3.7}$$

This calculation is done repeatedly, the tree node is determined when we maximize the information gain with that attribute.

To build the decision tree, it is necessary to understand that the root node is the one that initiates the construction of the tree, the leaves are the predictions of the model and the edges of the tree represent the predicate of the attribute, that is, what leads to the construction of that leaf or inner node.

To build the root node, it is first necessary to calculate the entropy of the classes that we want to predict, that is, in the example of table XX we will calculate the entropy of the classes through the equation 3.5. After performing this calculation, we already have the first factor of the information gain equation, so to label the root node it is necessary to calculate the value of the 3.6 equation for each attribute, the one that obtains the highest value for 3.7 will be what we will label as the root node of the tree that will be built. After using a given attribute as the root node, we will no longer use it to calculate the information gain, we will do the same as we did for the root node for the other available attributes. The idea of using the information gain to build a given decision tree is implemented by the Iterative Dichotomiser 3 algorithm (ID3) [(26)]. However, the ID3 algorithm for the decision tree may have some problems depending on the available data set, such as attributes such as a lot of data of a certain class. To solve this problem, there are other algorithms, such as algorithm C4.5[(27)], in this implementation, the gain rate measure is used, which can be seen in the equation 3.8.

$$GainRatio(Atr) = Gain(Atr)/Split.info(Atr) \tag{3.8}$$

In the equation 3.9, the $|D_j|$ and $|D|$ are respectively the number of base D elements with a given attribute and the number of base D elements.

$$Split.info(Atr) = \sum_{j=1}^{\nu} \frac{|D_j|}{|D|} \cdot \log_2 \left(\frac{|D_j|}{|D|} \right) \tag{3.9}$$

Another decision tree-based algorithm that is currently widely used is the Classification and Regression Trees (CART) algorithm [(28)], which is based on the Gini Index measure, that is, it aims to measure the degree of impurity of a given class distribution through the equation refGI. That is, the value of the equation 3.7 is calculated for the target attribute, then the impurity value of the partitions is calculated for each independent attribute through the equation 3.11 and then the reduction is calculated impurity through the 3.12 equation, the most appropriate value to label the current node is the one that most reduces the degree of impurity.

$$Gini(D) = 1 - \sum_{i=1}^{m} {p_i}^2 \tag{3.10}$$

$$Gini_{Atr}(D) = \sum_{v=1}^{V} \frac{|D_v|}{|D|} Gini(D_v) \tag{3.11}$$

$$\Delta Gini(Atr) = Gini(D) - Gini_{Atr}(D) \tag{3.12}$$

Therefore, it is noted that the decision tree algorithm has, in all its possible implementations, an associated cost function which is sought to be minimized by exploring the data space. The decision tree algorithm has some other ways to be explored, in this section we discuss how to build a model through a decision tree, however a set of different trees can be built and together to make the prediction.

The idea of using a single decision tree to build a representative model of the data set may be interesting depending on the complexity of the problem we are trying to model, however, there are other tree-based algorithms that can work with problems of greater complexity.

3.1.1.2.2 Bagging and Boosting Algorithms

The idea of joining several predictors in order to improve machine learning models comes from the principle that many predictors are better than just one. Thus, the so-called ensemble algorithms are several algorithms trained on different groups of the dataset that together will be able to make a certain prediction.

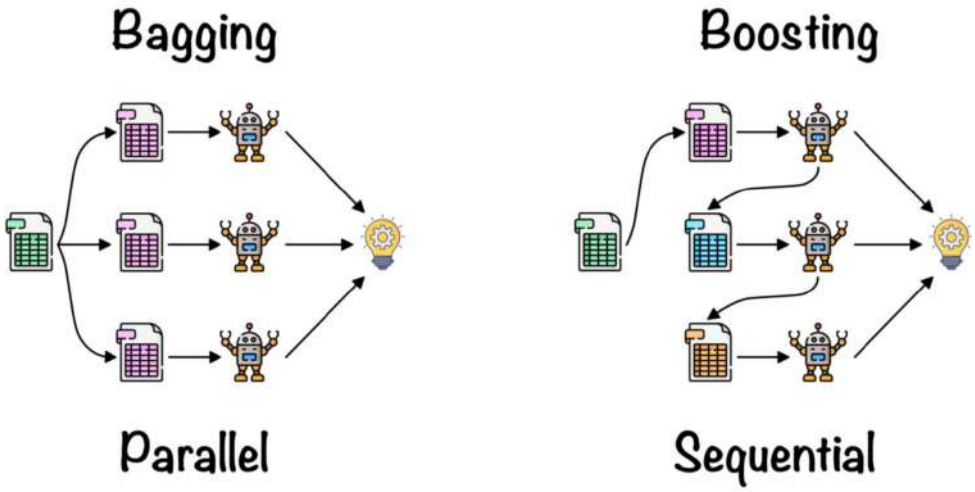

Figure 10 – Bagging vs Boosting.

Bagging (Bootstrap Aggregating) is a ensemble learning technique that aims to reduce the model variance, that is, to improve the generalization capacity of the model for new data different from the training data. The idea behind bagging models is to combine the predictions of multiple classifiers trained on different subsets of the original dataset. Bagging uses the bootstrap method to create subsets of data through sampling with replacement. As in figure 10, each classifier is trained on a different subset of data in parallel and the predictions are combined by polling (in the case of classification) or by averaging (in the case of regression).

$$\hat{y}_{bagging} = mode(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n) \tag{3.13}$$

Boosting is another ensemble learning technique that aims to improve model performance by combining weak classifiers sequentially in figure **??** you can see the illustration of how boosting algorithms work, each classifier is trained to correct the errors made by the previous classifier. Boosting assigns weights to data instances, giving more importance to instances that were misclassified by the previous classifier, which is why it is considered sequential. With this, the classifiers are trained iteratively, updating the weights of the instances in each iteration, and the final predictions are made through the weighted combination of the classifiers, as in the 3.14 equation.

$$\hat{y}_{boosting} = \sum_{i=1}^{n} w_i \cdot \hat{y}_i \tag{3.14}$$

The use of bagging and boosting algorithms follows the combination of several other algorithms to build models and make predictions, an example of bagging algorithms is the random forest which we will see in detail in the next section.

3.1.1.2.3 Random Forest

Decision Tree is an algorithm that explores the data space using just one tree to build the model. Random Forest is an ensemble learning algorithm that combines multiple decision trees to perform classification or regression. Each tree is trained on a random subset of data (bootstrap sampling) and uses a random selection of features at each node split. The final prediction is made by averaging (in the case of regression) or voting (in the case of classification) the predictions of each tree. The way in which each tree is generated follows the same fundamentals discussed for the decision tree.

$$\hat{y}_{rf} = \frac{1}{n} \sum_{i=1}^{n} \hat{y}_i \tag{3.15}$$

3.1.1.2.4 Gradient Boosting

The gradient boosting algorithm and the XGboosting algorithm are part of the "Boosting" algorithms, that is, they are sequential algorithms. The gradient boosting algorithm was devised by Jerome H. Friedman in the early 2000s[29,30].

Gradient Boosting is a powerful machine learning technique for classification and regression problems. It works by building a predictive model step by step and correcting the errors of previous models. The main idea is to minimize a loss function through a "boosting" approach, where weak models are sequentially combined to form a strong model.

Gradient Boosting is based on the minimization of a loss function $L(y, \hat{y})$[30], which depends on the difference between the true values y and the predictions $\hat{y}$. It combines several models $h(x)$, usually decision trees, so that the final model $F_M(x)$ is given by:

$$F_M(x) = F_0(x) + \sum_{m=1}^{M} \gamma_m h_m(x) \tag{3.16}$$

Where:

- $F_0(x)$ is the initial model (an initial guess, usually the mean of the y values in the case of regression),
- $h_m(x)$ is the model fitted at the m-th iteration,
- γ_m is a weighting factor or learning coefficient fitted to each model.

The central idea of Gradient Boosting is to iteratively minimize a loss function $L(y, F(x))$ as in equation 3.17. The updated model in each iteration is adjusted to reduce the residual errors of the previous prediction.

$$\text{Objective}: \quad \min_{F} \sum_{i=1}^{N} L(y_i, F(x_i)) \tag{3.17}$$

Where y_i is the true value and $F(x_i)$ is the model prediction for the i-th sample.

To find the next model $h_m(x)$ at each iteration, we compute the gradient of the loss function with respect to the previous model predictions $F_{m-1}(x)$. This is equivalent to computing the pseudo-residuals $r_i^{(m)}$ at the m-th iteration, we can see in the equation 3.18:

$$r_i^{(m)} = - \left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x)=F_{m-1}(x)} \tag{3.18}$$

These residuals indicate the direction in which the model needs to be adjusted to improve performance.

In the m-th iteration, we adjust the model $h_m(x)$ to approximate the pseudo-residuals $r_i^{(m)}$:

$$h_m(x) = \arg\min_{h} \sum_{i=1}^{N} \left(r_i^{(m)} - h(x_i) \right)^2 \tag{3.19}$$

After fitting the model $h_m(x)$, we update the final model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x) \tag{3.20}$$

Here, γ_m is the learning step or coefficient, which is adjusted to minimize the loss function along the direction $h_m(x)$:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)) \tag{3.21}$$

This process is repeated for M iterations, adding successively weak models to improve the predictions of the final model. The final model after M iterations will be:

$$F_M(x) = F_0(x) + \sum_{m=1}^{M} \gamma_m h_m(x) \tag{3.22}$$

To summarize the process of building the final model, we can summarize the algorithm in the following steps:

1. Initialize $F_0(x)$ (usually with the mean of y).
2. For each iteration $m = 1, 2, \ldots, M$:
   a) Compute the pseudo-residuals: $r_i^{(m)} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]$.
   b) Fit the model $h_m(x)$ to predict $r_i^{(m)}$.
   c) Find the coefficient γ_m.
   d) Update the model: $F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$.
3. The final model is $F_M(x)$.

There are other boosting algorithms that will be used and cited throughout this work, but gradient boosting is the method that originated other algorithms such as Extreme Gradient Boosting[31], Light Gradient-Boosting Machine[32] and Catboosting[33].

3.1.1.3 Probabilistic Based Methods - Naive Bayes

Probability-based methods can include several approaches, but one of the main ones is based on Bayes' theorem. The Naive Bayes algorithm is a probabilistic classifier that is based on the Bayes Theorem and on the assumption of conditional independence between the features (or attributes) of the data set[25,34,35]. For a classification problem with a set of classes C and an attribute vector $X = (x_1, x_2, \ldots, x_n)$, Naive Bayes estimates the conditional probability of a class given an instance of attributes , using the assumption of conditional independence. Bayes' theorem is given by the equation 3.23:

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}, \tag{3.23}$$

where $P(C|X)$ is the conditional probability of the class C given the attribute vector X,$P(X|C)$ is the conditional probability of the attributes X given the class C, $P(C)$ is the a priori probability of the class C and $P(X)$ is the probability of the attributes X.

The assumption of conditional independence allows simplifying the estimate of $P(X|C)$ as the product of the individual probabilities of the attributes, assuming that they are independent of each other, resulting in the so-called "naive" (naive) Bayes. Therefore, the equation can be rewritten as:

$$P(C|X) = \frac{P(C) \cdot \prod_{i=1}^{n} P(x_i|C)}{P(X)} \tag{3.24}$$

The Naive Bayes classifier assigns a class C to an instance X by choosing the class with the highest conditional probability $P(C|X)$. This can be expressed as:

$$\hat{C} = \arg\max_{C} P(C) \cdot \prod_{i=1}^{n} P(x_i|C) \tag{3.25}$$

where $\hat{C}$ is the assigned class.

To estimate the probabilities $P(C)$ and $P(x_i|C)$, it is common to use maximum likelihood estimators or smoothed estimators, such as the Laplace estimator, to avoid zero probability problems in cases where there are no instances of a particular attribute in a particular class.

The Naive Bayes algorithm is fast and efficient, being especially suitable for datasets with many attributes and when the assumption of conditional independence is reasonable. However, this assumption can be limiting in some cases, especially when attributes are correlated. Despite this simplification, Naive Bayes generally performs well on many classification problems and is a popular choice in practical applications.

3.1.1.4 Optimization Based Methods

Although what we have discussed so far is based on optimizations, there are methods that we consider optimization-based because they use a function to minimize or maximize. One of these techniques is the support vector machine, which has its origins in statistical learning theory[25]. Another optimization-based method is artificial neural networks, which we will discuss in more detail in the appendices. In this section, we will focus on detailing support vector machines.

3.1.1.4.1 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression problems[36]. It is based on the concept of finding the hyperplane that best separates the classes in the attribute space. The objective of SVM is to find the best margin of separation between classes, maximizing the distance between the hyperplane and the closest points of each class.

The SVM is based on convex optimization and Hilbert space theory. To understand the theoretical foundation of SVM, let's consider a binary classification problem with two linearly separable data sets.

Given a training set $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where x_i is the attribute vector and y_i is the corresponding class (-1 or 1), the SVM tries to find the hyperplane that separates the two classes. The hyperplane can be represented by the equation:

$$w \cdot x + b = 0$$

where w is the weight vector that defines the direction and orientation of the hyperplane, and b is the bias term that defines the position of the hyperplane.

The distance between the hyperplane and a point x can be calculated using the decision function $f(x) = w \cdot x + b$. If $f(x)$ is greater than zero, the point x is classified as belonging to class 1; if $f(x)$ is less than zero, the point x is classified as belonging to class -1. The margin of separation between classes is defined as the smallest distance between the hyperplane and the closest training points of each class.

The objective of the SVM is to maximize the margin of separation, which can be formulated as a convex optimization problem. The standard formulation of SVM is:

$$\min_{w,b} \frac{1}{2}\|w\|^2$$

subject to:

$$y_i(w \cdot x_i + b) \geq 1, \quad \forall i = 1, 2, \ldots, n$$

This formulation seeks to find the vector of weights w and the bias term b that minimize the norm of the vector w while ensuring that all training points are correctly classified and lie beyond a separation margin.

However, it is not always possible to perfectly separate the classes linearly in the attribute space. To deal with this case, the idea of soft margin is introduced, allowing some points to stay within the margin or even be misclassified. This relaxation is controlled by the regularization parameter C. The larger the C value, the smaller the margin and the stricter the correct classification of the training points.

If the classes are not linearly separable, it is possible to map the points to a higher dimensional space using a kernel function, allowing the SVM to find a hyperplane.

3.1.2 Unsupervised Learning

Unsupervised learning has this name because we don't have the target variable in the data set, i.e. we have a set of attributes that apriori don't have a category or value associated with observations.

This is the type of machine learning task we call descriptive, because we're not trying to predict a certain value or class, but rather to describe the set of data we have.

To describe the data, we will use a different set of algorithms to those used in supervised learning.

One unsupervised learning task is clustering, the main aim of which is to group similar data within the same cluster, other unsupervised learning task is a dimensionality reduction the main objective in this task is to reduce the set of attributes available for

training machine learning algorithms and/or to obtain a set of attributes that are more representative of the problem and we also have association rules.

To learn more about how unsupervised learning algorithms work, see the appendix A.1. In this appendix we highlight the main tasks in unsupervised learning and the main algorithms for each, such as K-MEANS and DBSCAN for clustering, among others.

So far, we have seen how the main machine learning algorithms work, especially algorithms for supervised learning tasks. However, algorithms are part of the process for building a model capable of making predictions. For this reason, the next chapter is dedicated to understanding the entire process of building a model, from addressing the problem to the phase of using the model to make predictions.

4 Machine Learning Life Cycle

In this chapter, we will explore the process of building a machine learning model, covering all the steps involved—from defining the problem to model deploying.

Building a machine learning model often goes beyond simply applying algorithms to a dataset. It requires a comprehensive understanding of the entire modeling process. Each stage plays a crucial role, and understanding the significance of each step is essential for successful implementation. Below are the key topics we will cover throughout this chapter.

- Trad-off Bias/Variance
- Setting Research Goal
- Data Understanding
- Data Exploration
- Data Preparation
- Data Modeling
- Deployment

4.1 Trad-off Bias-Variance

The Bias/Variance trade-off is not exactly a step in the model building process, but it is a fundamental concept that must be kept in mind throughout the process. When building a model, our goal is to capture knowledge from a specific domain, with the data serving as the key to uncovering this knowledge. Distinct algorithms represent different approaches to exploring the data space and deriving a model that can generalize to unseen data.

This process involves addressing two critical issues: overfitting and underfitting. As illustrated in Figure 11, overfitting occurs when the model performs exceptionally well on the training data but fails to generalize to new, unseen data. On the other hand, underfitting arises when the trained model fails to capture the underlying patterns in the data. This often happens when the algorithm used is constrained by limited degrees of freedom, making it unable to model the complexity of the data effectively.

Bias refers to errors caused by the model's overly simplistic assumptions or insufficient representation of the data. Variance, in contrast, captures the model's sensitivity to fluctuations in the training dataset.

A model with high bias tends to oversimplify the problem, failing to capture the complexity of the data and resulting in underfitting. In contrast, a high-variance model is excessively responsive to the training data, often leading to overfitting and poor performance on unseen data. Achieving a balance between bias and variance is essential for building a generalized model—one that performs well on new, unseen data. This balance

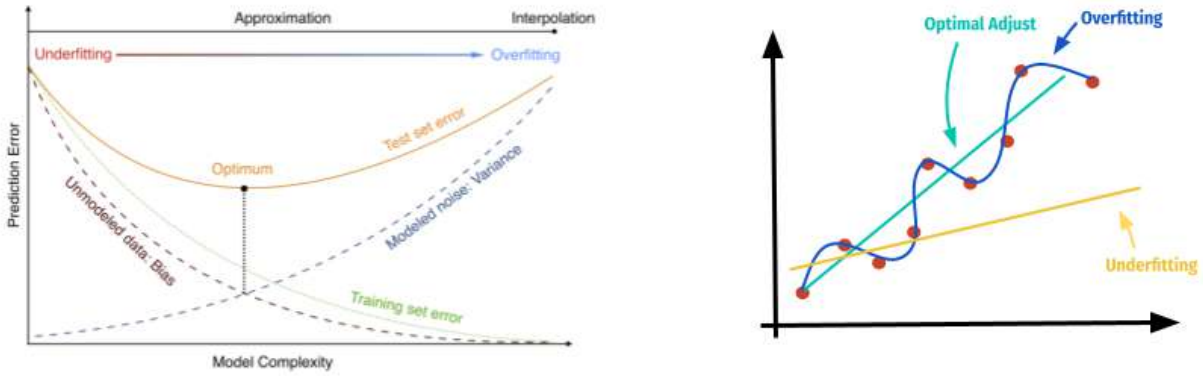

Figure 11 – Tradoff Vies/Variance.

can be accomplished through techniques such as hyperparameter tuning, cross-validation, and thoughtful feature selection.

Increasing a model's complexity enhances its capacity to explore the data space, effectively increasing its degrees of freedom. However, this heightened flexibility can lead to overfitting, as shown in Figure 11. On the other hand, reducing model complexity restricts its ability to capture patterns within the data, thereby increasing the likelihood of underfitting. The ideal model, therefore, achieves a balance—employing an algorithm that can accurately capture the data's underlying behavior while maintaining the ability to generalize effectively to new, unseen data.

4.2 Setting The Research Goal

The initial and most important step is defining the problem to be addressed. This step may require revisiting during the process to maintain alignment with objectives. It is crucial to identify the type of problem being tackled—classification, regression, clustering, association rule mining, or frequent pattern discovery. This involves specifying the goal or determining what needs to be uncovered from data.

In materials physics, asking the right questions is vital. For example, do we want to

distinguish good conductors from non-conductors? Predict a specific property? Verify if data for that property already exists? Investigate similarity relationships between materials? Questions framed during this stage guide the next phase: understanding data needed to solve the problem.

4.3 Data Understanding

In this phase, it is essential to understand the data we have, including its type: Are the variables qualitative or quantitative? If qualitative, are they nominal or categorical? If quantitative, are they discrete or continuous? Grasping these aspects is crucial for progressing to the next steps.

At this stage, assessing data quality is equally important. This involves checking for missing values and ensuring the variable types align with their respective attributes.

Another key task is examining how data is distributed across attributes. For instance, in a problem involving the classification of magnetic and non-magnetic materials, it's necessary to analyze the distribution of classes available for training the model. This understanding aids in selecting appropriate metrics to evaluate model performance and deciding whether techniques to balance the dataset are needed.

After completing this stage of data assessment, we move to the next phase: exploring the available data for algorithm training. However, before proceeding, it is critical to set aside a portion of the data that will remain unseen during training. This subset, referred to as validation data, plays a vital role later in the process, and its purpose will become clearer in subsequent steps.

4.4 Data Exploration

Data exploration is essential for understanding the problem and assessing whether available data is suitable for modeling. Formulating hypotheses about data is a key aspect of this phase. During this stage, univariate, bivariate, and multivariate analyses are conducted. Each type of statistical analysis provides insights to guide algorithm selection and ensure effective modeling.

In materials physics, this phase helps deepen understanding of available variables. For instance, when predicting bandgap values of materials, it can be used to generate and test hypotheses. One hypothesis could be: materials with low electronegativity have bandgaps above 3.2 eV. This hypothesis can then be validated or refuted based on data. This step prevents models from relying on spurious correlations, ensuring insights are meaningful and aligned with the problem.

4.5 Data Preparation

Data preparation precedes model, since models replicate patterns present in the data, inadequate preparation can lead to poor predictive accuracy.

This stage requires a thorough understanding of the available data and the machine learning algorithms to be used. The method of data preparation often depends on the chosen algorithm. For instance, the K-Nearest Neighbors (KNN) algorithm, which relies on distance metrics, is highly sensitive to the distances between points in the data space. If KNN is used, it becomes necessary to rescale numerical attributes to prevent the algorithm from disproportionately weighting attributes with larger ranges.

In addition to scaling, it is essential to encode categorical variables effectively to ensure meaningful input for the model. The next sections will explore specific techniques for preparing data to optimize model performance.

4.5.1 Feature Rescaling

There are various methods for rescaling data, each suited to the specific distribution of the dataset. Below, we present some of the most commonly used methods.

4.5.1.1 Normalization

If the feature under analysis follows a normal distribution, we recommend normalizing the variable. Normalization adjusts the variable to have a mean of zero and a standard deviation of one. The process for normalization is shown in Equation 4.1.

$$x_{norm} = \frac{x - \mu}{\sigma}, \tag{4.1}$$

in the equation 4.1 x represents a data point from the actual distribution, μ denotes the mean, and σ is the standard deviation.

4.5.1.2 Min-Max Scaler

Another method for rescaling data is the min-max scaler, as described in Equation 4.2. This approach is used when a variable with a mean of zero and a standard deviation of one is not suitable. Instead, this method scales all values to lie within the range [0,1].

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{4.2}$$

in the equation 4.2 x is the point of real distribution, x_{min} and $_{max}$ is the minimum and maximum observation in the data respectively.

4.5.1.3 Robust Scaler

This rescaling method leverages the quartiles of the data, providing a robust approach when outliers are present. Unlike methods that rely on the entire range of the dataset, such as min-max scaling, quartile-based rescaling is less sensitive to extreme values. It focuses on the interquartile range (IQR), which represents the spread of the middle 50% of the data, effectively reducing the impact of outliers.

By normalizing the data based on its quartiles, this method ensures that the central portion of the data is well-represented, while extreme values do not disproportionately influence the scaling process. This makes it particularly suitable for datasets with skewed distributions or a significant number of outliers, ensuring better model performance and reliability in subsequent steps.

$$x_{norm} = \frac{x - Q_1}{Q_3 - Q_1} \tag{4.3}$$

in the equation 4.3 x is the point of real distribution, Q_1 and Q_3 is the first and third quartile.

4.5.2 Feature Encoding

An important step in the data preparation process is the feature encoding of categorical variables. Many machine learning algorithms cannot process string-based attributes, making it necessary to convert these attributes into numerical formats. However, care must be taken to avoid potential issues arising from these transformations. One common problem is the introduction of spurious weighting, where numerical encoding inadvertently assigns significance to a variable that it does not inherently possess. To mitigate such issues, and depending on the nature of the categorical variable, various feature encoding methods can be employed.

Feature encoding methods can be broadly categorized into classical techniques, such as one-hot encoding, ordinal encoding, binary encoding, label encoding, frequency encoding, and count encoding. Each method has specific use cases and potential pitfalls, making the choice of encoding critical to maintaining the integrity of the data.

In this section, we will focus on two key encoding methods applied during the modeling process, which will be further detailed in the results section. To illustrate these methods, we will use the data presented in Table 1. The "type" variable in the table will serve as an example for encoding the materials.

4.5.2.1 One-hot Encoding

The one-hot encoding method transforms the observations of a given variable into new binary variables, each taking on values of 0 or 1. For instance, a non-ordinal variable

Table 1 – Example dataset of materials categorized by type (metallic or insulator).

Material	**Type**
Iron	Metallic
Aluminum	Metallic
Copper	Metallic
Glass	Insulator
Wood	Insulator
Rubber	Insulator
Gold	Metallic
Silver	Metallic
Plastic	Insulator

like the "type" variable in Table 1 can be encoded using one-hot encoding. To apply this method, the "type" variable is split into two new variables: "metallic" and "insulator." Observations classified as metallic will have a value of 1 in the "metallic" column and 0 in the "insulator" column, while insulating materials will have 1 in the "insulator" column and 0 in the "metallic" column.

Table 2 – Dataset transformed using one-hot encoding for the variable "Type".

Material	**Metallic**	**Insulator**
Iron	1	0
Aluminum	1	0
Copper	1	0
Glass	0	1
Wood	0	1
Rubber	0	1
Gold	1	0
Silver	1	0
Plastic	0	1

4.5.2.2 Target Enconding

Target encoding, also known as mean encoding, is commonly used in binary classification problems. In this method, the categorical variable is replaced with the average value of the target variable corresponding to each category. For instance, using the dataset in Table 1, target encoding can be applied to transform the categorical variable, and the resulting encoded table is shown in Table 3.

4.5.2.3 Label Enconding

The next encoding method is label encoding, which involves replacing the categorical variable with numeric values. In simple terms, this method assigns a unique numerical code to each category. For example, applying label encoding to the variable in Table 1 results in the transformed table shown in Table 4.

Table 3 – Dataset transformed using Target Encoding for the variable "Type".

Material	**Type (Encoded)**
Iron	0.75
Aluminum	0.75
Copper	0.75
Glass	0.25
Wood	0.25
Rubber	0.25
Gold	0.75
Silver	0.75
Plastic	0.25

Table 4 – Dataset with Label Encoding applied to the variable "Type".

Material	**Type (Encoded)**
Iron	1
Aluminum	1
Copper	1
Glass	2
Wood	2
Rubber	2
Gold	1
Silver	1
Plastic	2

In feature encoding, one must be carefully to address potential issues such as overfitting and spurious variable weighting, as discussed earlier in this section. Different encoding methods offer distinct advantages and challenges. For example, one-hot encoding helps eliminate spurious weightings but can significantly increase dimensionality. In contrast, target encoding avoids dimensionality issues but can lead to overfitting. Label encoding, while mitigating some of these concerns, may still introduce spurious weightings by assigning numerical values to variables that lack inherent weights.

Although there are many other feature encoding techniques, the fundamental goal remains the same: to convert categorical variables into numerical representations that can be processed effectively by machine learning algorithms.

4.5.3 Data Balancing Techniques

Another way of preparing the data is by balacing the classes. Data balancing is particularly interesting to use in classification problems. For example, if we are trying to predict magnetic and non-magnetic materials, if the amount of magnetic materials data available for training the model is much greater than the amount of non-magnetic materials available for training the model, we could run into the problem of overfitting, i.e. the model built will be much more accurate with the data from the majority class.

To avoid this problem, it is very interesting to use some class balancing methods. These methods are primarily divided into oversampling and undersampling methods, but there are cases where both techniques are combined.

4.5.3.1 (Oversampling and UnderSampling - SMOTE | ADASYN)

Oversampling techniques are used when we have class imbalance and seek to balance the number of observations from the minority class. This process can be done randomly generating new data in the data space, however, this way can construct observations outside of reality. Therefore, other methods emerge, such as the Synthetic Minority Oversampling Technique (SMOTE)[37] and Adaptive Synthetic(ADASYN) [38], which carry out this process using methods that consider the data space.

$$x_{new} = x_{minority} + rand(0,1) * (x_k - x_{minority}), \tag{4.4}$$

where:

$x_{minority}$ is a minority class instance. x_k is one of the k-nearest neighbors of $x_{minority}$ and $rand(0,1)$ is a random number between 0 and 1.

The SMOTE class balancing method is not the most appropriate of the methods, as the probability of creating sampling bias is very high. There are other methods such as ADASYN that seek to improve the SMOTE method, there are also strategies that combine both forms of resampling such as Smotetomek which oversamples and then undersamples.

4.6 Data Modeling

In this phase, we use the most diverse algorithms to build the model, test several algorithms, from methods based on distance to methods based on optimization, so that we can evaluate the constructed model it is necessary to use different metrics.

4.6.1 Feature Selection

Feature selection is a crucial step in the process of building machine learning models. It involves choosing the most relevant and informative features from the available data to improve model performance and reduce computational complexity. Feature selection is essential for several reasons:

- **Improved model performance**: Removing irrelevant or redundant features can enhance accuracy and generalization.
- **Reduced overfitting**: Eliminating unnecessary variables lowers the risk of the model fitting noise rather than capturing underlying patterns.

- **Faster training and inference**: Working with a smaller feature set significantly speeds up the learning and prediction processes.

.

There are several techniques for feature selection, including:

4.6.1.1 Filter Methods

These methods assess feature relevance independently of the machine learning algorithm. Common techniques include correlation analysis and statistical tests. At this stage, the correlation calculation between variables can be used to eliminate predictor variables that are highly correlated with each other. This should occur to reduce computational processing time, since two highly correlated predictor variables have the same weight for prediction.

4.6.1.2 Wrapper Methods

Wrapper methods use the machine learning algorithm's performance as a criterion for feature selection. Examples include recursive feature elimination and forward selection. Wrapper methods are performed by training a specific algorithm on a subset of attributes, using a metric as a target (accuracy for example), training the model with this subset and calculating the metric, then selecting a subset again and repeating the process, at the end the set of attributes that maximizes the chosen metric is selected.

4.6.1.3 Embedded Methods

These methods incorporate feature selection as part of the model training process. For instance, L1 regularization in linear models encourages sparsity and automatically selects relevant features, another example is when we train a decision tree and at the end we check the importance of the attributes for the predictions, then these attributes are used as the most relevant.

4.6.1.4 Dimensionality Reduction

In some cases, dimensionality reduction techniques like Principal Component Analysis (PCA) or t-SNE may be employed to reduce the number of features while preserving as much information as possible.

In summary, feature selection is a critical step in the machine learning pipeline. It requires a combination of data analysis, domain knowledge, and experimentation to identify and retain the most relevant features, ultimately leading to improved model performance and interpretability.

4.6.2 Model Evaluation Techniques

We have previously discussed the importance of data in building models, therefore, model evaluation techniques are a fundamental step in knowing whether the model created can actually represent the problem we are trying to model and whether this model can generalize to new data.

4.6.2.1 Hold-Out

The hold-out technique involves splitting the data into two distinct sets: a training set and a test set. The model is trained on the training set and evaluated on the test set. The test set data is not used during training, which helps assess how well the model generalizes to unseen data. The main advantages it's simple to implement and quick to execute. It is useful when dealing with a large dataset and the disadvantages is the random choice of the split can lead to different results in each run. Additionally, the effectiveness of the evaluation depends on the representativeness of the test set.

4.6.2.2 Cross Validation

Cross-validation is a technique that involves dividing the data into multiple partitions or folds and performing multiple rounds of training and testing. The model is trained on various combinations of training and test partitions, and performance metrics are averaged across all rounds. K-Fold Cross-Validation divided into k partitions, where k is an integer. The figure 12 it can be seen that model is trained k times, using a different partition as the test set in each iteration, that is, the model is trained and tested on all partitions of the data. The main advantages in cross-validation is provides a more robust assessment of model performance, as it uses all data for both training and testing, reducing evaluation variability and your disadvantages is it can be computationally expensive, especially with large datasets.

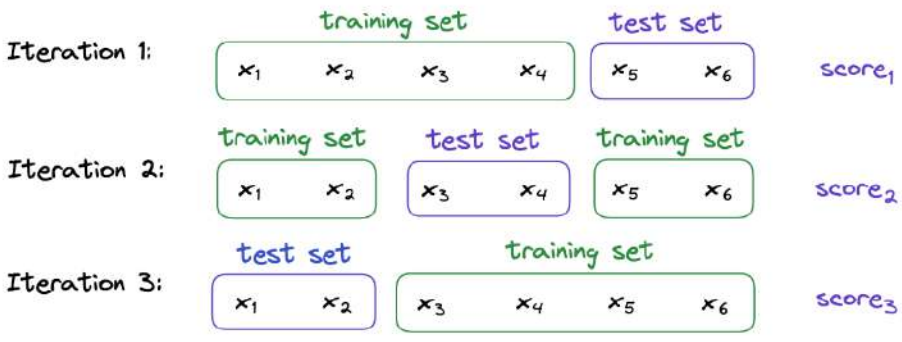

Figure 12 – Cross validation K-Fold representation.

4.6.2.3 Leave One Out

Leave-one-out is a variation of cross-validation where each data point is used as the test set once, with all other data points forming the training set in each iteration. It represents an extreme form of cross-validation, where the number of partitions equals the number of data points. The advantages is LOOCV provides an unbiased estimate of performance since each data point is individually tested and the disadvantages it can be extremely computationally intensive, particularly with large datasets. It can also be sensitive to outliers and data noise. For this reason, LOOCV is best used only on small datasets.

Figure 13 – Cross Validation Leave One Out.

In summary, the choice of a model evaluation technique depends on factors like dataset size, computational availability, and the need for a robust performance estimate. Hold-out is simple and quick but may be sensitive to the choice of split. Cross-validation, including leave-one-out, provides more reliable estimates but can be computationally intensive. Each technique has its place, and the choice should be based on the specific characteristics of the problem and data at hand.

4.6.3 Evaluation Metrics

Model evaluation metrics are a fundamental part of building efficient models capable of reproducing the reality of the data. Each task requires the use of different metrics to evaluate the model. For each type of machine learning task, we have different metrics to evaluate the algorithms.

In classification tasks, you need to keep the following definitions in mind:

1. **True Positive (TP):** The actual and predicted value are the positives.
2. **True Negative (TN):** The actual and predicted value are the negatives.
3. **False Positive (FP):** The actual value is negative and the model predict positive.
4. **False Negative (FN):** The actual value is positive and the model predict negative.

The first metrics we are using in classification task is accuracy can be observed in the equation 4.5, that metric measures the proportion of correctly classified instances out of the total instances in the dataset. It's a general metric suitable for balanced datasets when all classes have roughly equal representation.

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{4.5}$$

Precision quantifies the ability of the model to correctly identify positive instances among those it predicted as positive. It emphasizes the absence of false positives. Can be observed in the equation 4.6:

$$precision = \frac{TP}{TP + FP} \tag{4.6}$$

Recall measures the model's ability to correctly identify all positive instances. It emphasizes the absence of false negatives. can be observed in the equation 4.7:

$$recall = \frac{TP}{TP + FN} \tag{4.7}$$

Specificity measures the model's ability to correctly identify all negative instances. It emphasizes the absence of false positives. Can be observed in the equation 4.8:

$$specificity = \frac{TN}{TN + FP} \tag{4.8}$$

The F1-Score is the harmonic mean of precision and recall. It balances precision and recall, making it useful when both false positives and false negatives need to be minimized. Can be observed in the equation 4.9:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{4.9}$$

In regression tasks, you need to keep the following metrics:

Mean Absolute Error (MAE) calculates the average absolute differences between predicted values and actual target values. In the equation 4.10 it provides a measure of the model's accuracy in terms of absolute errors.

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{4.10}$$

Mean Squared Error(MSE), in equation 4.11, calculates the average of the squared differences between predicted values and actual target values. It emphasizes larger errors more than MAE. MSE penalizes larger errors and is commonly used for optimization algorithms.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4.11}$$

Root Mean Squared Error(RMSE) is the square root of MSE, in equation 4.12 which provides a measure of the model's accuracy in the same units as the target variable.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{4.12}$$

In equation 4.13 we can see R^2 quantifies the proportion of the variance in the target variable that is explained by the model. It ranges from 0 to 1, with higher values indicating a better fit. It provides a measure of the goodness of fit of the model to the data.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{4.13}$$

Viewing the metrics is a good choice to understand which models are good, but in cases of classification models, we can view the performance of the model through the Receiver operating characteristic (ROC) curve and the precision recall curve.

The ROC curve quantifies the proportion of true positives and false positives, in the figure 14 that the best curve is the one that approaches the axis of true positives, as this way we guarantee that we have a classifier with greater accuracy. For this reason, we use the area under curve (AUC) to compare different classifiers.

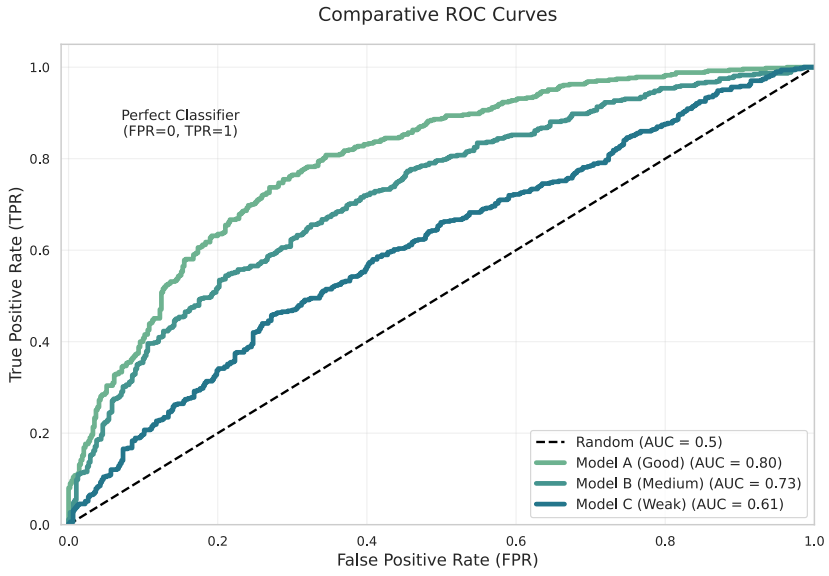

Figure 14 – Receiver Operating Characteristic Curve (ROC Curve).

The roc curve and the area under the curve are not always able to accurately measure a classifier, especially in unbalanced data, the area under the roc curve can provide a wrong metric, for this reason, we use the precision-recall curve which quantifie the proportion between Precision and Recall (also known as Sensitivity or True Positive Rate).

The Precision-Recall, Figure 28 , curve is especially useful for imbalanced data sets because it focuses on the performance of the minority class, ignoring the performance of the majority class which may dominate the ROC curve metrics.

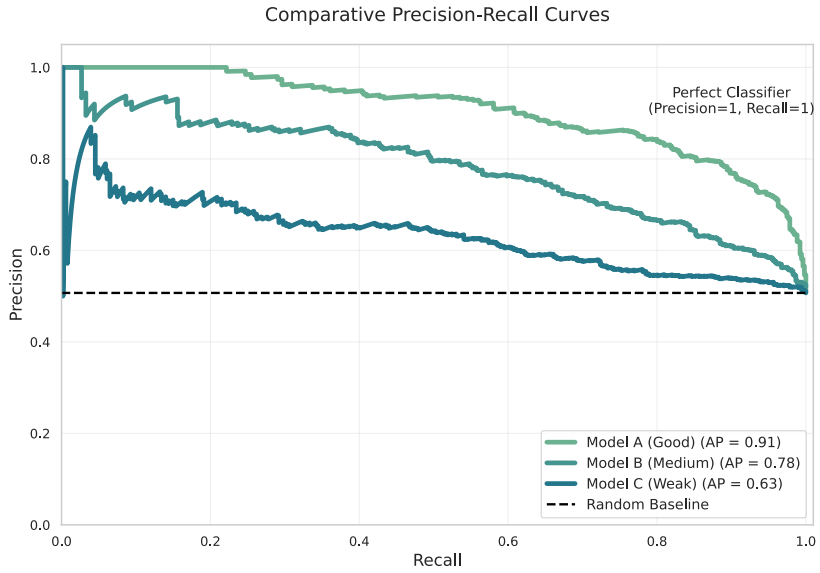

Figure 15 – Precision-Recall Curve.

Clustering tasks, association rules and frequent pattern mining have other different evaluation metrics, for more explation about this consult the appendice A.1.

4.6.4 Hyperparameter Fine Tuning

Hyperparameters are externally configurable parameters that influence the training process of a machine learning model, but that are not learned by the model during training. Unlike model parameters, which are automatically adjusted by the learning algorithm to minimize a specific loss function, hyperparameters are set before training and generally do not change during the training process. Some examples of hyperparameters are: learning rate, number of trees, maximum tree depth, number of estimators, among others that directly depend on the algorithm being used.

There are some ways to find the best hyperparameters for the model, let's delve deeper into each of them: grid search, random search and Bayesian optimization.

4.6.4.1 Grid Search

Grid Search is a systematic method for exploring the hyperparameter space, where a grid is defined with all possible combinations of values for the hyperparameters of interest. These combinations are then evaluated using cross-validation or a separate validation dataset.

The main benefit of Grid Search is its comprehensive and systematic approach, ensuring that all combinations are explored. However, Grid Search can become computationally expensive in large hyperparameter spaces or when there are many options for each hyperparameter.

4.6.4.2 Random Search

Random Search is an alternative to Grid Search that selects random samples from the hyperparameter search space instead of evaluating all possible combinations. This approach enables more efficient exploration of the hyperparameter space, particularly in high-dimensional spaces where Grid Search becomes computationally expensive and impractical. While Random Search does not guarantee exhaustive coverage of the search space, studies have shown that it often outperforms Grid Search in efficiency and effectiveness, particularly when dealing with a large number of hyperparameters[(?, 39)].

4.6.4.3 Bayesian Optimization

Bayesian search is a more sophisticated approach that models the relationship between hyperparameters and performance metrics using probabilistic modeling techniques[(39)].

It uses a probabilistic model to predict which hyperparameter configurations are most promising based on previous observations. As a result, Bayesian search is able to direct the search to the most promising regions of the hyperparameter space, reducing the total number of evaluations required.

Although Bayesian search may require more complex initial setup, it is generally more efficient than Grid Search and Random Search, especially in high-dimensional hyperparameter spaces.

4.7 Deployment

In this fase, we use the model to make predictions for new data that is unknown, that is, not part of the dataset used to train the model.

There are many ways to deploy a model, such as: One-Off Deployment, batch deployment, real-time deployment, streaming deployment, edge deployment. The simplest of these is to use the functions of the packages used to build the model, which we call one-fff Deployment. However, this may be a way that only the person who built the model will

have access to. So there are other model deployment strategies that can be seen in more detail in the appendix A.4.

5 Results and Discussions

In this chapter, we have two main objectives, the first is related to the construction of the models, we will show how it was built and the evolution of the metrics until the choice of the model used to make the predictions. The second objective is related to the model's predictions. We will verify its predictions by selecting some materials to perform the DFT calculation and the orbital hall conductivity property using PAOFLOW.

5.1 Model Building

We will dedicate ourselves to presenting each step of the model construction, as previously discussed, we will build a classification model to separate metallic and insulating materials, then we will build the regression model to predict the orbital hall conductivity. The first step in building the model is the data source we will use and this will be the topic of discussion in the next section.

5.1.1 Feature Engineering - Building the Dataset

To build the classification model we use the c2db database, this database organises a variety of structural, thermodynamic, elastic, electronic, magnetic, and optical properties of around 1500 two-dimensional materials distributed over more than 30 different crystal structures. [14] From this database, we searched for all materials in different steichiometries and all with their bandgap values, with these values we were able to build a database with metallic and insulating materials. In the Table 5 shows the base attributes for building the model.

However, to train the model, we use these features as a foundation while constructing what we refer to as statistical attributes. For each property, we compute the mean, weighted mean, maximum value, minimum value, standard deviation, and weighted standard deviation. The calculations for these attributes can be found in Table 6. In Table 7 we show the construction of the statistical features for the atomic number for a sample of our data with different materials and prototypes.

This dataset construction approach is also used to construct the dataset for the regression model, the difference being that we only use the orbital hall conductivity data calculated in PAOFLOW. We will explore in more detail the prototypes and materials used in the regression model results section.

The selection of features in this work was guided by both physical intuition and prior literature on materials informatics. Specifically, elemental properties such as electronegativity, atomic radius, ionization energy, and valence electron count were used as the basis

Table 5 – Features used in training machine learning models

Property	**Description**
Z	Atomic number
Electronegativity	Tendency to attract electrons
Ionization Potential	Energy to remove an electron
Electron Affinity	Energy change when gaining an electron
HOMO	Highest occupied molecular orbital
LUMO	Lowest unoccupied molecular orbital
r_s	Radius of the s orbital
r_p	Radius of the p orbital
r_d	Radius of the d orbital
r_{atomic}	Atomic radius (non-bonded)
r_{valence}	Valence orbital radius
r_{covalent}	Covalent radius
Valence	Number of valence electrons
PeriodicColumn	Periodic table column
PeriodicColumn_upto18	Adjusted column (1–18)
UnfilledOrbitals	Unoccupied orbitals
Polarizability	Electron cloud distortion

Table 6 – Summary of statistical metrics for γ values

Symbol	**Description**	**Formula**
γ	Arithmetic mean	$\gamma = \frac{\sum_{i=1}^{n_s} \gamma_i}{n_s}$
$\tilde{\gamma}$	Weighted mean by the number of each atom type	$\tilde{\gamma} = \frac{\sum_{i=1}^{n_s} \gamma_i n_i}{N}$
γ_M	Maximum value	$\gamma_M = \text{Max}(\gamma_i)$
γ_m	Minimum value	$\gamma_m = \text{Min}(\gamma_i)$
γ_σ	Standard deviation relative to the mean	$\gamma_\sigma = \sqrt{\frac{\sum_{i=1}^{n_s} (\gamma - \gamma_i)^2}{n_s}}$
$\tilde{\gamma}_\sigma$	Standard deviation relative to the weighted mean	$\tilde{\gamma}_\sigma = \sqrt{\frac{\sum_{i=1}^{n_s} (\tilde{\gamma} - \gamma_i)^2}{n_s}}$

for generating features. These quantities are known to influence fundamental properties in solid-state physics, including band gap, charge carrier mobility, and orbital interactions.

For instance, differences in electronegativity between constituent atoms affect charge transfer and bonding character, which in turn relate to band dispersion and potential topological effects. Atomic radii provide insight into lattice packing and orbital overlap, while ionization energies reflect the ease of electron excitation—directly relevant to optical and electronic transitions.

Material	Prototype	Media_Z	Media_pon_Z	Max_Z	Min_Z	Desvio_Z	Desvio_pon_Z
ScF_2	MoS_2	15	13	21	9	6	6.3246
ScF_2	CdI_2	15	13	21	9	6	6.3246
ScF_2	WTe_2	15	13	21	9	6	6.3246
ScF_2	GeS_2	15	13	21	9	6	6.3246
$ScCl_2$	MoS_2	19	18.3333	21	17	2	2.1082
$ScCl_2$	CdI_2	19	18.3333	21	17	2	2.1082
$ScCl_2$	WTe_2	19	18.3333	21	17	2	2.1082
$ScCl_2$	GeS_2	19	18.3333	21	17	2	2.1082
$ScBr_2$	MoS_2	28	30.3333	35	21	7	7.3786
$ScBr_2$	CdI_2	28	30.3333	35	21	7	7.3786

Table 7 – Atomic number (Z) statistical features of different materials

To represent multi-element compounds, statistical descriptors such as mean, maximum, minimum, and standard deviation were computed over the set of atomic properties. These statistical aggregations provide a way to capture both central tendencies and variations in the compositional space. For example, a large standard deviation in electronegativity may indicate strong polar bonding, whereas a small mean atomic radius might correlate with tight packing and higher orbital overlap.

This approach is rooted in the compositional model paradigm, where properties of materials are predicted using only the identities and ratios of constituent elements. While this abstraction simplifies the modeling process and enables broader coverage of chemical space, it does come with limitations. Notably, explicit structural information such as atomic positions, bonding geometries, and crystal symmetries are not directly encoded in these features. As a result, certain emergent properties that are highly structure-sensitive—such as orbital Hall conductivity (OHC)—may be only partially captured by this feature set.

Nonetheless, compositional features are often sufficient to distinguish broad classes of materials (e.g., metals vs. insulators, topological vs. trivial phases) and offer a computationally inexpensive pathway for initial screening. This justifies their use in the early-stage modeling efforts pursued in this study.

5.1.2 Classification Model

In this section, we will dedicate ourselves to explaining how we built the model to distinguish between materials that are metals and materials that are insulators.

5.1.2.1 Classification Model - Simple Model

The model that we call simple models is the model built without much preparation on the data, that is, we just used the data and treated the missing values as well as

encoding the prototype using label encoding, disregarding the distribution of the data, the correlation between features, class balancing and the importance of features. We only build the model by exploring the data with the algorithms and using cross-validation of the data to obtain the model evaluation metrics.

Table 8 – Results Simple Classification Model

Model	auprc	accuracy	precision	recall	f1_score
XGBoosting	0.920	0.895	0.878	0.804	0.838
CatBoost	0.917	0.891	0.890	0.776	0.828
Decision Tree	0.718	0.869	0.812	0.799	0.805
Gradient Boosting	0.885	0.860	0.857	0.709	0.774
AdaBoostClassifier	0.840	0.831	0.812	0.654	0.723
Random Forest	0.824	0.810	0.749	0.666	0.703
Extra Trees	0.763	0.806	0.735	0.675	0.703
Logistc Regression	0.653	0.742	0.641	0.553	0.592
GaussianNB	0.579	0.690	0.542	0.570	0.555
KNN	0.564	0.711	0.591	0.479	0.528

Note in the table 8 that the KNN, distance-based model had the worst performance, this is strongly related to the data preparation process, that is, being highly dependent on the data distribution, this algorithm suffers from spurious powers which leads to errors of prediction, which justifies the low metrics. In contrast, tree-based models and ensemble models had the best performances.

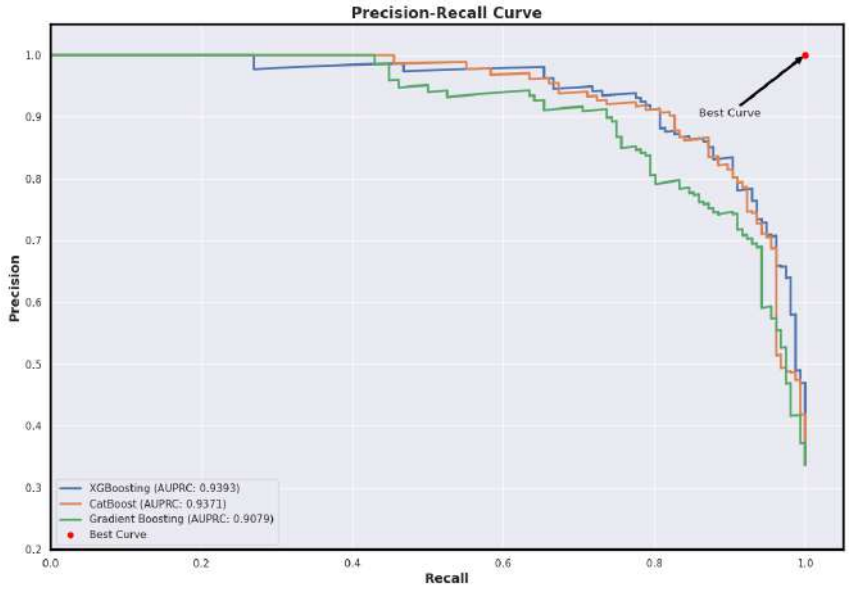

Figure 16 – Result of the precision-recall curve for the simple model. The arrow on the graph points to the point where the best model would be, that is, the closer to the point the better.

As we disregarded class imbalance in the data, it was important to plot the precision and recall curve, see in figure 16, this curve helps to verify the best model as it takes into

account both precision and recall, that is, it considers false positives when calculating the precision, as well as false negatives in the case of recall, not allowing us to visualize a metric that does not show the reality of the model's performance.

5.1.2.2 Classification Model - Model With Data Preparation

In building the classification model using data preparation, in addition to the preparations made for the previous model, we used the encoding of the prototype categorical feature, in addition to normalizing the data by performing feature rescaling.

Table 9 – Model Result With Data Preparation

Model	auprc	accuracy	precision	recall	f1_score
CatBoost	0.933	0.903	0.899	0.807	0.851
XGBoosting	0.926	0.898	0.875	0.818	0.844
Decision Tree	0.752	0.886	0.840	0.825	0.831
Gradient Boosting	0.904	0.873	0.889	0.718	0.792
Extra Trees	0.877	0.847	0.801	0.734	0.765
AdaBoostClassifier	0.843	0.848	0.832	0.694	0.756
Random Forest	0.839	0.822	0.774	0.678	0.722
Logistc Regression	0.760	0.786	0.715	0.616	0.660
GaussianNB	0.674	0.745	0.630	0.632	0.621
KNN	0.622	0.731	0.634	0.503	0.558

Note that in table 9 we had a significant improvement in the KNN distance-based model and an improvement in the f1 score for all models.

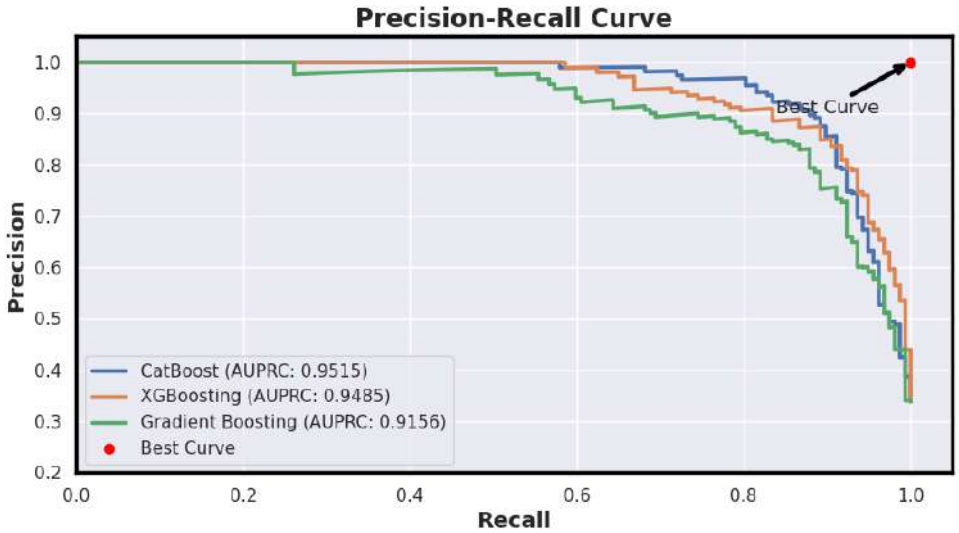

Figure 17 – Precision-Recall Curve Result With Data Preparation. The arrow on the graph points to the point where the best model would be, that is, the closer to the point the better.

5.1.2.3 Classification Model - Model With Feature Selection

In table 10, we can see the results for the model with attribute selection, previously the attribute selection we had 104 attributes, that is, all results so far were produced using the 104 attributes. The results in the following table were produced with only 22 of the 104 attributes, that is, we were able to reduce the number of attributes and maintain the model metrics.

Table 10 – Model Results With Feature Selection

Model	auprc	accuracy	precision	recall	f1_score
CatBoost	0.933	0.904	0.899	0.807	0.851
XGBoosting	0.926	0.898	0.875	0.818	0.844
Decision Tree	0.733	0.876	0.822	0.817	0.818
Gradient Boosting	0.902	0.873	0.889	0.717	0.791
Extra Trees	0.878	0.849	0.808	0.734	0.768
AdaBoostClassifier	0.843	0.848	0.832	0.694	0.756
Random Forest	0.838	0.827	0.784	0.682	0.728
Logistc Regression	0.760	0.786	0.715	0.616	0.660
GaussianNB	0.674	0.745	0.630	0.632	0.621
KNN	0.621	0.731	0.634	0.503	0.558

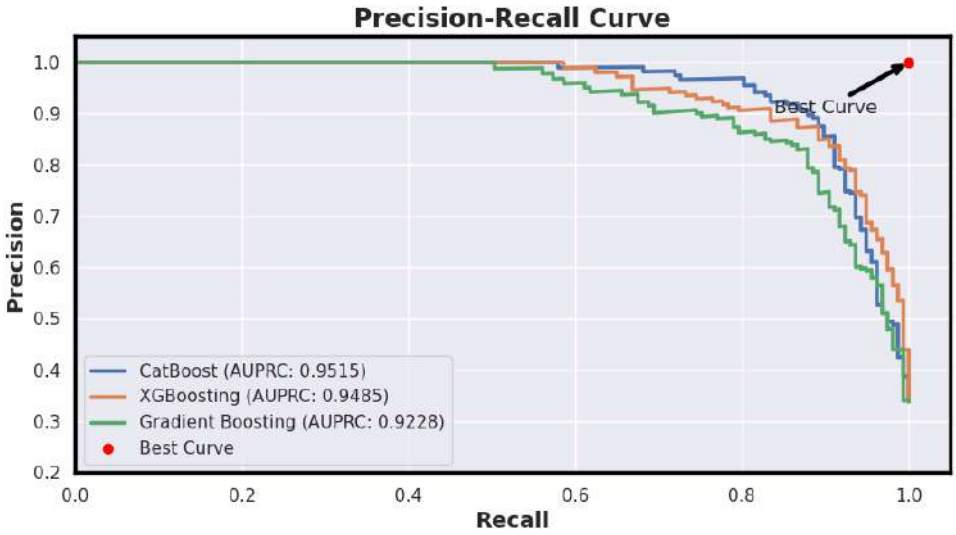

Figure 18 – Precision-Recall Curve Result with Feature Selection.

5.1.2.4 Classification Model - Model With Data Balacing Technics

In this section, we will discuss the results of the model using data balancing techniques. We use random under sampling techniques and the ADASYN algorithm to perform oversampling.

5.1.2.5 Classification Model - Random Undersampling

Using random undersampling, we removed random observations from the data and trained the algorithms with balanced data. Note that we had a gain in the f1 score metrics for most models, although not significant, it shows that with less data we were able to maintain the model's performance .

Table 11 – Model Results With Random Undersampling (RUS)

Model	**auprc**	**accuracy**	**precision**	**recall**	**f1_score**
CatBoost	0.939	0.871	0.869	0.876	0.872
Decision Tree	0.803	0.857	0.848	0.872	0.859
Gradient Boosting	0.928	0.853	0.859	0.849	0.853
XGBoosting	0.933	0.849	0.843	0.860	0.850
AdaBoostClassifier	0.901	0.810	0.821	0.798	0.808
Extra Trees	0.894	0.798	0.803	0.794	0.797
Random Forest	0.872	0.782	0.788	0.775	0.780
Logistc Regression	0.831	0.762	0.771	0.751	0.759
KNN	0.734	0.707	0.723	0.672	0.696
GaussianNB	0.786	0.637	0.850	0.339	0.480

Although the training data apparently shows a performance gain, we can see something in the precision recall curve in Figure 19 for validation data, in data that the model is unaware of, it performs worse, indicating a loss of accuracy that could be evidence of model overfitting.

5.1.2.6 Classification Model - Oversampling with ADASYN

In this section, we will check how the model testing and validation metrics behaved when using the ADASYN algorithm to resample the minority class.

Note in the table 12 that we had a significant gain in the aurpc and f1 score metrics, however, in this type of data balancing it is important to validate that we are not creating selection biases in the data, since we are creating new data from data that already exists. they exist.

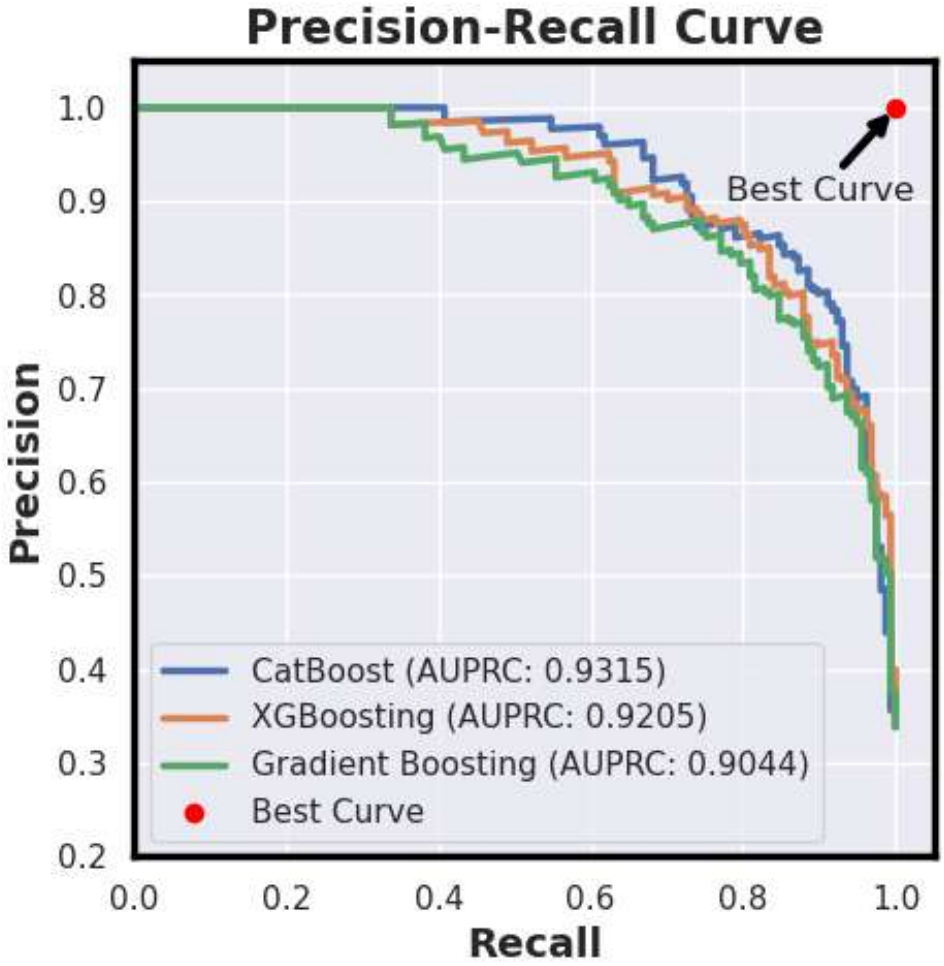

Figure 19 – Precision-Recall Curve Result Data Balancing Using RUS.

Table 12 – Model Results with using ADASYN

Model	auprc	accuracy	precision	recall	f1_score
CatBoost	0.978	0.931	0.919	0.946	0.932
XGBoosting	0.980	0.925	0.919	0.932	0.925
Gradient Boosting	0.948	0.902	0.889	0.919	0.903
Extra Trees	0.960	0.896	0.892	0.903	0.896
Decision Tree	0.850	0.890	0.897	0.881	0.889
Random Forest	0.935	0.869	0.868	0.873	0.870
AdaBoostClassifier	0.913	0.849	0.844	0.855	0.849
Logistic Regression	0.899	0.806	0.815	0.790	0.802
KNN	0.769	0.744	0.691	0.882	0.775
GaussianNB	0.754	0.674	0.697	0.651	0.652

Although the metrics presented are good, note that we had worse performance on the validation data, indicating some overfitting in the model training.

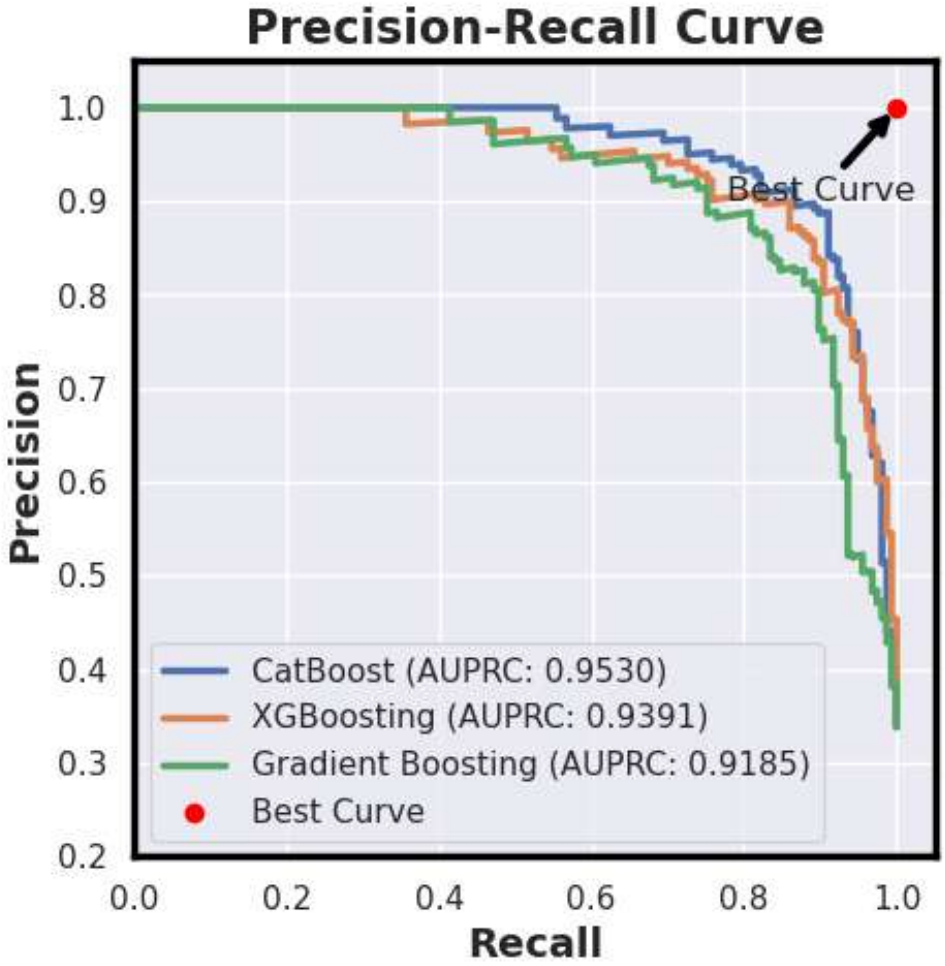

Figure 20 – Precision-Recall Curve Results with ADASYN.

5.1.2.7 Classification Model - Training the Best Model using Feature Selection and Data Preparation

In this section, we will present the results for the model that we will use to make predictions, here we will use two steps that we did individually, the preparation of the data, that is, removing any purifying power while maintaining the variability and the selection of features that we saw that eliminating some features we do not lose predictive accuracy.

Note that in table 13 we had excellent aurpc and f1-score results, validating the predictive accuracy of our model.

Table 13 – Model Results With Feature Selection and Data Preparation

Model	auprc	accuracy	precision	recall	f1_score
XGBoosting	0.945	0.916	0.896	0.852	0.873
CatBoost	0.942	0.908	0.896	0.823	0.858
Decision Tree	0.756	0.889	0.844	0.827	0.834
Random Forest	0.923	0.888	0.859	0.803	0.829
Gradient Boosting	0.918	0.883	0.878	0.760	0.814
Extra Trees	0.895	0.865	0.806	0.793	0.798
AdaBoostClassifier	0.867	0.847	0.829	0.696	0.754
KNN	0.832	0.836	0.769	0.739	0.752
Logistic Regression	0.742	0.778	0.722	0.562	0.630
GaussianNB	0.671	0.727	0.591	0.640	0.613

By visualizing the aurpc curve in figure 21, we can see that the model gained in aurpc performance and f1 score in the training metrics and when we evaluate the validation data we see an improvement in the model's behavior, indicating a gain in predictive accuracy.

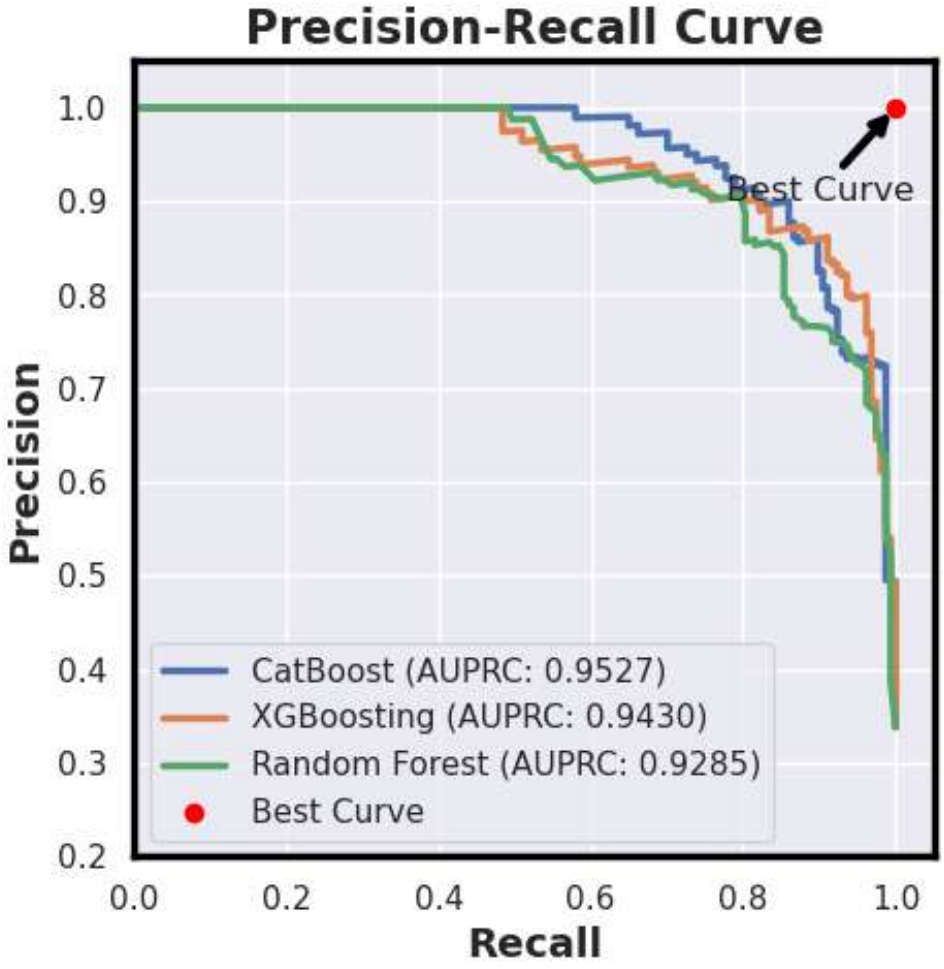

Figure 21 – Precision-Recall Curve With Feature Selection and Data Preparation.

5.1.3 Regression Model

The regression model was built to make predictions of the orbital hall conductivity.

5.1.3.1 Regression Model - Data Description

Before starting to build the model, we made a quick description of the data we had for development. In figure 22 we can see that we have materials with different stoichiometries and different numbers of atoms.

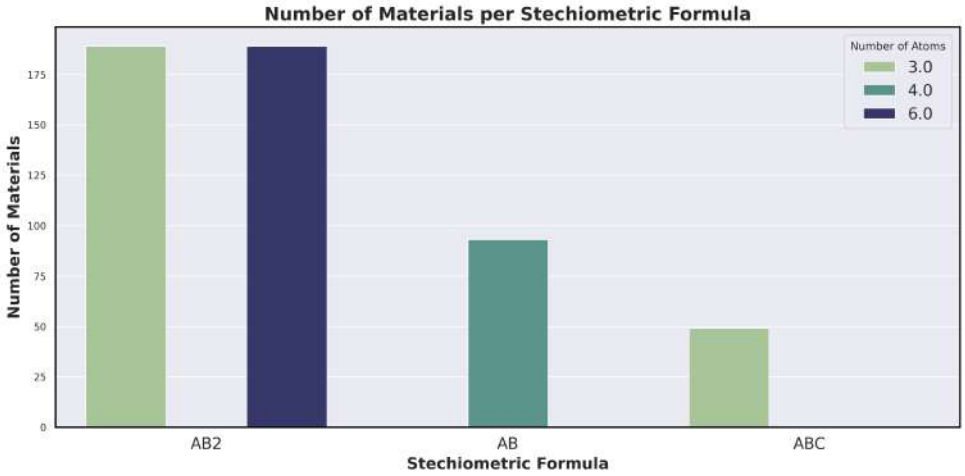

Figure 22 – Number of materials by Stoichiometry.

In figure 23 you can see a larger volume of the MoS2 prototype and a smaller volume of PbS, however all of them will be used to build the model.

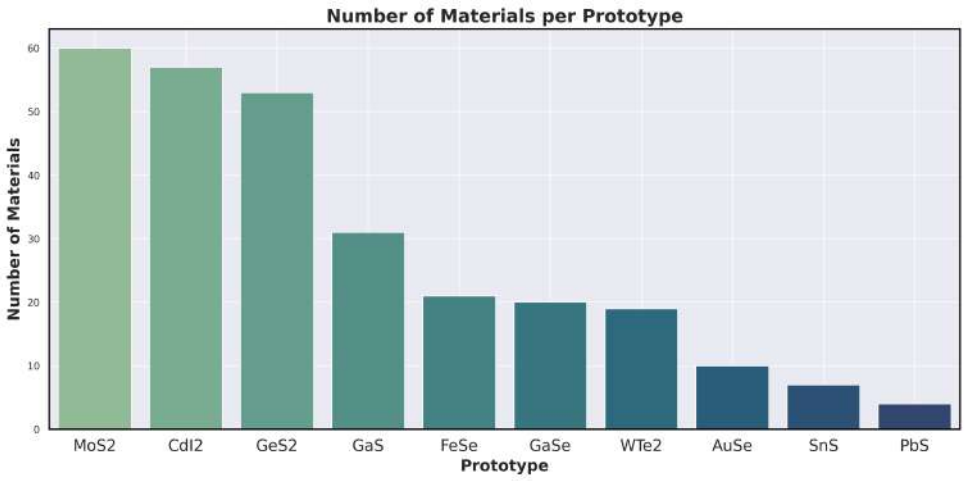

Figure 23 – Number of Materials by Prototype.

In figure 24 we can observe the distribution of the orbital hall conductivity of all prototypes.

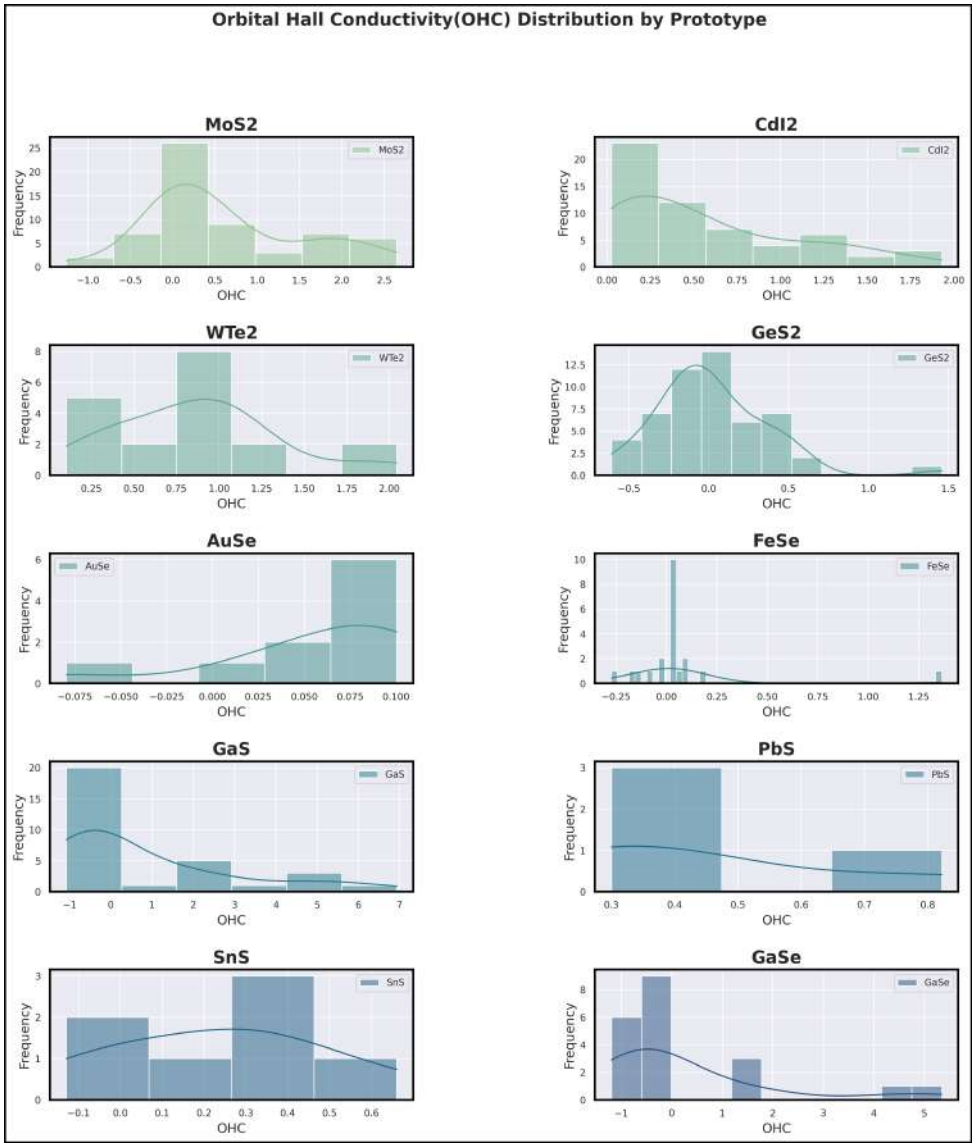

Figure 24 – Hall conductivity distribution by prototype.

In figure 25 one can observe the distribution of the orbital hall conductivity for all prototypes.

In figure 26 you can see a correlation analysis of the numerical variables with the orbital hall conductivity. Note that of the total feature space we have, only 23 of them have a high correlation, which can be taken into account during modeling in order to use fewer variables to build the model.

5.1.3.2 Regression Model - Simple Model

To build the simplest model we use all available attributes and build all models, in the table 14 we can see the calculation of the mean square error metrics, root mean square error, r2 score.

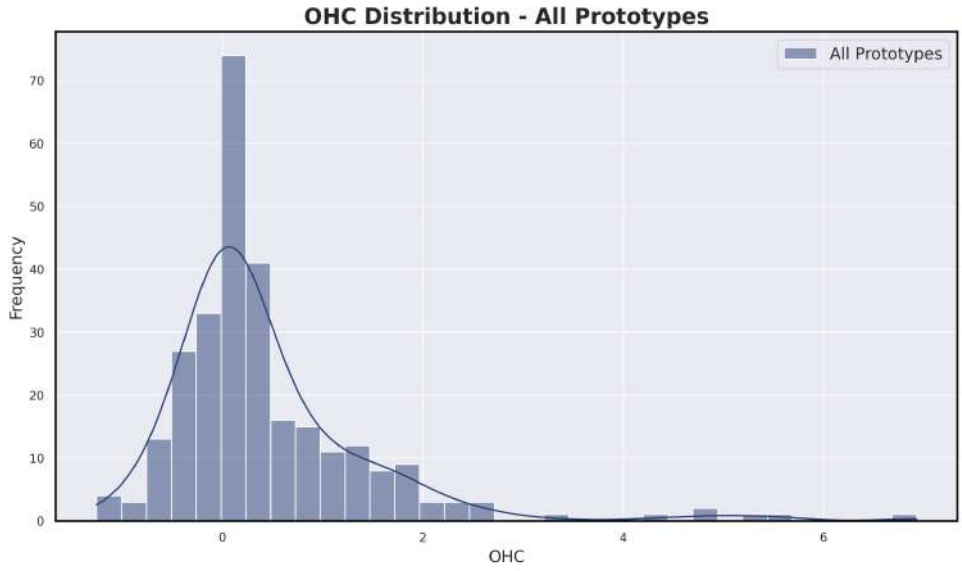

Figure 25 – Total distribution of orbital hall conductivity.

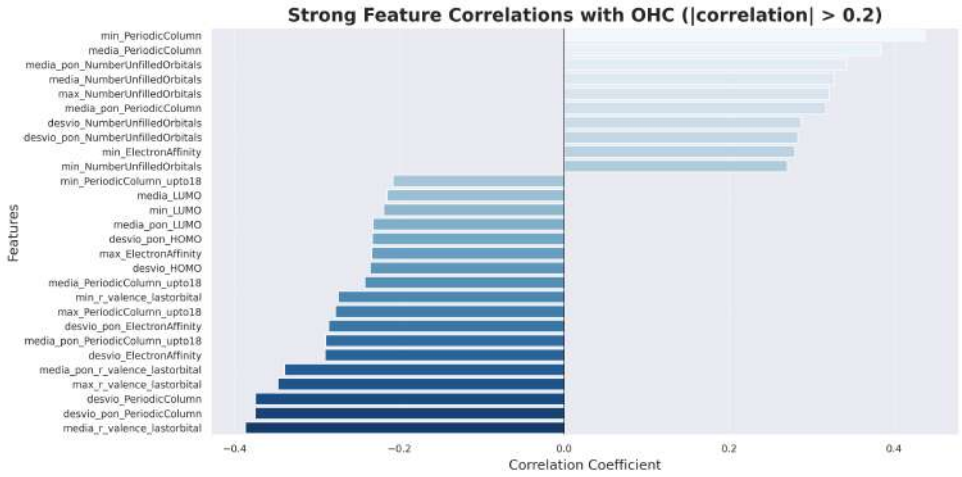

Figure 26 – Correlation between the variables used and the orbital hall conductivity.

Table 14 – Performance comparison of different machine learning algorithms using Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R2-Score.

Model	**MSE**	**RMSE**	**R2-Score**
XGBoosting	0.216	0.465	0.757
Extra Trees	0.229	0.478	0.743
Random Forest	0.239	0.488	0.732
Decision Tree	0.253	0.503	0.716
AdaBoosting	0.326	0.571	0.634
Ridge	0.534	0.731	0.401
Bayesian Regressor	0.554	0.744	0.378
KNN	0.698	0.836	0.216
Linear Regressor	16.2	4.03	-17.2

By analyzing the data in the table 14, it can be seen that XGBoosting presented the best results with the lowest MSE (0.216360) and RMSE (0.465145), in addition to a high R2-Score (0.757089). This indicates that the model was able to explain the variance of the data well, even with limited preparation.

Extra Trees and **Random Forest** showed competitive performance, with relatively low MSE and RMSE and R2-Score above 0.7. These models generally perform well on data with complex patterns due to their decision tree-based nature.

DecisionTree performed worse than **XGBoosting**, **Extra Trees**, and **Random Forest**, but still maintained an acceptable R2-Score (0.716078).

AdaBoosting showed worse performance, with R2-Score dropping to 0.634246. This could be due to limited data or a lack of hyperparameter fine-tuning.

Ridge and **Bayesian Regressor** performed moderately, with R2-Scores of 0.400746 (Ridge) and 0.378252 (Bayesian Regressor). These values suggest that the models failed to fully capture the non-linear patterns present in the data.

The **KNN** model only had an R2-Score of 0.216059, indicating that it failed to model the data well. This could be due to a lack of normalization or data preparation, which are crucial for KNN.

The **Linear Regressor** performed extremely poorly, with an MSE of 16.203664 and a negative R2-Score (-17.192159). This result suggests that the model was completely unsuited to the data, likely due to its inability to capture complex relationships.

The lack of data preparation (such as normalization, handling missing values, or creating derived variables) particularly impacted sensitive models such as KNN and linear regression.

Tree-based models (**XGBoosting**, **Extra Trees**, **Random Forest**) showed robustness even with limited preparation, suggesting that they are better suited to dealing with "raw" data.

Figure 27 presents a set of scatter plots comparing the values predicted by the models (y-axis) with the measured or actual values (x-axis) for different machine learning algorithms. The dashed line represents the ideal diagonal, where the predicted values would be equal to the measured values, indicating perfect prediction.

Models such as Random Forest, Gradient Boosting, and Extra Trees exhibit points that align more closely with the diagonal line, indicating their strong ability to capture the relationship between predictor variables and the target variable. In contrast, models like Linear Regression and K-Nearest Neighbors show greater scatter, demonstrating their lower effectiveness in modeling the data.

The Linear Regression plot clearly reveals underfitting, with many predicted values deviating significantly from the ideal line, highlighting its inability to capture complex patterns.

Dispersion at the extremes: Models such as Decision Tree and AdaBoost show greater

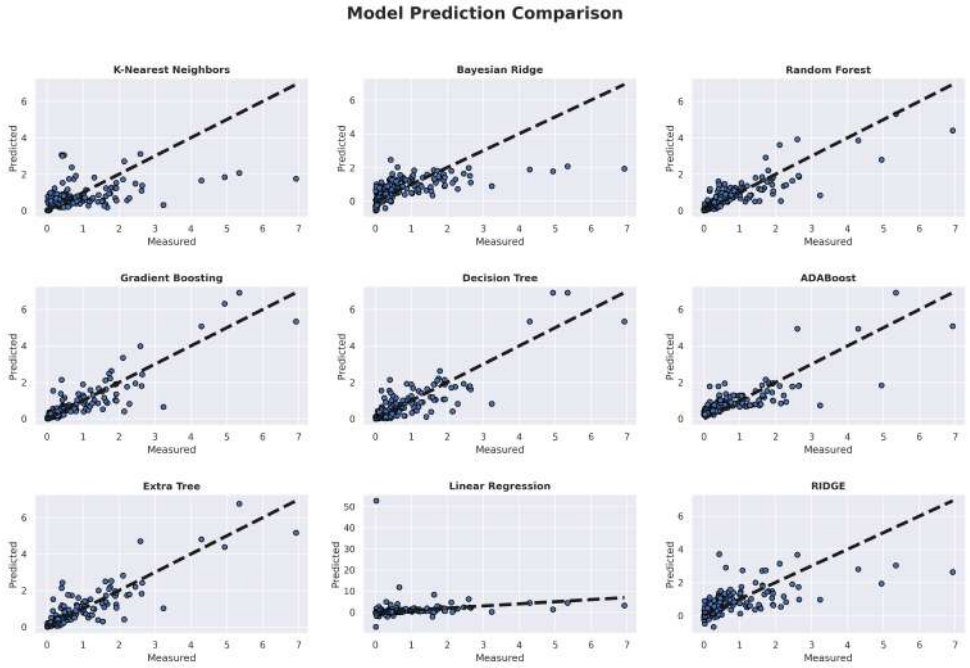

Figure 27 – Scatterplot of model predictions for the simple model. The closer the values are to the black dotted line, the better the model.

dispersion at extreme values (graph edges), suggesting difficulties in generalizing beyond the data's central range.

Data distribution: In all graphs, data points are most concentrated around lower "Measured" values, implying a possible bias in the dataset toward smaller values.

Overall, the closer proximity of points to the diagonal line in models like Random Forest and Gradient Boosting confirms their superior predictive performance.

5.1.3.3 Regression Model - With Data Preparation and Beast Features

The best model in the ensemble, with the lowest MSE (0.205363) and the highest R^2 (0.818442). The combination of randomness and splitting based on information maximization criteria allowed this model to efficiently capture patterns in the data.

Random Forest was the second best model, with an MSE of 0.264640 and R^2 of 0.766035. While close to the performance of Extra Trees, the smaller number of trees or more conservative parameters may have contributed to a slightly lower performance.

XGBoosting had an MSE of 0.285320 and R^2 of 0.747753, falling behind Extra Trees and Random Forest. Its solid performance demonstrates its ability to learn well from data, but it may be more sensitive to hyperparameter choices.

AdaBoosting (MSE: 0.452301) and Decision Tree (MSE: 0.479031) performed in the middle, but considerably worse than the more robust ensemble-based models.

Linear models (Bayesian Regressor, Ridge, Linear Regression) continue to perform the worst, with R^2 below 0.21. This confirms their inability to capture non-linear patterns in

the dataset, even with improved features.

Although KNN improves on the initial analysis, its performance (MSE: 0.486524, R^2: 0.569871) is still limited. The algorithm seems to be unable to cope well with high dimensionality or sparsity of the data.

Table 15 – Performance comparison of different machine learning algorithms using Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R2-Score.

Algorithm	**MSE**	**RMSE**	**R2-Score**
Extra Trees	0.205	0.453	0.818
Random Forest	0.265	0.514	0.766
XGBoosting	0.285	0.534	0.748
AdaBoosting	0.452	0.673	0.600
Decision Tree	0.479	0.692	0.576
KNN	0.487	0.698	0.570
Bayesian Regressor	0.895	0.946	0.209
Ridge	0.901	0.949	0.204
Linear Regressor	0.907	0.952	0.198

Random Forest and Extra Trees show the greatest proximity to the reference line over a wide range of values. This reinforces the numerical result, which points to these approaches as the most accurate. Gradient Boosting Despite performing well, it presents greater dispersion compared to Random Forest and Extra Trees.

Linear Regression and Ridge The predictions are significantly far from the line y=x, with high bias and poor generalization, consistent with the low values of R^2

KNN and Decision Tree Demonstrate better performance than linear models, but suffer from significant dispersion in some regions.

5.1.3.3.1 Model Performance Discussion

Among the models evaluated, tree-based ensemble methods such as Extra Trees, Random Forest, and XGBoost consistently achieved superior performance in both classification and regression tasks. This can be attributed to several advantageous characteristics inherent to these models. First, they are well-suited for capturing complex non-linear relationships and high-order feature interactions, which are likely present in the mapping from elemental descriptors to emergent material properties such as band gap or orbital Hall conductivity. Unlike linear models or K-Nearest Neighbors, tree ensembles do not assume linear separability or distance-based similarity, making them more robust to irregular decision boundaries.

Additionally, these models are inherently insensitive to monotonic transformations and do not require feature scaling, which is particularly useful when combining elemental properties of different physical dimensions (e.g., electronegativity vs. atomic mass). Their

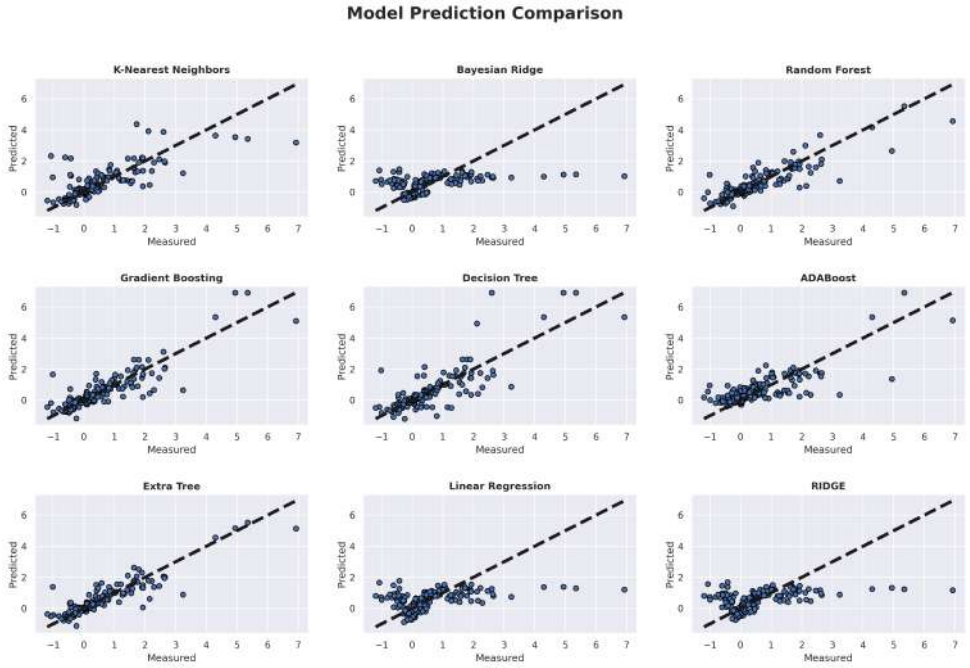

Figure 28 – Scatterplot of model predictions for model with Data Preparation and Beast Features. The closer the values are to the black dotted line, the better the model.

inductive bias—favoring hierarchical, rule-based partitioning of the feature space—aligns well with the heterogeneity present in materials data. Extra Trees, in particular, tends to reduce variance through randomized splits, offering better generalization in small to moderate-sized datasets, which is often the case in computational materials science. These factors collectively explain their strong performance in this study.

5.2 Model Predictions

In this section, we present the calculations for some selected materials based on the model predictions. From the predictions of materials classified as insulators and their respective orbital Hall conductivity (OHC) values, we perform more accurate calculations using Density Functional Theory (DFT) to determine the energy bands. Then, we use the PAOFLOW tool to calculate the OHC values with greater accuracy. This process allows us to validate the model predictions and evaluate them.

We use the model to make predictions for some materials. A brief sample of the results can be seen in Table 16, however the table is only a small sample with all the predicted insulators as well as the value of the orbital hall conductivity. We can see the complete predictions in the appendix A.5.

Table 16 – Predicted OHC and Band Gap for Different Materials

Material	**Prototype**	**ohc_predict**	**gap_predict**
ScCl2	MoS2	0.074	0.462
ScCl2	WTe2	0.081	0.463
ScI2	GeS2	-0.054	0.176
TiCl2	MoS2	0.158	0.252
TiCl2	WTe2	0.176	0.253
TiCl2	GeS2	0.086	0.252
TiBr2	MoS2	1.495	0.208
TiI2	GeS2	0.704	0.185
TiS2	MoS2	0.677	0.529
TiS2	WTe2	0.617	0.525
TiS2	GeS2	0.404	0.55
TiTe2	WTe2	1.027	0.179
TiTe2	GeS2	0.853	0.185
ZnF2	MoS2	-0.118	3.876
ZnF2	CdI2	0.022	3.895
ZnF2	WTe2	-0.079	3.668

5.2.1 DFT Calculations

First-principles calculations based on Density Functional Theory (DFT) were employed to compute key electronic and structural properties. These simulations provided reliable ground-truth data to train and validate the machine learning models developed in this work.

The calculations were performed using the plane-wave pseudopotential method within the Generalized Gradient Approximation (GGA) for the exchange-correlation functional. The pseudopotentials used are based on the fully relativistic PBE parameterization, as provided in the PSLibrary.

A prototypical WTe_2-like structure was selected to define the simulation conditions. The lattice parameter was set to 3.771325743 Å, and convergence tests were performed to ensure accuracy in both total energy and force calculations. A kinetic energy cutoff of 72.800 Ry was used for the wavefunctions, and 728.000 Ry for the charge density. Brillouin zone integrations were performed using a Monkhorst-Pack k-point mesh of $11 \times 7 \times 1$ with no shift. Force convergence was ensured below 0.01 eV/Å.

All the simulation parameters used in this study are summarized in Table 17.

Table 17 – Simulation parameters used in the DFT calculations

Parameter	**Value**
Prototype material	WTe_2
Lattice parameter (a)	3.771325743 Å
Pseudopotential for Zr	`Zr.rel-pbe-spn-kjpaw_psl.1.0.0.UPF`
Pseudopotential for Se	`Se.rel-pbe-dn-kjpaw_psl.1.0.0.UPF`
K-points (automatic)	$11 \times 7 \times 1$ (offsets: 0 0 0)
Wavefunction cutoff energy (ecutwfc)	72.800 Ry
Charge density cutoff energy (ecutrho)	728.000 Ry
Forces convergence threshold	0.01 eV/Å
Exchange-correlation functional	GGA

Figure 29 shows the energy bands of the material $ZrSe_2$. The prediction made by the model indicated that this material is an insulator, and this characteristic was confirmed by the calculations, which revealed an energy gap of 0.27 eV. This consistency between the model prediction and the results obtained through the energy bands reinforces the effectiveness of the model in identifying the electronic properties of materials.

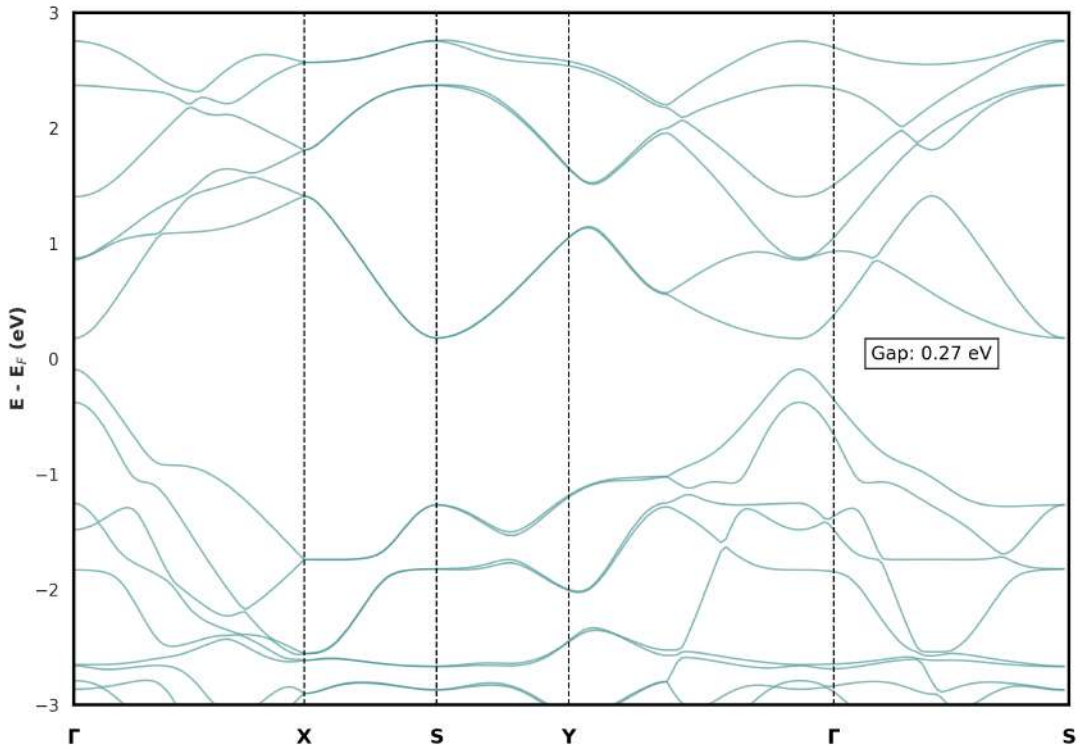

Figure 29 – Band structure for $ZrSe_2$.

Figure 30 shows the orbital Hall conductivity (OHC) curve of the material $ZrSe_2$. The calculated value for the OHC was -0.61 eV, while the model predicted a value of 0.72 eV. The difference between the predicted value and the calculated value is 0.11 eV, which is acceptable considering that the RMSE (Root Mean Squared Error) of the model is 0.46

eV. It is worth noting that the negative sign of the orbital Hall conductivity is related to the orientation of the material, and does not affect the magnitude of the prediction made by the model. These results demonstrate the ability of the model to provide estimates close to the true value, even with small discrepancies.

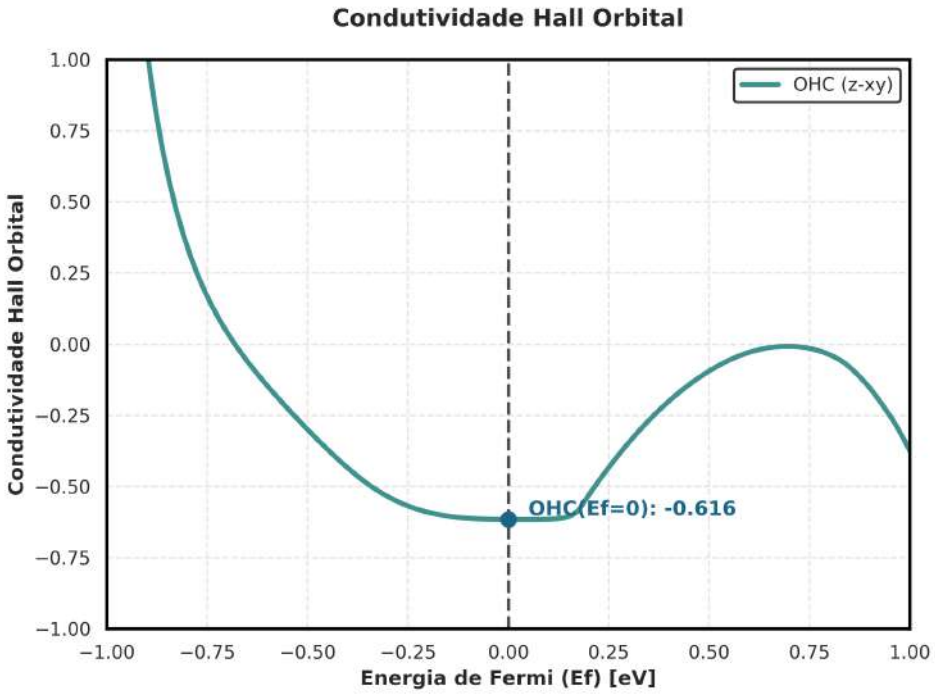

Figure 30 – Hall Orbital Conductivity Curve

5.3 Web Application for Material Classification and Orbital Hall Conductivity Prediction

As part of this project, an interactive web application was developed to explore the capabilities of machine learning models. The application was implemented using **Streamlit**, a Python framework for creating responsive web applications in an intuitive way and with simplified code.

5.3.1 Main Features

- **Material Classification**: Users can classify materials as metals or insulators based on their electronic structures. The application accepts material characteristics as input and provides classification results in real time.
- **Orbital Hall Conductivity (OHC) Prediction**: The application also allows predicting the orbital Hall conductivity of materials. From specific material properties, the regression model calculates the expected value of OHC.

- **Interactive Data Visualizations**:

  A variety of visualizations are integrated into the application, including:

  - Volume distribution by material prototype.
  - Comparisons between metals and insulators.
  - Histograms and boxplots showing the distribution of OHC predictions.

  These visualizations allow users to better understand the dataset and the model predictions.

- **Custom Filters**:

  The application includes tools to dynamically filter predictions. Users can specify ranges for OHC values or focus on specific prototypes for more detailed analysis.

- **Authentication**:

  To ensure data privacy and control, the application incorporates a simple authentication mechanism, allowing only authorized users to access the data.

5.3.2 Benefits and Applications

The web application connects theoretical research with practical applications, making the results of this dissertation accessible to a wider audience. Researchers, students, and industry professionals can use the application to:

- Quickly explore the dataset and model predictions.
- Perform *what-if* analyses by varying input parameters.
- Gain intuitive insights into the relationships between electronic structure and material behavior.

5.3.3 Implementation

The application was developed using **Python 3.9.16** and is hosted on a cloud platform, for the most current url check the code repository hosted at:

https://github.com/erickfog/Master$_P$*roject*.

Its modular design ensures that future improvements, such as new models or additional datasets, can be integrated.

In this figures 31 you can see the home page of the web application which contains a user authentication method and some descriptions about the project including how to perform the initial authentication.

Figure 31 – This is the home page of the website.

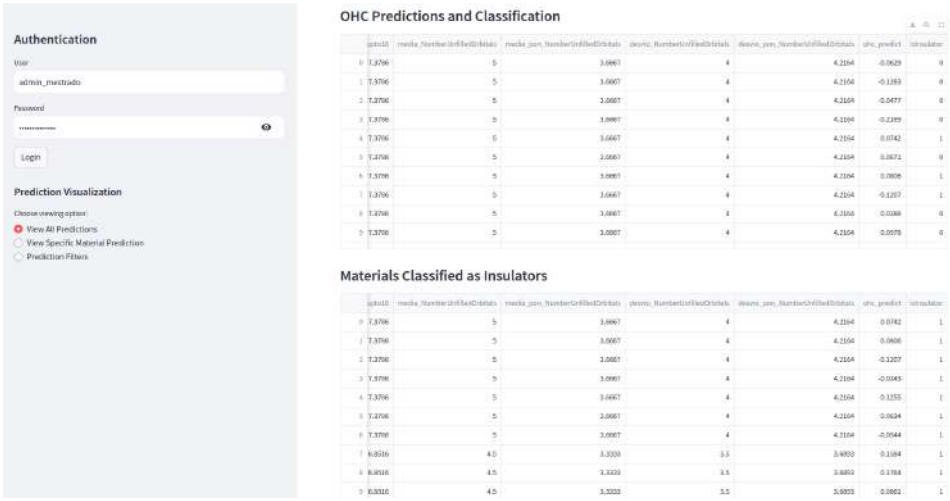

Figure 32 – Page with complete predictions (metals/insulator) and only insulators.

In figure 32 you can see the application page that delivers all the predictions made by the model, as well as exporting only the materials that are insulators.

In figure 33 you can see the page where it is possible to make a prediction for a specific material that the user wants by choosing the material and the prototype.

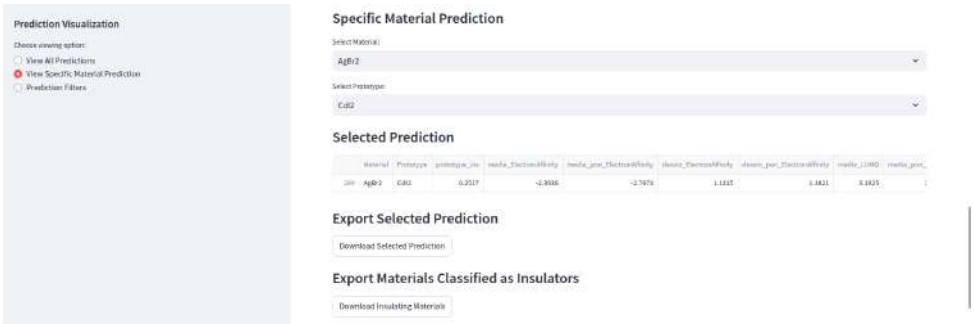

Figure 33 – Page with material selection and prototype for individual predictions.

In figure 34 you can see the page for filtering orbital hall conductivity predictions, and it is possible to export the filtered predictions.

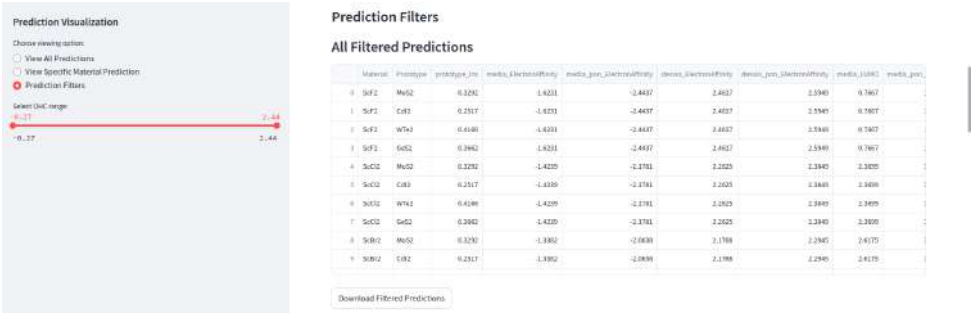

Figure 34 – Page for selecting specific ohc values.

In figure 35 you can see a part of the project's analytics where you can create customized views using the orbital hall conductivity filter.

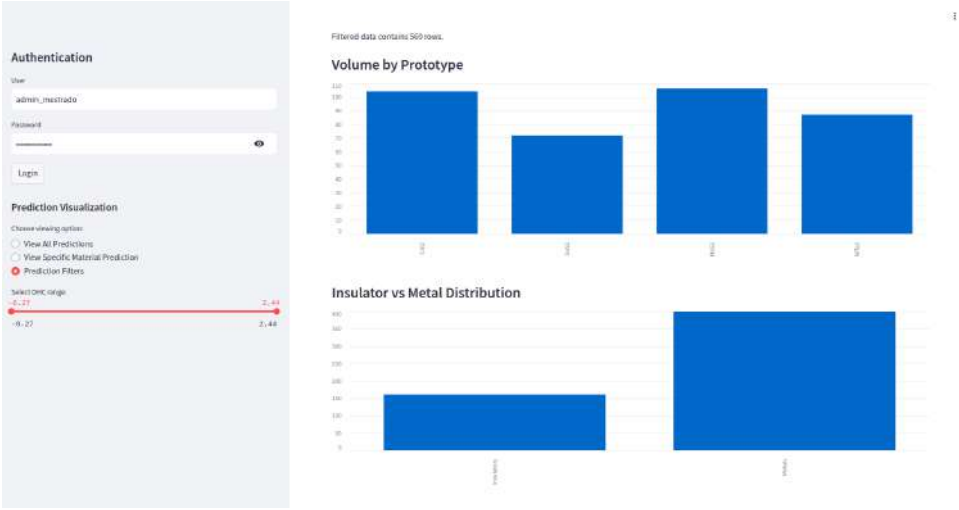

Figure 35 – Page with data visualization of selected ohc values.

6 Conclusions and Perspectives

This work presents a novel approach for classifying materials as metals or insulators and predicting their orbital Hall conductivity (OHC) using machine learning models. By exploiting the Computational 2D Materials Database (C2DB), it was possible to build an efficient pipeline that combines classification and regression techniques, significantly reducing the computational cost associated with first-principles calculations.

The classification between metals and insulators was performed with high accuracy, demonstrating the ability of machine learning models to capture fundamental characteristics of materials. The prediction of OHC for insulating materials proved to be an efficient and accurate alternative to traditional methods, highlighting the potential of data-driven models to complement computational studies in materials physics.

In addition to contributing to the advancement of understanding the quantum properties of two-dimensional materials, this work establishes a solid foundation for the integration of machine learning into materials discovery processes. This approach allows not only to reduce computational costs, but also to accelerate the development of new technologies based on functional materials.

The results demonstrated that the model presents good accuracy and robustness, with an RMSE of 0.46 and satisfactory performance in material classification. The model was able to correctly identify $ZrSe_2$ as an insulating material, and the prediction of the OHC (0.72 eV) presented an error of only 0.11 eV in relation to the value calculated via PAOFLOW (-0.61 eV). These results confirm the viability of machine learning models as complementary tools in the design and study of new functional materials.

In addition, the proposed approach allowed the efficient screening of candidate materials with potential for technological applications, optimizing computational and experimental resources. The integration of machine learning and advanced theoretical methods reinforces the potential of this approach to expand knowledge about material properties and accelerate the discovery and design process of new compounds.

The project achieved several important goals:

- **Accurate classification of materials**: Using supervised learning algorithms, it was possible to classify materials with high accuracy, demonstrating the potential of machine learning to assist in the identification of fundamental properties of matter.

- **Quantitative prediction of orbital Hall conductivity**: Using an efficient regression model, the work provided quantitative predictions for orbital Hall conductivity, highlighting its relationship with electronic characteristics of materials, especially at the critical points Γ, X, S, Y and Γ.

- **Development of an interactive web application**: The creation of an accessible tool for data analysis and visualization allows researchers and practitioners to explore the model's predictions and insights in a practical and intuitive way, bringing theoretical research closer to real-world applications.

This work opens promising avenues for the application of machine learning in computational physics and materials science. By connecting the intrinsic electronic properties of materials to the results of predictive models, it contributes to accelerating the design of new materials with specific properties.

Future perspectives include:

- **Exploring more complex architectures**: Investigating the use of deep neural networks or transfer learning models can further improve the accuracy of predictions.
- **Expanding the dataset**: Including experimental data or data generated by first-principles methods could increase the generalizability of the models.
- **Integration with other material properties**: Expanding the study to other physical properties, such as thermal conductivity and elasticity, diversifying the insights obtained.
- **Web application enhancement**:

  Incorporating new features, such as sensitivity analysis and support for multiple languages, can broaden the reach and impact of the application.

While this work focused primarily on compositional descriptors and conventional supervised learning models, several advanced techniques could be explored in future research to further enhance predictive power and applicability. For instance, incorporating explicit structural information using graph-based representations—such as those employed in graph neural networks (GNNs)—could enable more accurate modeling of orbital interactions and local coordination environments. Additionally, adopting physics-informed machine learning approaches, which embed domain knowledge or constraints into the learning process, may lead to more physically consistent predictions.

By bringing together advanced computational tools, machine learning, and theoretical physics, this work demonstrates the transformative potential of interdisciplinary approaches. The contributions presented here are expected to inspire future research and applications in materials science, fostering a better understanding of the fundamental properties of matter and encouraging the development of innovative technological solutions.

Bibliography

1 SCHLEDER, G. R. et al. From dft to machine learning: recent approaches to materials science–a review. *Journal of Physics: Materials*, IOP Publishing, v. 2, n. 3, p. 032001, may 2019. Disponível em: <https://dx.doi.org/10.1088/2515-7639/ab084b>. Citado 2 vezes nas páginas and 25.

2 KRISHNAN, N. M. A.; KODAMANA, H.; BHATTOO, R. *Machine learning for materials discovery*. 1. ed. Cham, Switzerland: Springer International Publishing, 2024. Citado na página 3.

3 LI, J. et al. Ai applications through the whole life cycle of material discovery. *Matter*, Elsevier BV, v. 3, n. 2, p. 393–432, ago. 2020. ISSN 2590-2385. Disponível em: <http://dx.doi.org/10.1016/j.matt.2020.06.011>. Citado na página 3.

4 LIU, Y. et al. Materials discovery and design using machine learning. *Journal of Materiomics*, Elsevier BV, v. 3, n. 3, p. 159–177, set. 2017. ISSN 2352-8478. Disponível em: <http://dx.doi.org/10.1016/j.jmat.2017.08.002>. Citado na página 3.

5 HIMANEN, L. et al. Data-driven materials science: Status, challenges, and perspectives. *Advanced Science*, Wiley, v. 6, n. 21, set. 2019. ISSN 2198-3844. Disponível em: <http://dx.doi.org/10.1002/advs.201900808>. Citado na página 3.

6 GUBERNATIS, J. E.; LOOKMAN, T. Machine learning in materials design and discovery: Examples from the present and suggestions for the future. *Physical Review Materials*, American Physical Society (APS), v. 2, n. 12, dez. 2018. ISSN 2475-9953. Disponível em: <http://dx.doi.org/10.1103/PhysRevMaterials.2.120301>. Citado na página 3.

7 BHOWAL, S.; SATPATHY, S. Intrinsic orbital and spin hall effects in monolayer transition metal dichalcogenides. *Physical Review B*, American Physical Society (APS), v. 102, n. 3, jul. 2020. ISSN 2469-9969. Disponível em: <http://dx.doi.org/10.1103/PhysRevB.102.035409>. Citado na página 3.

8 ZHOU, M. et al. Formation of quantum spin hall state on si surface and energy gap scaling with strength of spin orbit coupling. *Scientific Reports*, Springer Science and Business Media LLC, v. 4, n. 1, nov. 2014. ISSN 2045-2322. Disponível em: <http://dx.doi.org/10.1038/srep07102>. Citado na página 4.

9 LEITAO, D. C. et al. Enhanced performance and functionality in spintronic sensors. *npj Spintronics*, Springer Science and Business Media LLC, v. 2, n. 1, nov. 2024. ISSN 2948-2119. Disponível em: <http://dx.doi.org/10.1038/s44306-024-00058-9>. Citado na página 4.

10 CYSNE, T. P. et al. *Orbitronics in Two-dimensional Materials*. 2025. Disponível em: <https://arxiv.org/abs/2502.12339>. Citado na página 4.

11 BERNEVIG, B. A.; HUGHES, T. L.; ZHANG, S.-C. Orbitronics: The intrinsic orbital current in<mml:math xmlns:mml="http://www.w3.org/1998/math/mathml" display="inline"><mml:mi>p</mml:mi></mml:math>-doped silicon. *Physical Review*

Letters, American Physical Society (APS), v. 95, n. 6, ago. 2005. ISSN 1079-7114. Disponível em: <http://dx.doi.org/10.1103/PhysRevLett.95.066601>. Citado na página 6.

12 GO, D. et al. Intrinsic spin and orbital hall effects from orbital texture. *Phys. Rev. Lett.*, American Physical Society, v. 121, p. 086602, Aug 2018. Disponível em: <https://link.aps.org/doi/10.1103/PhysRevLett.121.086602>. Citado na página 6.

13 WANG, P. et al. Orbitronics: Mechanisms, materials and devices. *Advanced Electronic Materials*, Wiley, dez. 2024. ISSN 2199-160X. Disponível em: <http://dx.doi.org/10.1002/aelm.202400554>. Citado na página 6.

14 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, sep 2018. Disponível em: <https://dx.doi.org/10.1088/2053-1583/aacfc1>. Citado 2 vezes nas páginas 8 and 52.

15 KOHN, W.; SHAM, L. J. Self-consistent equations including exchange and correlation effects. *Phys. Rev.*, American Physical Society, v. 140, p. A1133–A1138, Nov 1965. Disponível em: <https://link.aps.org/doi/10.1103/PhysRev.140.A1133>. Citado 2 vezes nas páginas 9 and 19.

16 GIUSTINO, F. Materials Modelling using Density Functional Theory. Properties and Predictions. p. 303, 2014. Citado 3 vezes nas páginas 10, 13, and 17.

17 TRZESNIAK, Daniel Rodrigo Ferreira. Modelagem quântica de inibidores enzimáticos [doi:10.11606/D.43.2002.tde-26062002-143722]. São Paulo : Instituto de Física, Universidade de São Paulo, 2002. Dissertação de Mestrado em Física. [acesso 2021-04-22]. Disponível em: <https://repositorio.ufscar.br/handle/ufscar/5073>. Citado na página 10.

18 AMORIM, Rodrigo Garcia. Cálculos de primeiros princípios para BaO.. 2005. 95 f. Dissertação (Mestrado em Ciências Exatas e da Terra) - Universidade Federal de São Carlos, São Carlos, 2005. Disponível em: <https://repositorio.ufscar.br/handle/ufscar/5073>. Citado 3 vezes nas páginas 11, 14, and 15.

19 TEORIA do Funcional da Densidade. <https://www.maxwell.vrac.puc-rio.br/16578/16578_3.PDF>. Acessado em 22/04/2021. Citado na página 11.

20 ROSTGAARD, C. *The Projector Augmented-wave Method*. 2009. Citado 2 vezes nas páginas 18 and 19.

21 GIANNOZZI, P. et al. Quantum espresso: a modular and open-source software project for quantum simulations of materials. *Journal of Physics: Condensed Matter*, IOP Publishing, v. 21, n. 39, p. 395502, set. 2009. ISSN 1361-648X. Disponível em: <http://dx.doi.org/10.1088/0953-8984/21/39/395502>. Citado na página 20.

22 Buongiorno Nardelli, M. et al. Paoflow: A utility to construct and operate on ab initio hamiltonians from the projections of electronic wavefunctions on atomic orbital bases, including characterization of topological materials. *Computational Materials Science*, v. 143, p. 462–472, 2018. ISSN 0927-0256. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0927025617306651>. Citado na página 20.

23 FIX, E.; HODGES, J. L. Discriminatory analysis - nonparametric discrimination: Consistency properties. *International Statistical Review*, v. 57, p. 238, 1989. Disponível em: <https://api.semanticscholar.org/CorpusID:120323383>. Citado na página 25.

24 COVER, T.; HART, P. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, v. 13, n. 1, p. 21–27, 1967. Citado na página 25.

25 FACELI, K. e. a. *Inteligência artificial: uma abordagem de aprendizado de máquina*. Rio de Janeiro: LTC, 2011. Citado 4 vezes nas páginas 25, 26, 32, and 33.

26 QUINLAN, J. R. Induction of decision trees. *Machine Learning*, Springer Science and Business Media LLC, v. 1, n. 1, p. 81–106, mar. 1986. ISSN 1573-0565. Disponível em: <http://dx.doi.org/10.1007/BF00116251>. Citado na página 28.

27 SALZBERG, S. L. C4.5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993. *Machine Learning*, Springer Science and Business Media LLC, v. 16, n. 3, p. 235–240, set. 1994. ISSN 1573-0565. Disponível em: <http://dx.doi.org/10.1007/BF00993309>. Citado na página 28.

28 STEINBERG, D. Cart: Classification and regression trees. In: . [s.n.], 2009. Disponível em: <https://api.semanticscholar.org/CorpusID:116184048>. Citado na página 28.

29 FRIEDMAN, J. H. Stochastic gradient boosting. *Computational Statistics amp; Data Analysis*, Elsevier BV, v. 38, n. 4, p. 367–378, fev. 2002. ISSN 0167-9473. Disponível em: <http://dx.doi.org/10.1016/S0167-9473(01)00065-2>. Citado na página 30.

30 FRIEDMAN, J. H. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, Institute of Mathematical Statistics, v. 29, n. 5, p. 1189–1232, 2001. ISSN 00905364, 21688966. Disponível em: <http://www.jstor.org/stable/2699986>. Citado 2 vezes nas páginas 30 and 31.

31 CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. arXiv, 2016. Disponível em: <https://arxiv.org/abs/1603.02754>. Citado na página 32.

32 LI, F. et al. A light gradient boosting machine for remainning useful life estimation of aircraft engines. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018. Disponível em: <http://dx.doi.org/10.1109/ITSC.2018.8569801>. Citado na página 32.

33 PROKHORENKOVA, L. et al. *CatBoost: unbiased boosting with categorical features*. arXiv, 2017. Disponível em: <https://arxiv.org/abs/1706.09516>. Citado na página 32.

34 YANG, F.-J. An implementation of naive bayes classifier. In: *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*. [S.l.: s.n.], 2018. p. 301–306. Citado na página 32.

35 VIKRAMKUMAR; B, V.; TRILOCHAN. *Bayes and Naive Bayes Classifier*. arXiv, 2014. Disponível em: <https://arxiv.org/abs/1404.0933>. Citado na página 32.

36 HEARST, M. et al. Support vector machines. *IEEE Intelligent Systems and their Applications*, v. 13, n. 4, p. 18–28, 1998. Citado na página 33.

37 CHAWLA, N. V. et al. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, AI Access Foundation, v. 16, p. 321–357, jun. 2002. ISSN 1076-9757. Disponível em: <http://dx.doi.org/10.1613/jair.953>. Citado na página 43.

38 HE, H. et al. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. [S.l.: s.n.], 2008. p. 1322–1328. Citado na página 43.

39 SNOEK, J.; LAROCHELLE, H.; ADAMS, R. P. *Practical Bayesian Optimization of Machine Learning Algorithms*. 2012. Disponível em: <https://arxiv.org/abs/1206.2944>. Citado na página 50.

40 JAIN, A. K.; DUBES, R. C. *Algorithms for clustering data*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988. Disponível em: <http://portal.acm.org/citation.cfm?id=46712>. Citado na página 81.

41 MORISSETTE, L.; CHARTIER, S. The k-means clustering technique: General considerations and implementation in mathematica. *Tutorials in Quantitative Methods for Psychology*, TQMP, v. 9, n. 1, p. 15–24, 2013. Disponível em: <http://www.tqmp.org/RegularArticles/vol09-1/p015/p015.pdf>. Citado na página 81.

42 ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: . AAAI Press, Menlo Park, CA (United States), 1996. Disponível em: <https://www.osti.gov/biblio/421283>. Citado na página 82.

43 JOLLIFFE, I. T.; CADIMA, J. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, The Royal Society, v. 374, n. 2065, p. 20150202, abr. 2016. ISSN 1471-2962. Disponível em: <http://dx.doi.org/10.1098/rsta.2015.0202>. Citado na página 82.

44 AGRAWAL, R.; IMIELIńSKI, T.; SWAMI, A. Mining association rules between sets of items in large databases. In: *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*. ACM, 1993. (SIGMOD/PODS93). Disponível em: <http://dx.doi.org/10.1145/170035.170072>. Citado na página 83.

45 SUTTON, R. S.; BARTO, A. G. *Reinforcement learning: An introduction*. [S.l.]: The MIT Press, 2020. Citado na página 84.

46 WATKINS, C. J. C. H.; DAYAN, P. Q-learning. *Machine Learning*, Springer Science and Business Media LLC, v. 8, n. 3–4, p. 279–292, maio 1992. ISSN 1573-0565. Disponível em: <http://dx.doi.org/10.1007/BF00992698>. Citado na página 84.

47 ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, American Psychological Association (APA), v. 65, n. 6, p. 386–408, 1958. ISSN 0033-295X. Disponível em: <http://dx.doi.org/10.1037/h0042519>. Citado na página 85.

A Appendix

A.1 Unsupervised Learning

A.1.1 Clustering

There are several unsupervised learning algorithms, but each algorithm is more suitable depending on how the data is dispersed in the data space. The aim of a clustering task is to find a structure of clusters in the data in which the objects belonging to each cluster share some characteristic or property relevant to the domain of the problem under study, i.e. they are similar in some way[40].

A.1.1.1 (K-means | DBSCAN)

K-Means is a partitioning algorithm whose objective is to minimize the sum of the quadratic distances between data points and the center of their respective clusters. Its operation involves the following steps:

Initialization: A number of clusters is define k and the initial centroids are chosen (usually randomly).

Assignment: Each point is associated with the cluster whose centroid is closest, according to a distance measure (usually the Euclidean distance).

Update: The centroids are recalculated as the average of the points assigned to each cluster.

Iteration: The assignment and update steps are repeated until there are no more significant changes in the centroids or a maximum number of iterations is reached.

The simplicity and computational efficiency of K-Means make it a popular choice, especially in applications with large volumes of data. However, it has important limitations, such as the need to previously define the number of clusters and the sensitivity to outliers and the initialization of centroids, which can influence convergence to suboptimal solutions [41].

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) adopts a density-based approach to identify clusters in data sets. Unlike K-Means, DBSCAN does not require prior specification of the number of clusters. Its fundamental concepts include:

Eps (): The maximum radius that defines the neighborhood of a point. minPts: The minimum number of points required for a region to be considered densely populated and, therefore, constitute a cluster. The algorithm classifies points into three categories:

Central points: Points that have at least minPts minPts points in their neighborhood of radius . Border points: Points that are in the neighborhood of a central point, but

do not meet the density condition to be centers. Outliers (noise): Points that do not belong to any dense neighborhood. DBSCAN is particularly useful for detecting clusters of arbitrary shapes and dealing with noise in a robust manner. However, the appropriate definition of the parameters and minPts minPts can be challenging, especially in datasets with heterogeneous densities [(42)].

A.1.2 Dimensionality Reduction

Dimensionality reduction is the process of decreasing the number of attributes available for training an algorithm.

A.1.2.1 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a fundamental statistical technique for dimensionality reduction and exploratory data analysis. It transforms a set of correlated variables into a new set of uncorrelated variables, called principal components, that retain most of the variability of the original data[(43)].

Fundamental Concepts

• Dimensionality Reduction: PCA projects the data into a new space, where each axis (principal component) represents a direction of maximum variance. This transformation allows the number of variables to be reduced while maintaining as much information as possible.

• Principal Components: The first principal component captures the largest variance of the data. Each subsequent component, orthogonal to the previous ones, captures the largest remaining variance. Thus, it is possible to order the components by the amount of variance explained.

• Covariance Matrix and Eigenvalues: To identify the directions of greatest variability, the covariance matrix of the data is calculated. From it, the eigenvalues (which indicate the variance explained by each component) and the eigenvectors (which define the direction of the components) are extracted.

A.1.2.2 PCA Steps

-processing: Normalization or standardization of data so that each variable contributes equally to the analysis.

• Calculation of the Covariance Matrix: Identifies how variables relate to each other, highlighting existing correlations.

• Calculation of Eigenvalues and Eigenvectors: Determines the directions (eigenvectors) and relative importance (eigenvalues) of each principal component.

• Selecting Components: A subset of the principal components is chosen that, together, explain a significant percentage of the total variance of the data.

• Projecting Data: The original data is projected into the space formed by the selected components, resulting in a more compact representation with fewer dimensions.

A.1.3 Associantion Rules

Association rules are expressions of the type "if A, then B", where A and B represent sets of items (itemsets) that frequently occur together in a dataset. This technique is widely used in market basket analysis, where the objective is to identify, for example, which products are usually purchased together.

A.1.3.1 Main Metrics

• Support: Represents the frequency with which an item or set of items appears in the dataset. It is defined as the proportion of transactions that contain the itemset in question.

• Confidence: Measures the probability of item B occurring given that item A is present. In other words, it is the proportion of transactions that contain A and B in relation to the total number of transactions that contain A.

• Lift: Indicates how much the joint occurrence of A and B differs from the occurrence that would be expected if A and B were independent events. A lift greater than 1 suggests that there is a positive association between the items.

A.1.3.2 Apriori Algorithm

The Apriori algorithm, proposed by Agrawal et al. (1993)[(44)], is one of the most traditional methods for discovering frequent itemsets and extracting association rules. It is based on the principle that "all subsets of a frequent itemset must also be frequent". This property allows the search space to be significantly reduced, making the process computationally more feasible.

A.1.3.3 Algorithm Steps

• Candidate Generation: Initially, the algorithm identifies all itemsets of size 1 that meet the minimum support specified. Then, itemsets of larger size are generated from the combination of the frequent itemsets found in the previous iteration.

• Pruning: During candidate generation, itemsets that do not have all their frequent subsets are eliminated, since they cannot be frequent according to the Apriori principle.

• Support Calculation: For each candidate, the actual support is calculated from the data set. Only candidates that meet or exceed the minimum support threshold are kept for the next iteration.

• Rule Generation: From the obtained frequent itemsets, association rules are extracted. Each rule is evaluated in terms of confidence and, if this metric meets the established minimum threshold, the rule is considered significant.

A.1.3.4 Advantages and Limitations

• Advantages: Efficient in reducing the search space through candidate pruning. Applicability in several contexts, such as consumer analysis, bioinformatics and social networks.

• Limitations: It can become computationally intensive in datasets with a very high number of items, due to the exponential growth of candidates. Adequately defining the support and confidence thresholds is crucial to avoid both the generation of many irrelevant rules and the loss of interesting patterns.

A.2 Reinforcement Learning

Reinforcement learning is a machine learning paradigm in which an agent learns to make sequential decisions through interaction with an environment. The agent's objective is to learn an action policy that maximizes a measure of reward over time.

In reinforcement learning, the agent observes the current state of the environment, chooses an action based on a policy, and then receives a reward and transitions to a new state. The goal is to find a policy that maximizes the total expected return, that is, the sum of the discounted future rewards[45].

A.2.1 Q-learning

Q-learning is an off-policy algorithm that learns the action-value function, known as the Q-function, from interaction with the environment. This function estimates the quality (or value) of executing a given action in a given state, and is defined by the following update formula[46]:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{A.1}$$

- s and a represent, respectively, the current state and the action taken.
- r is the reward received after the transition.
- s' is the new state.
- α is the learning rate, which determines how much the new values influence the old ones.
- γ is the discount factor, which weights the importance of future rewards.
- $\max_{a'} Q(s',a')$ represents the best estimate of the value for the next state s'.

Main Features

• Model-free: Q-learning does not require prior knowledge about the dynamics of the environment, learning only through direct interaction.

• Convergence: Under appropriate conditions (e.g., a suitable exploration policy and well-defined learning parameters), the algorithm converges to the optimal Q function, allowing the derivation of an optimal policy.

• Flexibility: It can be applied to problems with discrete state and action spaces, although there are extensions (such as the Deep Q-Network) to deal with continuous or high-dimensional spaces.

A.3 Deep Learning

In deep learning, models are composed of multiple processing layers. Each layer extracts information at different levels of abstraction, starting with simpler features and building more complex representations of the data as the layers go deeper. This layered structure allows neural networks to learn highly non-linear patterns, which is essential for tasks involving large volumes and variability of data.

The perceptron is the simplest neural network model and one of the first supervised learning algorithms. Developed by Frank Rosenblatt in the 1950s[(47)], it serves as the basic building block for more complex neural networks. The following are the main aspects of the perceptron:

Input: The data is presented in the form of an input vector:

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]$$

Weights and Weighted Sum: Each input is associated with a weight w_i. The weighted sum of the inputs is calculated, often by adding a bias term b. The equation is given by:

$$z = \sum_{i=1}^{n} w_i x_i + b \tag{A.2}$$

Activation Function: The weighted sum z is then passed through an activation function, which introduces nonlinearity into the model. In its original form, the perceptron uses the step function, where the output y is defined as:

$$y = \begin{cases} 1, & \text{if } z \geq 0, \\ 0, & \text{otherwise.} \end{cases} \tag{A.3}$$

Output: The result of the activation function constitutes the output of the perceptron, determining the class to which the input example belongs.

The Figure 36 illustrates a simple perceptron-based neural network, consisting of an input layer, an activation function, and an output layer:

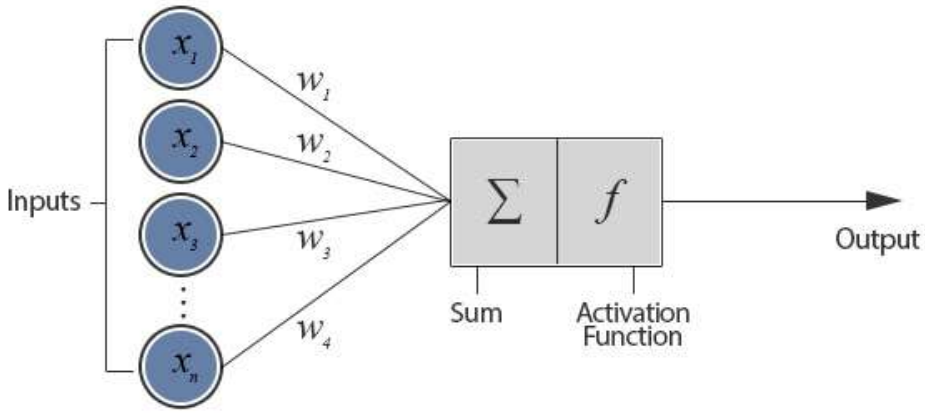

Figure 36 – Perceptron illustrate.

A.3.0.1 Neural Network Architectures

Neural network architectures define the structure and flow of information within a deep learning model, determining how data is processed and transformed across multiple layers. Below, I cover some common concepts and types of architectures:

Basic Structure

- **Input Layer:** Receives the raw data. Each node in this layer represents a feature of the input data.
- **Hidden Layers:** Are responsible for extracting representations and patterns from the data. The depth (number of hidden layers) and width (number of neurons in each layer) directly influence the modeling ability of the network.
- **Output Layer:** Provides the final prediction or classification. Its format and activation function depend on the task (e.g., regression or classification).
- **Connections and Weights:** Each connection between neurons is associated with a weight, which is adjusted during training to minimize model error.

Types of Architectures

- **Feedforward Networks (Multilayer Perceptron):** These are the simplest architectures, where data flows unidirectionally from the input layer to the output layer, passing through one or more hidden layers. These networks are the basis for more complex models.
- **Convolutional Neural Networks (CNNs):** Designed to deal with data that have spatial structure, such as images. They use convolution layers to extract local

features, followed by pooling layers to reduce dimensionality and fully connected layers for classification.

- **Recurrent Neural Networks (RNNs):** Suitable for processing sequential data, such as time series or text. They have connections that form cycles, allowing information from previous states to influence current processing. Extensions such as LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) help mitigate gradient problems and capture long-term dependencies.
- **Autoencoders:** Mainly used for unsupervised learning and dimensionality reduction, these models learn to reconstruct the input data, forcing the network to extract the most meaningful representations of the data.
- **Generative Adversarial Networks (GANs):** Composed of two networks – a generator and a discriminator – that compete with each other. This architecture has been widely used in the generation of synthetic data, such as realistic images and videos.

Considerations and Challenges

- **Depth and Complexity:** Deeper architectures can capture complex patterns, but they also require more computational power and care to avoid problems such as vanishing or exploding gradients.
- **Activation Functions:** Choosing appropriate activation functions (such as ReLU, sigmoid, or tanh) is crucial to introduce nonlinearities and allow the network to model complex relationships.
- **Regularization and Generalization:** Techniques such as dropout, batch normalization, and L2 regularization are employed to avoid overfitting and improve the generalization ability of the model.
- **Design and Experimentation:** The design of the architecture depends on the task and the available data. It is often necessary to experiment with different configurations to find the structure that best fits the problem at hand.

A.4 Deployments Strategy

In the context of software development and machine learning models, deployment strategies refer to the methods employed to put a model or application into production. Choosing the appropriate strategy depends on factors such as latency requirements, data volume, update frequency, and scalability needs. The following are five common deployment strategies:

1. **One-Off Deployment:** In this approach, the application or model is deployed once, usually in a controlled environment. This strategy is appropriate for applications that do not require continuous updates or frequent interactions after initial deployment.

2. **Batch Deployment:** Involves processing and deploying data or tasks in batches, at scheduled intervals. This approach is suitable for scenarios where real-time processing is not essential, allowing periodic execution of tasks efficiently.

3. **Real-Time Deployment:** In this strategy, models or applications are deployed in a way that responds instantly to inputs or events, providing immediate feedback. It is crucial for systems that require fast responses, such as fraud detection or recommendation systems.

4. **Streaming Deployment:** Intended for environments where data is generated continuously, this approach allows real-time processing of continuous information streams. It is ideal for real-time monitoring and data analysis applications.

5. **Edge Deployment:** Consists of deploying models or applications directly to edge devices (such as IoT devices or smartphones) instead of in centralized environments in the cloud. This strategy reduces latency, improves data privacy, and enables operations even in environments with limited connectivity.

Each of these strategies presents its own advantages and challenges, and choosing the most appropriate approach should take into account the specific requirements of the project.

A.5 Predictions

Table 18 – Predicted OHC and Band Gap for Different Materials

Material	**Prototype**	**ohc_predict**	**gap_predict**
ScCl2	MoS2	0.074	0.462
ScCl2	WTe2	0.081	0.463
ScCl2	GeS2	-0.121	0.462
ScBr2	WTe2	-0.024	0.205
ScI2	MoS2	0.126	0.176
ScI2	WTe2	0.083	0.167
ScI2	GeS2	-0.054	0.176
TiCl2	MoS2	0.158	0.252
TiCl2	WTe2	0.176	0.253
TiCl2	GeS2	0.086	0.252

Material	**Prototype**	**ohc_predict**	**gap_predict**
TiBr2	MoS2	1.495	0.208
TiBr2	WTe2	1.257	0.214
TiBr2	GeS2	0.612	0.208
TiI2	MoS2	1.436	0.185
TiI2	WTe2	0.83	0.176
TiI2	GeS2	0.704	0.185
TiS2	MoS2	0.677	0.529
TiS2	WTe2	0.617	0.525
TiS2	GeS2	0.404	0.55
TiSe2	MoS2	0.729	0.328
TiSe2	WTe2	0.678	0.329
TiSe2	GeS2	0.424	0.349
TiTe2	MoS2	1.187	0.164
TiTe2	WTe2	1.027	0.179
TiTe2	GeS2	0.853	0.185
ZnF2	MoS2	-0.118	3.876
ZnF2	CdI2	0.022	3.895
ZnF2	WTe2	-0.079	3.668
ZnF2	GeS2	-0.116	3.876
ZnCl2	MoS2	-0.013	3.459
ZnCl2	CdI2	0.253	3.478
ZnCl2	WTe2	0.047	3.251
ZnCl2	GeS2	-0.11	3.459
ZnBr2	MoS2	-0.129	2.815
ZnBr2	CdI2	0.241	2.834
ZnBr2	WTe2	-0.043	2.612
ZnBr2	GeS2	-0.131	2.815
ZnI2	MoS2	0.064	1.328
ZnI2	CdI2	0.374	1.347
ZnI2	WTe2	0.123	1.125
ZnI2	GeS2	-0.08	1.328
ZnS2	WTe2	0.15	0.462
ZnSe2	WTe2	0.143	0.508
ZnTe2	WTe2	0.196	0.206
ZrCl2	MoS2	0.245	0.343
ZrCl2	WTe2	0.209	0.344
ZrCl2	GeS2	0.169	0.343
ZrBr2	MoS2	0.262	0.279

Material	**Prototype**	**ohc_predict**	**gap_predict**
ZrBr2	WTe2	0.14	0.286
ZrBr2	GeS2	0.165	0.279
ZrI2	MoS2	0.936	0.262
ZrI2	WTe2	0.693	0.268
ZrI2	GeS2	0.559	0.262
ZrS2	MoS2	0.644	0.703
ZrS2	CdI2	0.95	0.718
ZrS2	WTe2	0.596	0.713
ZrS2	GeS2	0.471	0.724
ZrSe2	MoS2	0.78	0.398
ZrSe2	CdI2	1.244	0.413
ZrSe2	WTe2	0.727	0.414
ZrSe2	GeS2	0.514	0.419
ZrTe2	MoS2	0.678	0.137
ZrTe2	WTe2	0.655	0.168
ZrTe2	GeS2	0.523	0.157
MoS2	MoS2	2.442	0.195
MoS2	WTe2	1.942	0.2
MoS2	GeS2	2.065	0.195
MoSe2	MoS2	2.347	0.135
MoSe2	WTe2	1.767	0.145
MoSe2	GeS2	1.97	0.135
MoTe2	MoS2	2.167	0.132
RuCl2	CdI2	0.452	0.423
RuCl2	WTe2	0.611	0.436
RuCl2	GeS2	0.332	0.216
RuBr2	CdI2	0.552	0.405
RuBr2	WTe2	0.709	0.423
RuBr2	GeS2	0.439	0.198
RuI2	CdI2	0.737	0.41
RuI2	GeS2	0.582	0.203
RuS2	WTe2	1.026	0.254
RuSe2	WTe2	1.07	0.143
RuTe2	WTe2	1.056	0.122
RhF2	WTe2	0.89	0.354
PdS2	CdI2	1.3	0.489
PdS2	WTe2	1.213	0.511
PdS2	GeS2	1.155	0.432

Material	**Prototype**	**ohc_predict**	**gap_predict**
PdSe2	CdI2	1.748	0.238
CdF2	MoS2	-0.108	2.579
CdF2	CdI2	-0.004	2.598
CdF2	WTe2	-0.07	2.636
CdF2	GeS2	-0.147	2.579
CdCl2	MoS2	-0.093	2.717
CdCl2	CdI2	0.219	2.736
CdCl2	WTe2	-0.04	2.759
CdCl2	GeS2	-0.119	2.717
CdBr2	MoS2	-0.106	1.85
CdBr2	CdI2	0.294	1.869
CdBr2	WTe2	-0.023	1.896
CdBr2	GeS2	-0.12	1.85
CdI2	MoS2	-0.061	1.639
CdI2	CdI2	0.341	1.658
CdI2	WTe2	0.027	1.685
CdI2	GeS2	-0.133	1.639
CdS2	WTe2	0.134	0.525
CdSe2	WTe2	0.149	0.439
CdTe2	WTe2	0.196	0.225
HfCl2	MoS2	0.3	0.407
HfCl2	WTe2	0.315	0.427
HfCl2	GeS2	0.181	0.407
HfBr2	MoS2	-0.217	0.289
HfBr2	WTe2	-0.211	0.28
HfBr2	GeS2	-0.162	0.289
HfI2	MoS2	0.999	0.267
HfI2	WTe2	0.758	0.258
HfI2	GeS2	0.57	0.267
HfS2	MoS2	0.622	0.826
HfS2	CdI2	0.967	0.826
HfS2	WTe2	0.571	0.822
HfS2	GeS2	0.464	0.847
HfSe2	MoS2	0.668	0.479
HfSe2	CdI2	1.171	0.479
HfSe2	WTe2	0.633	0.497
HfSe2	GeS2	0.473	0.5
HfTe2	MoS2	0.575	0.244

Material	**Prototype**	**ohc_predict**	**gap_predict**
HfTe2	WTe2	0.57	0.261
HfTe2	GeS2	0.509	0.265
WS2	MoS2	2.025	0.179
WS2	WTe2	1.768	0.187
WSe2	MoS2	1.754	0.177
WSe2	WTe2	1.575	0.172
WSe2	GeS2	1.463	0.177
WTe2	MoS2	1.673	0.118
WTe2	GeS2	1.571	0.118
OsF2	CdI2	0.507	0.297
OsCl2	MoS2	0.731	0.366
OsCl2	CdI2	0.485	0.573
OsCl2	WTe2	0.647	0.591
OsCl2	GeS2	0.365	0.366
OsBr2	CdI2	0.552	0.412
OsBr2	GeS2	0.439	0.205
OsI2	MoS2	0.905	0.208
OsI2	CdI2	0.737	0.415
OsI2	GeS2	0.582	0.208
OsS2	MoS2	1.382	0.062
OsS2	WTe2	1.026	0.157
OsS2	GeS2	1.038	0.062
OsSe2	MoS2	1.57	0.066
OsSe2	WTe2	1.059	0.147
OsSe2	GeS2	1.261	0.066
OsTe2	MoS2	1.611	0.046
OsTe2	WTe2	1.075	0.126
OsTe2	GeS2	1.413	0.046
PtS2	CdI2	1.016	0.579
PtS2	WTe2	1.02	0.607
PtS2	GeS2	0.913	0.522
PtSe2	CdI2	1.134	0.487
PtSe2	WTe2	1.098	0.514
PtSe2	GeS2	1.013	0.43
PtTe2	CdI2	1.699	0.265
PtTe2	GeS2	1.567	0.208