Prediction of Thermoelectric Figure of Merit (ZT) using Machine Learning Techniques

Morsheda Parvin – 30072702

September 08, 2023

University of South Wales
Prifysgol De Cymru

FACULTY OF COMPUTING, ENGINEERING & SCIENCE
SCHOOL OF COMPUTING & MATHEMATICS

STATEMENT OF ORIGINALITY

This is to certify that, except where specific reference is made, the work described in this project is the result of the investigation carried out by the student, and that neither this project nor any part of it has been presented, or is currently being submitted in candidature for any award other than in part for the degree of MSc Data Science of the University of South Wales.

Signed Morsheda Parvin..... (student)

Date 08/09/2023.....

Acknowledgments

Embarking on this Master's project has not only marked a significant milestone in my academic journey but also deepened my understanding and expertise in the expansive field of Machine Learning within Data Science. I am deeply indebted to my project mentor, Dr. Samuel A. Jobbins, whose guidance has been a beacon of light throughout this project.

Sam, I cannot express enough how your consistent support, motivation, and mentorship have played a pivotal role in shaping our Machine Learning project. The enjoyment and enlightenment I found in this year's Machine Learning module were unparalleled. Through one of the assignments, I got the chance to delve into the practical applications of Machine Learning systems in analyzing real-world data, a venture that was further deepened in this project. Our weekly discussions over the recent months have been nothing short of insightful, steering the project's progression significantly. Your invaluable guidance and feedback have been the cornerstone of this project, a role that I am profoundly grateful for. I will miss working on this project as I have enjoyed it so much and I wish we had more time on this project.

Moreover, I would like to acknowledge my peers and fellow researchers, whose camaraderie and insightful discussions through Padlet enriched this journey manifold. Your collective passion and curiosity have fueled my drive, making this project not only a scholarly pursuit but a fulfilling and collaborative experience.

Furthermore, I extend my heartfelt appreciation to the team of academic staff in the Data Science department. I enjoyed my postgraduate studies so much and I really look forward to apply these Data Science skills and techniques in my career after University.

Abstract

In recent years, the focus on renewable energy solutions has intensified, with an emphasis on studying thermoelectric materials and their dimensionless figure of merit, ZT. These materials are promising in the conversion of waste heat to usable electricity, contributing to sustainable energy advances. This project utilized machine learning models including Random Forest, Decision Tree Regression, Support Vector Regression, and XGBoost to accurately predict the ZT values of various complex chemical materials, setting a solid foundation for further nuanced research in this field. The initial findings indicate that the Random Forest model excels in prediction accuracy and stability, followed closely by the Decision Tree regression model, while the SVR maintains considerable predictive power.

Moreover, the study employed principal component analysis (PCA) to glean deeper insights into the underlying data patterns, particularly highlighting the significant positive correlations found in the first two principal components, notably with MagpieData minimum number in the case of PC1. The project also employed the K-Means algorithm to cluster data based on the first 10 principal components, indicating meaningful cluster formations with a supportive average silhouette score of 0.57 at k=3.

As we progress, the strategy is to explore 111 elements previously left unstudied, anticipating collaborations with domain experts to augment the prediction accuracy beyond the present 88% R^2 value. Further, an exploration into the renewable and sustainable energy facets of these elements is projected, aiming to foster advancements in green energy solutions.

In conclusion, this project serves as a vital step in using machine learning to predict ZT values, potentially spearheading innovations in the renewable energy domain.

Contents

List of Figures

List of Tables

1 Chapter 1 - Introduction

The pursuit of efficient and sustainable energy sources has been a critical challenge in these days, driven by the growing concerns over environmental impact and the limited availability of traditional fossil fuels. Among various alternative energy sources, thermoelectric materials have gained significant attention due to their ability to directly convert waste heat into electricity, offering a promising pathway for enhancing energy efficiency and reducing greenhouse gas emissions.

In the recent past, the effectiveness of a thermoelectric material is quantified by its dimensionless figure of merit (ZT), a metric influenced by key factors such as the Seebeck coefficient, electrical conductivity, and thermal conductivity. While traditional computational techniques like density functional theory (DFT) offer accurate predictions, they are often computationally intensive and time-consuming, limiting their application in high-throughput material discovery[1]. In response, machine learning algorithms have emerged as a promising alternative, providing rapid and precise predictions of material properties. Utilising the strength of data-driven models, researchers may effectively explore a broad chemical and structural design space, making predictions and directing experimental efforts.

An auto-generated thermoelectric-materials database is presented, containing 22,805 data records, automatically generated from the scientific literature, spanning 10,641 unique extracted chemical names. Each record contains a chemical entity and one of the seminal thermoelectric properties: thermoelectric figure of merit, ZT; thermal conductivity, κ; Seebeck coefficient, S; electrical conductivity, σ; power factor, PF; each linked to their corresponding recorded temperature, T. The database was auto-generated using the automatic sentence-parsing capabilities of the chemistry-aware, natural language processing toolkit, ChemDataExtractor 2.0, adapted for application in the thermoelectric-materials domain, following a rule-based sentence-simplification step. Data were mined from the text of 60,843 scientific papers that were sourced from three scientific publishers: Elsevier, the Royal Society of Chemistry, and Springer. To the best of our knowledge, this is the first automatically generated database of thermoelectric materials and their properties from existing literature. The database was evaluated to have a precision of 82.25% and has been made publicly available to

facilitate the application of data science in the thermoelectric-materials domain, for analysis, design, and prediction. We interested only ZT data that filtered from 22805 data records.

This project seeks to leverage Machine Learning methodologies to predict the thermoelectric figure of merit (ZT) for different materials based on their intrinsic properties. By training models on a curated dataset of known thermoelectric materials and their corresponding ZT values, the project aims to develop accurate predictive models that can guide the discovery of new materials with enhanced thermoelectric efficiency.

Chapter 2 looks at related work to this project, including overviews of papers, studies and projects that have applied Machine Learning systems to ZT data. Chapter 3 focuses on the preparation of the data for analysis, involving cleaning, standardization, crystal structure analysis of chemical elements, and an initial exploration to understand its characteristics and potential correlations. Chapter 4 reviews different machine learning systems that have previously been utilized for analyzing ZT data, evaluating their effectiveness and methodologies. Chapter 5 details the application of supervised learning techniques to the ZT data, involving the training of models to predict or classify materials based on their ZT values and evaluating these models' performance using various metrics. Chapter 6 embarks on an unsupervised learning approach to analyze the ZT data further, utilizing techniques like clustering and dimensionality reduction to uncover underlying patterns and trends in the data. We conducted all of our Machine Learning experiments in a Python environment. It was a great choice for our project because we were able to handle our machine learning methods, data processing, and handling enormous amounts of data using rich libraries like scikit-learn, scipy, and pandas.

Overall, this project endeavors to utilize machine learning techniques to foster a deeper understanding and identify optimization strategies for thermoelectric materials, with a primary focus on analyzing ZT data. It aims to bridge the gap between materials science and machine learning, introducing a novel framework that accelerates the

discovery and enhancement of thermoelectric materials with improved ZT values. Through the synergistic integration of data analytics and materials research, this initiative is poised to significantly contribute to the progression of sustainable energy solutions and the broader sphere of materials informatics.

2 Chapter 2 – Related Work

Thermoelectric materials have the remarkable ability to transform heat into electricity, offering a sustainable solution to the pressing global energy demands and environmental concerns. These devices proficiently capture waste heat generated by various electrical and mechanical activities, presenting a solid-state, scalable, and durable solution for energy harvesting. Being lightweight and reliable, thermoelectric devices pave the path towards a more sustainable future by optimizing energy usage and reducing environmental impact.

The effectiveness of a thermoelectric material at converting heat into electricity, and vice versa, is determined by the thermoelectric figure of merit, denoted as ZT[2].The value is established by Eq (1), in which S signifies the Seebeck coefficient, σ represents the electrical conductivity, κ stands for the thermal conductivity, and *T* is the temperature of the material.

$$ZT = \frac{S^2\sigma}{\kappa}T \tag{1}$$

[ZT = \frac{S^2\cdot\sigma\cdotT}{\kappa}]

To achieve a high thermoelectric figure of merit, a material should combine a high Seebeck coefficient and/or electrical conductivity, coupled with low thermal conductivity. This is a difficult task due to the dependency of these parameters, and a number of techniques, including doping and nano structuring, are being used to improve the thermoelectric performance of materials[3]. The field attracts significant interest, with an increasing number of articles being published in recent years.

The first evidences of the direct conversion between thermal and electrical energy date back to the end of the XVIII century. In those years, the Italian physician, physicist,

biologist and philosopher Luigi Aloisio Galvani (Bologna, September 9, 1737 - Bologna, December 4, 1798), who was conducting experiments at the university of Bologna on the effect of electricity on animals, was about to discover what is now given the name of galvanism. In the late 1780s, Galvani accidentally observed. Heeger and co-workers reported a high conductivity for iodine doped polyacetylene in 1977 [564], organic semiconductors have attracted a great deal of attention, as they combine (semi)conducting properties with the attributes of soft matter, which paves the way to the fabrication of flexible, light and bio-compatible electronic devices via cost-efficient processing from melt or solution.

By the 20th century, the industry had matured, offering reliable solutions for niche applications such as space missions and cardiac pacemakers. The debate between Galvani and Volta, at the end of the XVIII century, on the nature of electrical phenomena in animals, ultimately led to Volta's conclusion that the electromotive force producing dead frogs muscles response originated from the temperature difference between the junctions of two dissimilar conducting materials in contact with them: it was at this point in time that he unveiled the thermoelectric effect, although Volta's name is not directly linked today to any thermoelectric

Traditionally, the discovery of novel materials was guided by the intuition of researchers and trial-and-error process. However, recent developments in data science and the creation of commercial machine-learning algorithms have the potential to accelerate the identification of materials[4]. As the name suggests, for data-driven methods to be effective, a large dataset of relevant information is required.

Gaultois et al. undertook the first data-driven evaluation of thermoelectric materials in the field of thermoelectric based on a manual effort, cataloguing the five thermoelectric values listed above, for various temperatures, and for various families of thermoelectric materials[5]. Information extracted from 100 articles resulted in a database with 1,100 data records, essentially pioneering data-driven analysis in the thermoelectric-materials domain. In a subsequent study[6], these data were combined with ab-initio calculated electronic-structure properties from Hautier et al.[7] and Pymatgen[8], to train

a machine-learning system, which operates as an online recommendation engine (http://thermoelectrics.citrination.com). 25,000 computationally generated compounds were screened, and a novel compound, “far” from the chemical space of familiar thermoelectric compounds, was suggested as a promising, yet unstudied, thermoelectric material. Ab-initio Density Functional Theory (DFT) calculations are a typical source of data for thermoelectric applications[9], allowing the computation of values for different thermoelectric properties spanning thousands of materials. To rank thousands of compounds based on their calculated thermoelectric properties, Gorai et al. created the Thermoelectric Design Lab[10], an online tool. They used a custom predictor[11] that has been shown to accurately predict Z, at peak ZT, to within about a factor of two. According to the website, "the database contains calculated transport properties and rankings of 2701 materials for thermoelectric performance."

Recently, Tshitoyan et al.[12] trained word embeddings on abstracts from the field of materials science to find chemicals that are likely to fall within the category of thermoelectric materials but had not previously been investigated as potential candidates in isolation.

The prospect of a sizable and trustworthy thermoelectric-materials database is frequently discussed by authors of such studies. Ab-initio computationally produced databases have the capacity to generate enormous amounts of data, but they are plagued by the requirement to use approximations, which calls into question their dependability. However, building manually curated databases takes a lot of time and requires the expertise of subject matter experts.

The use of text mining to automatically extract data from the scientific literature can combine the accuracy of experimental data, with the high-throughput capacity offered by computational means. This project describes the automatic data extraction from a large corpus of relevant research papers using the chemistry-aware natural language processing toolkit, ChemDataExtractor 2.0, based on software developments by Cole and co-workers[13,14], to create a thermoelectric-materials database.

This is the first automated database of thermoelectric materials, created through academic research and ChemDataExtractor enhancements. The database aids in

identifying thermoelectric materials and serves as a user look-up table for scientists to discover specific properties.

Chapter 3 - Data Pre-Processing

In this chapter, we introduced the data pre-processing procedures. We begin with a description of the thermoelectric databases, then we discussed the process of cleaning and filtering the data for supervised and unsupervised learning. Additionally, Exploratory Data Analysis conducted to better understand the data. The adapted thermoelectric-centred version of ChemDataExtractor is available at

https://github.com/odysie/thermoelectricsdb/tree/main/chemdataextractor_thermoelectrics.

The full database is publicly available for download from *Figshare*[15] in three standard formats: CSV, JSON, and MongoDB. We work on only main_tedb.csv.

3.1 Dataset Description

The original database contains 22,805 rows and 24 columns. But we filtered the original dataset to only include rows where the column 'Model' has the value 'ZT'. So, filtered database consists of 11,668 entries or rows. Each row represented a separate entry or data point, and each column represented a different attribute of that entry. The columns included various pieces of information such as names, labels, values, units, temperatures, authors, publication details, and more. The DataFrame seems to be a comprehensive collection of data related to a scientific context, possibly pertaining to thermoelectric materials or another domain.

The Name column, which is a crucial categorical variable, represents the materials through their chemical formulae, indicating the type and proportion of elements within each compound. They varied from simple binary compounds like SnSe to more complex formulations like In0.1Yb0.2Co4Sb12 which indicate a mixture of elements in

specific ratios. The numerical subscript in the formulae represents the stoichiometry or the atomic proportion of each element in the compound.

Complex Formulations:

In0.1Yb0.2Co4Sb12: This suggests that the material is an alloy or compound of Indium (In), Ytterbium (Yb), Cobalt (Co), and Antimony (Sb) with their respective atomic proportions.

Simple Compounds:

YNiSb: This indicates a compound of Yttrium (Y), Nickel (Ni), and Antimony (Sb).

Bi2Te3: Bismuth Telluride, a well-known thermoelectric material.

SnSe: Tin Selenide, another common compound.

Mg3Sb2: Magnesium Antimonide.

BiSe: Bismuth Selenide

ZnO is a known compound, Zinc Oxide

Elemental Variations:

MnSi1.75: This indicates a compound of Manganese (Mn) and Silicon (Si) where the Si to Mn ratio is slightly adjusted.

Unnamed: 0: Typically, this represents a default index added by Pandas when reading a dataset. Depending on its nature, it could be Numerical or a simple Row Identifier.

Specifier: This field encompasses essential textual information imperative for the extraction of each property. Notably, it might also include prefixes that suggest the extreme nature or an average of a property, such as 'peak', 'max/min', 'highest/lowest', 'mean/average', etc. Additionally, it can contain information about the semiconductor type of the compound.

Model: This differentiates between the five crucial properties: thermoelectric figure of merit, thermal conductivity, Seebeck coefficient, electrical conductivity, and power factor. We filtered only ZT data for prediction.

Model_Type: Especially relevant for thermal conductivity and electrical conductivity, this field provides further distinctions, differentiating between 'lattice', 'electronic', and 'total' thermal conductivity and between 'electronic' and 'ionic' electrical conductivity.

Label: It indicates the value of a variable composition label found in the compound.

Value & Units: While this represents measurements and seems numerical, it's stored as an object type, which means it could contain strings and might need preprocessing.

Temperature_Value & Temperature_Units: Every data record is associated with these fields, normalized to Kelvin. When representing a range, the Value and Temperature_Value fields consist of two components indicating the minimum and maximum value of that range. Averages for these values are stored in the Temperature_Average and Value_Average columns.

Editing: Contains details about the material that aren't part of the chemical name, like doping.

Pressure: This field captures the pressure conditions of the study and consists of both a numerical value and its unit.

Process: Contains specifics about the process related to the data record.

Direction_of_Measurement: Provides insights into the orientation of the property Authors: Authors of the entry.

Resolution: A crucial field that resolves combinations of CEMs into 'host' and 'dopant', while distinguishing the methods of combination. It uses a python dictionary to store this information. An example provided elucidates its structure - '10 at. % Nb doped TiO2' is resolved to {'host': 'TiO2', 'dopant': 'Nb', 'amount': '10 at. %', 'method': 'doping'}.

DOI, Title, Access_Type, Publisher, Journal, Authors, and Publication_Year: These fields collectively provide exhaustive bibliographic information about the source of the data record.

3.2 Data Cleaning

Out of the entire dataset comprising 11,668 instances and 24 features, we undertook an in-depth analysis to determine the completeness of the data. We found noticeable discrepancies in the form of missing values in several columns. Specifically, columns Unnamed: 0', 'Temperature_Units', 'Model', 'Label', 'Editing', 'Model_Type', 'Authors', 'Process', 'Specifier', 'Journal', 'Units', 'Pressure', 'Publication_Year', 'Direction_of_Measurement', 'Resolution', 'DOI', 'Title', 'Access_Type', and 'Publisher' showed significant gaps. To avoid potential biases and inaccuracies, we discarded these columns. Last but not the least, we employed the ast.literal_eval function for Value and Temperature_Value column inorder to evaluate a string containing a Python expression and returns its actual Python data structure. It extracted the first element from the list, and stored the extracted values in new columns like 'numerical_Value' and 'temperature_value'. And hence, we dropped Value and Temperature_Value columns. After cleaning we had 11668 instances and 5 features that was still an excellent data for further processing or analysis of the data.

3.3 The Thermoelectric Figure of merit subset Dataset

In this new subset of thermoelectric figure of merit data with 11668 data points and 5 features, numerical_Value is the target variable for supervised learning. For this data, unsupervised learning aims to uncover underlying structures and relationships in data without a clear target variable, such as clustering materials based on thermoelectric properties or reducing dimensionality.

3.4 Exploratory Data Analysis

We now investigate the thermoelectric figure of merit data and summarize key insights. The distribution numerical_Value, Value_Average as depicted by the histogram, are bimodal, signifying that there are two distinct groups or peaks within the data. Temperature_value and Temperature _Average as depicted by the histogram, are

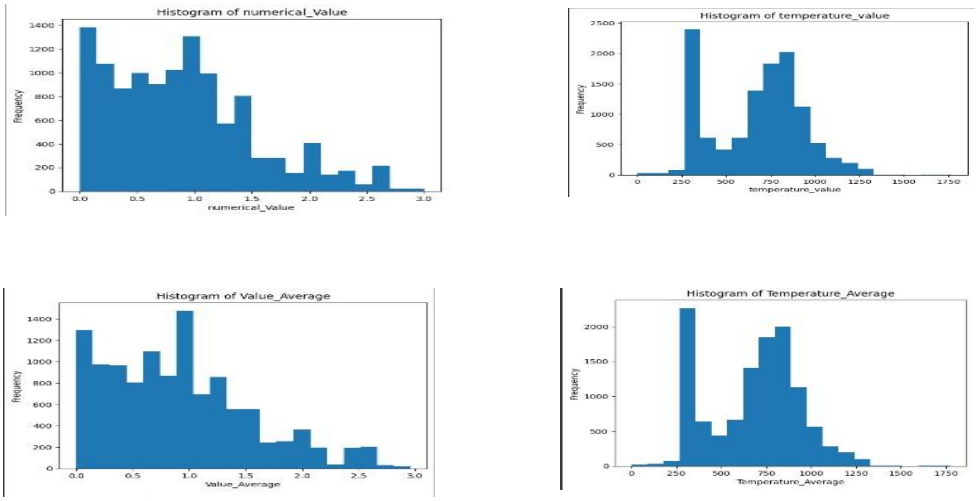


Figure 1: Histograms of numerical_Value, temperature_value, Value_Average and Temperature_Average

unimodal. There was no outliers that seemed to vastly affect the distribution.

Visulisation for model column by Pie charts and correlation heatmap for 5 features

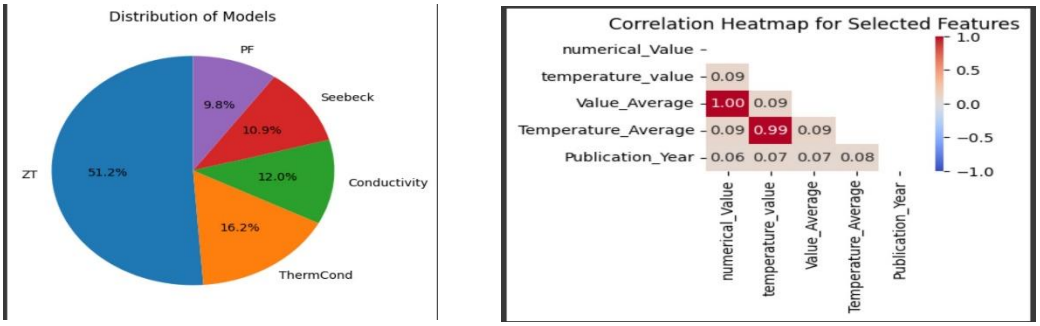


Figure 2: Pie charts for visualize Model and heatmap for correlation

From the pie chart, it can be observed that (before filtering) a significant focus by 51.2% has been placed on capturing or studying the ZT model in comparison to the others. From the heatmap, it can be identified that strong positive relationships between numerical_Value and Value_Average, temperature_value and Temperature_Average but we did not eliminate the temperature_value and Temperature_Average highly-correlated features as there was a risk of removing too much important information. But we dropped Value_Average. Besides, by removing features we would only have 4 (after dropping Publication_Year) features and 11668 rows that would be interesting to compare supervised and unsupervised learning model performances on data for future work.

3.5 Further preprocessing for prediction

First and foremost, we would like to remove non-chemical words or descriptors from the 'Name' column of the dataframe.

non_chemical_words=['single-crystal','n-type','p_type','Afree','Misfit','Aluminum', 'Bulk','Chlorine','Polyline','Pure','Snn','Zfree','Telluride','Zcontaining','Tin','n-','Ac','Py','Camphor','Sex','Alxx','Doped','Sny','Bismuth','Cubic'…….]

```
import re

import pandas as pd

# Join the non-chemical words list to form a regex pattern

pattern = '|'.join(map(re.escape, non_chemical_words))

# Remove the words from the 'Name' column using the pattern

df1['Name']=df1['Name'].str.replace(pattern,'',regex=True).str.strip()

# Display the cleaned 'Name' column

print(df1['Name'])
```

The code imports the re module for regular expressions and pandas for data manipulation. It has a list of non-chemical words that are intended to be removed from the 'Name' column of a dataframe. To facilitate this removal, the list of words converted

into a single regex pattern. The re.escape() function ensures that any special characters in the words treated literally. Using map() and ' | '.join(), a regex pattern created to match any word from the list. In the dataframe df1, any occurrence of these words in the 'Name' column replaced with an empty string, and any leading or trailing spaces then removed.

After this we replaced outliers in its numerical columns with specific lower and upper bounds calculated using the Interquartile Range (IQR) method. It ensures that data points significantly different from others are adjusted to these bounds, mitigating their potential distortive effects on further analyses, without removing any data points from the dataset. After the operation, the cleaned data stored in a new DataFrame named df_iqr_cleaned. Finally, we got 11668 rows and 4 columns from the original data frame.

3.6 Crystal Structure Analysis

We picked an element BiSe(Bismuth Selenide) is a well-studied topological insulator and a prominent thermoelectric material, known for its unique electronic properties and potential applications in various fields such as electronics, thermoelectric devices, and spintronics.

Chemical Formula: Bi2Se3

Crystal Structure: Rhombohedral, characterized by quintuple layers (Se-Bi-Se-Bi-Se) that are weakly bonded to each other, which facilitates its two-dimensional electronic characteristics.

Thermoelectric Performance: The thermoelectric properties of Bi2Se3 are notable, especially at higher temperatures, where it exhibits a high thermoelectric efficiency, quantified by the dimensionless figure of merit (ZT).

Applications

Thermoelectric Devices: It can be used in thermoelectric generators and coolers due to its high thermoelectric performance.

Topological Insulator: As a topological insulator, it has surface states that are protected by time-reversal symmetry, offering new opportunities in quantum computing and spintronics.

Photonic Devices: Its unique electronic properties have also made it a material of interest in photonic devices.

Research and Development

Bi2Se3 is extensively researched to enhance its thermoelectric performance further through various strategies, including doping with other elements, nanostructuring, and the development of composite materials. Its role as a topological insulator is also a thriving field of research, aiming to harness its potential for futuristic technologies.

We applied code for obtaining its structure .For this both the Pymatgen (Python Materials Genomics) and Materials Project API are valuable tools in the field of materials science, offering computational and data resources to accelerate the discovery of new materials. Pymatgen is an open-source Python library for advanced materials analysis, while the Materials Project API provides an interface for querying and accessing the vast dataset of tens of thousands of materials. After creating the API key from the Materials Project, we can access their vast database of computed materials properties. The API key allows everyone to programmatically query the database and fetch specific information about materials, such as their crystal structures, band structures, density of states, and other relevant properties. BiSe crytallize in the tetragonal 14/mmm space group. The structure is two dimensional and consists of two BiSe sheets oriented in the (0, 0, 1) direction[16]. Bi^{2+} is bonded to five equivalent Se^{2-} atoms to form a mixture of corner and edge-sharing $BiSe_5$ square pyramids. There is one shorter (2.95 Å) and four longer (3.02 Å) Bi-Se bond lengths. Se^{2-} is bonded to five equivalent Bi^{2+} atoms to form a mixture of distorted corner and edge-sharing $SeBi_5$ square pyramids. Here's a brief guide on how we could use the API key with Pymatgen to get information about the BiSe crystal structure

```
# To access the Materials Project API through pymatgen

from pymatgen.ext.matproj import MPRester

# To use the Materials Project API directly
```

```
from mp_api.client import MPRester

api_key = 'YOUR_NEW_API_KEY'

# Initialize MPRester with the new API key

mpr = MPRester(api_key)

structures = mpr.get_structures("BiSe")

BiSe = structures[0]

BiSe
```

Structure Summary

Lattice

abc : 4.177454152317294 4.177454152317294 18.110445204189126

angles : 87.23580513849087 87.23580513849087 89.81963080579416

volume : 315.31246899082106

A : 2.949253 2.958552 0.0

B : -2.949253 2.958552 0.0

C : 0.0 1.233216 18.068409

pbc : True True True

PeriodicSite: Bi (0.0, 4.992, 10.77) [0.7195, 0.7195, 0.5962]

PeriodicSite: Bi (0.0, 2.158, 7.297) [0.2805, 0.2805, 0.4038]

PeriodicSite: Se (0.0, 2.128, 10.31) [0.2407, 0.2407, 0.5706]

PeriodicSite: Se (0.0, 5.022, 7.759) [0.7593, 0.7593, 0.4294]

abc: The lengths of the lattice vectors a, b, and c.

angles: The angles between the lattice vectors. Here, the lattice angles are close to right angles, which suggests an orthorhombic or tetragonal structure, but slight deviations are noted.

volume: The volume of the unit cell.

A, B, C: The three lattice vectors A, B, and C. These vectors define the unit cell in 3D space.

pbc: Indicates that the system is periodic in all three dimensions.

PeriodicSite: This section lists the atomic sites in the unit cell. For each site, the type of atom (Bi or Se), its Cartesian coordinates, and its fractional coordinates within the unit cell are given

```
BiSe.composition

BiSe.density

BiSe.distance_matrix

BiSe.get_space_group_info()
```

Composition ('Bi2 Se2')

BiSe.density : 3.0327775148425826

The value 3.0327775148425826 seems to be the computed density of Bi2Se2 (or BiSe) in g/cm^3. This means the crystal structure of Bi2Se2 has a density of approximately 3.03 g/cm3. For comparison, this value is close to the experimental and computed densities available in literature for Bismuth Selenide. Bismuth Selenide is an interesting compound because of its topological insulator properties. Knowing its density is important for various applications and experimental measurements.

The distance matrix represents the distances between each atom in BiSe structure. The matrix is symmetric, as the distance from atom A to atom B is the same as the distance from atom B to atom A. Here's a breakdown of the matrix:

array([[0. , 4.4848264 , 2.90147483, 3.0133613],

[4.4848264 , 0. , 3.0133613 , 2.90147483],

[2.90147483, 3.0133613 , 0. , 3.85799653],

[3.0133613 , 2.90147483, 3.85799653, 0.]])

The diagonal elements are all zeros, indicating that the distance of an atom to itself is zero. The element at position [0][1] (or [1][0]) is 4.4848264, meaning the distance between the first and the second atom is 4.4848264 Ångströms (or in whatever unit you're working in, most likely Ångströms in crystallography). Similarly, the element at position [0][2] is 2.90147483, meaning the distance between the first and the third atom

is 2.90147483 Ångströms, and so on for the rest of the matrix. This matrix can be used to infer atom positions, check bond lengths, and perform other analyses.[17]

BiSe.get_space_group_info()

('C2/m', 12)C2/m: This is a space group symbol in Hermann-Mauguin notation. The C2/m space group belongs to the monoclinic crystal system. The symbol provides information on the symmetry elements present in the structure:

C indicates a unique axis (the c-axis in monoclinic systems) with a center of inversion.

2 indicates a 2-fold rotation axis. /m indicates a perpendicular mirror plane.

12: This is the space group number. Each space group is assigned a unique number between 1 and 230 in the International Tables for Crystallography. The number 12 corresponds to the C2/m space group.[18]

Knowing the space group of a crystal is essential as it provides comprehensive information about the symmetry elements of the crystal, which in turn dictates its atomic arrangement and overall structure.

```
# We took mp-570286 from materials ids

mpr.get_materials_ids('BiSe')

mpr.get_structure_by_material_id('mp-570286')
```

```
Structure Summary

Lattice

abc   :  13.047544596010622  13.047544596010622  13.047544596010622
angles : 161.31842212098357 161.31842212098357 26.539488344796343
volume : 227.80542644763926                                      A
: -2.117699 2.117699 12.699178                                   B
:             2.117699            -2.117699             12.699178
C         :         2.117699          2.117699          -12.699178
pbc             :             True             True             True
PeriodicSite:  Bi   (0.0,   0.0,   17.37)   [0.6841,   0.6841,   0.0]
```

```
PeriodicSite: Bi (0.0, 0.0, 8.024) [0.3159, 0.3159, 0.0]
PeriodicSite: Se (0.0, 0.0, 20.33) [0.8004, 0.8004, 0.0]
PeriodicSite: Se (0.0, 0.0, 5.07) [0.1996, 0.1996, 0.0]
```

The structure represents a crystallographic structure with details about the lattice parameters and atomic positions. Here's a breakdown of the information:

Lattice Parameters:

Lattice Vectors (abc): The three vectors (in Angstroms) representing the three dimensions of the lattice. The given structure has approximately equal lattice vectors, indicating that it is close to cubic in nature.

Angles: The angles (in degrees) between the lattice vectors. They tell us about the geometry of the unit cell.

Lattice Vectors (A, B, C): These are the Cartesian coordinates of the lattice vectors. Each vector is represented by the x, y, and z components.

Atomic Positions:These positions are given in both Cartesian coordinates and fractional coordinates with respect to the lattice vectors:

Bi Atom 1: Located at (0.0, 0.0, 17.37) in Cartesian coordinates or at [0.6841, 0.6841, 0.0] in fractional coordinates.

Bi Atom 2: Located at (0.0, 0.0, 8.024) in Cartesian coordinates or at [0.3159, 0.3159, 0.0] in fractional coordinates.

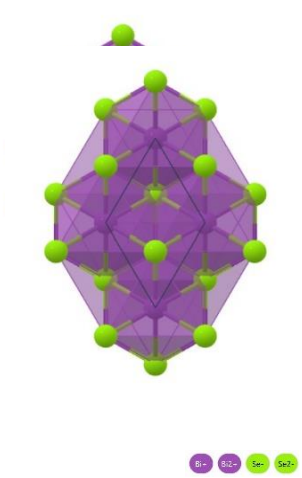
Se Atom 1: Located at (0.0, 0.0, 20.33) in Cartesian coordinates or at [0.8004, 0.8004, 0.0] in fractional coordinates.

Se Atom 2: Located at (0.0, 0.0, 5.07) in Cartesian coordinates or at [0.1996, 0.1996, 0.0] in fractional coordinates.

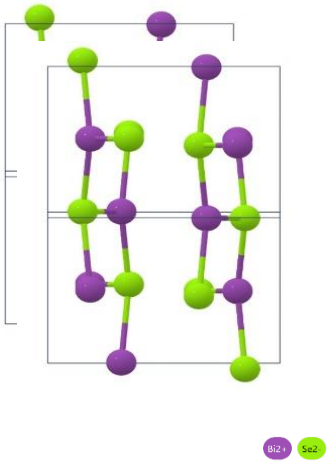
This structure provides all the information required to understand the geometric arrangement of the atoms in the unit cell and the overall crystallographic details of the material. It can be used to perform various materials characterization, electronic structure calculations, or other computational materials science tasks.

```
band_structure = mpr.get_bandstructure_by_material_id('mp-570286')

from pymatgen.electronic_structure.plotter import BSPlotter
```

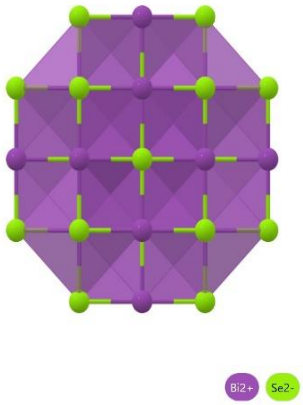
```
plot = BSPlotter(band_structure)

plot.get_plot()
```



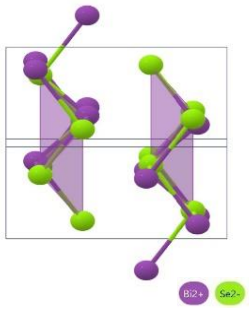
mp-27902, trigonal



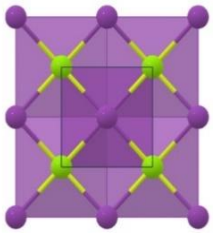
mp-1182022, monoclinic



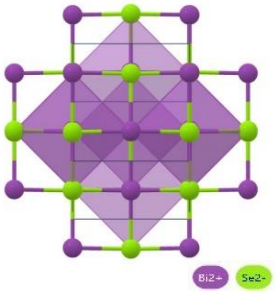
mp-568844, cubic



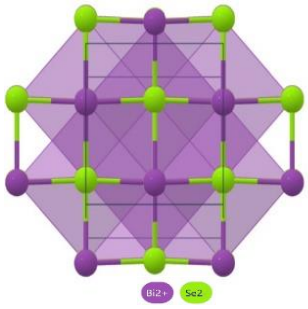
mp-1078607, monoclinic



mp-570286, tetragonal

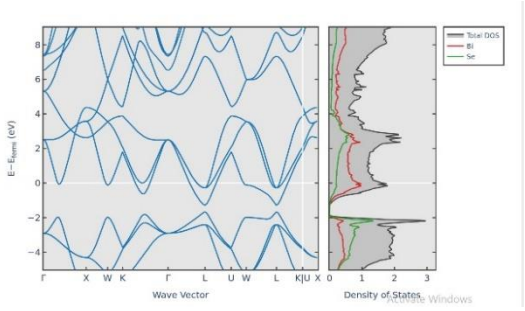


mp-1063764, monoclinic

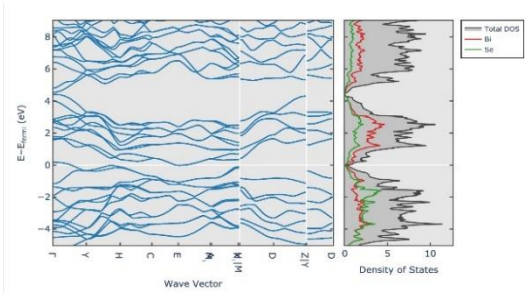


mp-1063755, monoclinic

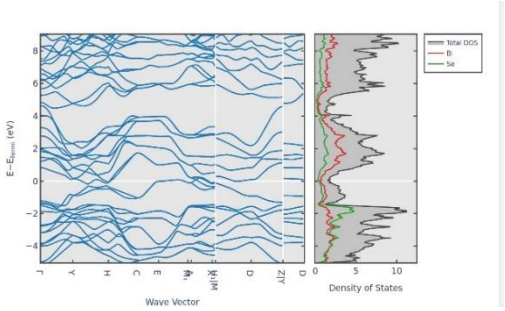
Figure3: Different Crystal Structure of BiSe



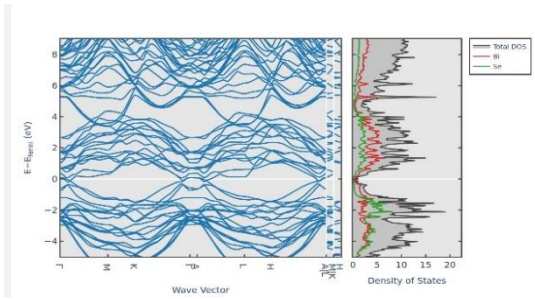
(a)



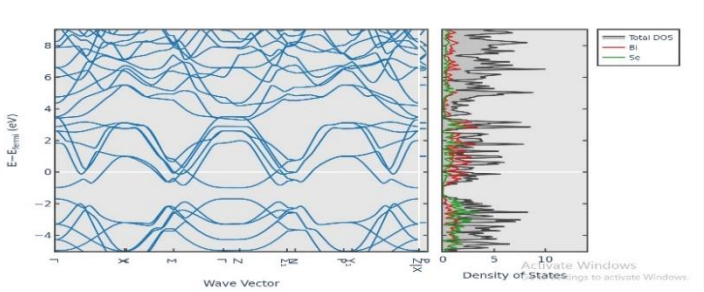
(b)



(c)



(d)



(e)

Figure 4: (a), (b), (c), (d), (e) are wave vectors for mp-27902, mp-1182022, mp-568844, mp-1078607and mp-570286

```
band_structure.get_band_gap()
```

```
band_structure.get_direct_band_gap()
```

The term "band gap", refers to the energy difference between the top of the valence band and the bottom of the conduction band in insulators and semiconductors which is 0.0 here for BiSe. It's the energy required to move an electron from the valence band to the conduction band.

3.7 Feature Scaling

Feature scaling was one of the most important transformations to apply to thermoelectric figure of merit data. Several numerical features in the dataset were of different scales and took different units of measurements, therefore we would not expect Machine Learning algorithms to perform well with this kind of data. Features were therefore transformed to comparable scales using standardisation. This prevented features with wider ranges dominating those with smaller ranges. The Z-score is a popular method to standardise data. This Z-score was implemented on this data by subtracting the mean and dividing the the standard deviation for each value of each feature. The Z-score is given by

$$Z = \frac{\text{mean\_value}}{\text{standard deviation}}$$

Once standardised, the data had a mean of zero and a standard deviation of one, and therefore of the same scale.

3.8 Feature Engineering

Feature engineering is the process of transforming raw data into meaningful features that can be used to improve the performance of machine learning models. Feature generation is a subset of feature engineering, and it specifically refers to the process of creating new features from the existing data. These new features can be combinations of existing ones, or they can be derived from domain-specific knowledge.

In the context of materials science and using tools like Matminer[26], Pymatgen and Pydantic, feature generation is particularly crucial because of the unique nature and complexity of materials data.

Specifically, from pymatgen.core import Composition:

The Composition class provides methods for handling and analyzing chemical compositions. It enables users to extract information about a material based on its chemical formula, like the atomic fractions of each element.

Matminer:

Purpose: A Python library for data mining the properties of materials.

Key Features: 1. Integrates with Pymatgen to provide tools to turn complex materials data (like crystal structures) into numerical or categorical descriptors (features) for machine learning.

2. Offers a wide range of featurizers that can turn materials data into a format suitable for machine learning models.

3. It also contains utilities for data retrieval, making it easier to access datasets from various sources.

Specifically, from matminer.featurizers.composition import ElementProperty:

The ElementProperty is a featurizer within Matminer that provides numerical descriptors based on the properties of elements present in a material. This helps in translating something like a chemical formula into a format that can be understood by machine learning algorithms.

Pydantic:

Pydantic is not specifically related to materials science like Matminer and Pymatgen. Instead, it's a data validation and settings management tool for Python.

Purpose: It uses Python type annotations to validate data and convert (or "cast") input data to the appropriate type.

Key Features: 1. Provides a way to validate data to ensure it meets certain conditions before it's processed.

2. Helps catch potential errors early by ensuring data is in the expected format.

3. Often used in scenarios like web APIs, configuration management, and data parsing.

In summary, while Pymatgen and Matminer are closely related and used for materials informatics tasks, Pydantic is a more general-purpose tool for data validation in Python.

```
from pymatgen.core import Composition
from matminer.featurizers.composition import ElementProperty
import pandas as pd
from pymatgen.core.composition import Composition
from matminer.featurizers.composition import ElementProperty
# Function to split and clean weight percentage
def split_weight_percentage(s):
    parts = s.split('+')
main_comp = parts[0].strip() if parts else ''
wt_percent = parts[1].strip() if len(parts) > 1 else None
 return main_comp, wt_percent
# Function to further clean names for composition extraction
def clean_compositions(name):
    # Removing content within brackets
    name = re.sub(r'\(.*?\)', '', name)
      # Removing content after 'with'
    if 'with' in name:
        name = name.split('with')[0].strip()
      # Handling dashes and other separators
```

```
    name = name.split('-')[0].strip()  # Keeping only before the dash
    name = name.split('–')[0].strip()  # Keeping only before the en-dash
    # Removing percentages
    name = re.sub(r'%', '', name)
     # Avoid removing lowercase letters that can be part of element symbols (e.g., "Na")
    # Instead, let's remove potential words or descriptors
    for word in ['wt', 'co', 'of']:
        name = name.replace(word, '')
    return name.strip()
df_iqr_cleaned['Main_Composition'], df_iqr_cleaned['Weight_Percentage'] = zip(*df_iqr_cleaned['Name'].map(split_weight_percentage))
df_iqr_cleaned['Main_Composition'] = df_iqr_cleaned['Main_Composition'].apply(clean_compositions)
def get_composition(name):
    try:
        return Composition(name)  # Return Composition object
    except:
        return None
df_iqr_cleaned['composition'] = df_iqr_cleaned['Main_Composition'].apply(get_composition)
# Isolate problematic rows
invalid_compositions_df = df_iqr_cleaned[df_iqr_cleaned['composition'].isnull()]
```

The Composition class from the pymatgen library is used to convert chemical formula strings (assumed to be in the 'Name' column) into a standardized composition representation. The script uses the Python libraries pandas, re, and pymatgen to clean and process a dataset of chemical compositions, presumably stored in a

DataFrame named df_iqr_cleaned. The dataset contains a column called "Name" which holds the chemical compositions in various textual formats.

split_weight_percentage(s): Splits the input string s into two parts: the main composition and the weight percentage, based on the '+' delimiter. It returned a tuple with these values.

clean_compositions(name): Cleans the input string by removing content inside brackets, following the word 'with', percentages, and certain words. It also processed dashes and other separators to better isolate the chemical name.

get_composition(name): Tried to create a Composition object using the cleaned name string. If successful, returns the object; otherwise, returns None.

Initially, it created 'Main_Composition' and 'Weight_Percentage' columns in df_iqr_cleaned by applying the split_weight_percentage function to each entry in the 'Name' column. Subsequently, it further processed the 'Main_Composition' column using the clean_compositions function to remove undesirable text and descriptors. Then, it created a new column named 'composition' by applying the get_composition function to each entry in the 'Main_Composition' column, to generate chemical composition objects.

Isolation of Invalid Rows: The script identifies and isolates rows with undetermined composition (where the 'composition' column is null) into a separate data frame, invalid_compositions_df.

Output of Invalid Rows: Lastly, it outputs the rows where the composition could not be determined to identify entries (only 111) that might require additional cleaning or manual intervention. As it contains Ho 3H11001 : C 2 NaGdCl 6 57 Slfide element that has no information.

This script operates as a data cleaning pipeline to process chemical composition information, helping to isolate main compositions and weight percentages from each entry, clean text data, and derive chemical compositions using the pymatgen library. It facilitated the identification of entries with unclear or invalid compositions, aiding in

data validation and preparation for subsequent analytical steps, crucial in materials informatics or computational materials science. After running the code we got the following columns

```
[32] print(df_iqr_cleaned[['Name', 'composition']])
```

```
                              Name       composition
3701            In0.1Yb0.2Co4Sb12  (I, Yb, Co, Sb)
3702            In0.1Yb0.2Co4Sb12  (I, Yb, Co, Sb)
3703            In0.1Yb0.2Co4Sb12  (I, Yb, Co, Sb)
3704                         YNiSb       (Y, Ni, Sb)
3705                       Bi2Te3          (Bi, Te)
...                           ...               ...
15364        Na- PbTe-PbS ( 12 %              (Na)
15365  Pb0.60Sn0.40Te0.995I0.005       (Pb, S, Te)
15366  Pb0.60Sn0.40Te0.995I0.005       (Pb, S, Te)
15367  Pb0.60Sn0.40Te0.995I0.005       (Pb, S, Te)
15368  Pb0.60Sn0.40Te0.995I0.005       (Pb, S, Te)

[11668 rows x 2 columns]
```

Feature Extraction with Matminer:

Applying matminer's ElementProperty featurizer:

featurizer = ElementProperty.from_preset(preset_name="magpie")

df_features_final = featurizer.featurize_dataframe(df_no_outliers, col_id="composition", ignore_errors=True)

ElementProperty is a featurizer from the matminer library. A featurizer is a tool that can convert complex information (like the composition of a material) into numerical or categorical features that machine learning algorithms can work with. from_preset(preset_name="magpie") initializes the featurizer with a set of predefined properties. The "magpie" preset provides a comprehensive set of features derived from elemental properties[27]. Some of these features include average atomic mass, atomic radius, electronegativity, and many more. Ultimately, by using the "magpie" preset, we're asking the featurizer to extract a broad range of elemental features from the provided material compositions.

After this step, the DataFrame where the original "composition" column has been expanded into numerous new columns, each representing a specific elemental property or feature derived from the original compositions. This transformation makes the data suitable for input into machine learning algorithms, as these algorithms

typically require numerical input features such as for the composition "In0.1Yb0.2Co4Sb12":

Individual Element Properties: The featurizer would fetch properties for each of the elements (In, Yb, Co, Sb). These properties might include things like atomic number, atomic radius, electronegativity, and so on. If we have four elements and ten properties for each, that's already 40 features.

Statistics on Element Properties: For each property, the featurizer can compute statistics based on the relative amounts of the elements. For instance, it might compute a weighted mean atomic number, where the weights are based on the stoichiometry of each element. Other statistics might include the range, standard deviation, and other statistical measures. So if we have 10 properties and generated 5 statistical measures for each, that's another 50 features.

Compound-level Features: Beyond properties of individual elements, there might be features that describe the compound as a whole. For instance, the featurizer might compute a feature that indicates the overall ionic character of the compound, or perhaps features that describe the electron configuration of the compound as a whole.

Interactions Between Elements: Some features might be based on interactions between pairs or groups of elements. For example, the featurizer might compute a feature that describes the bond strength between the most and least electronegative elements in the compound.

The actual columns and features we got depend on the details of the featurizer and its settings. The goal, however, is to transform the original composition, which is a complex descriptor, into a set of numerical features that capture as much relevant information as possible about the material's composition. These features can then be used as input to machine learning models. The "MagpieData" columns in the dataframe result from featurization, turning elemental properties of a compound into statistics:

Statistical Descriptors: These include minimum, maximum, range, mean, avg_dev, and mode.

Property Indicators: Categories like Number (atomic number), GSmagmom (ground state magnetic moment), and SpaceGroupNumber (crystallographic space group) specify the kind of elemental property.

"MagpieData minimum Number" represents the smallest atomic number in the compound.

"MagpieData range Number" is the difference between the highest and lowest atomic numbers.

In (Indium): Atomic number = 49

Yb (Ytterbium): Atomic number = 70

Co (Cobalt): Atomic number = 27

Sb (Antimony): Atomic number = 51

To calculate the weighted average atomic number, we multiplied each atomic number by its respective fraction in the compound and sum these products.

Weighted Average Atomic Number = ∑(fraction of element×atomic number of element)

Weighted Average = (0.1×49)+(0.2×70)+(4×27)+(12×51) = 4.9+14+108+612 = 4.9+14+108+612 = 738.9

Now, we divided this sum by the total number of atoms in the compound:

Weighted Average Atomic Number = 738.9/(0.1+0.2+4+12) = 738.9/16.3 ≈ 45.33

So, the weighted average atomic number for the compound "In0.1Yb0.2Co4Sb12" is approximately 45.33. The ground state magnetic moments span a range of roughly 1.548, with an average value near 0.38. Its space group numbers go from 139 to 225, averaging around 173.43 when considering stoichiometry.

Then applied this code

df_features_final = featurizer.featurize_dataframe(df_no_outliers, col_id="composition", ignore_errors=True)

After this step, df_features_final will be a DataFrame where the original "composition" column has been expanded into numerous new columns, each representing a specific elemental property or feature derived from the original compositions. This transformation makes the data suitable for input into machine learning algorithms, as these algorithms typically require numerical input features. Then applying scripts filled missing values (NaN) in the df_features dataframe with the mean of their respective columns directly within the existing dataframe was filled, saving memory but altering the original data. It's a common preprocessing step for machine learning models which cannot work with NaN values. Besides, outliers in the df_features1 dataframe were captured by using the Interquartile Range (IQR) method. For each column, it calculates the IQR (Q3 - Q1) and defines bounds (Q1 – 1.5IQR, Q3 + 1.5IQR). analysis or machine learning model training process. Here we got 135 features for further analysis.

4 Chapter 4 – Methods

In this Chapter, we consider the Machine Learning systems that would be suitable for supervised learning to achieve our aims with the thermoelectric figure of merit data. We consider the methodology and results of algorithms seen in literature[20]. We obtained additional methodology and ideas behind each algorithm from Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow by O'Reilly[21].

4.1 Support Vector Regression (SVR):

SVR aims to find a function that approximates the true regression function in the best possible way within an epsilon (ε) distance. This means it tries to fit the error rate within a certain threshold.

How It Works:

ε-insensitive loss: SVR uses an ε-insensitive loss. If the difference between the predicted value and actual value is less than ε, then the error is considered as zero.

This means it doesn't penalize errors within this margin, but errors outside this margin are penalized.

Hyperplane: Just like SVM for classification, the goal of SVR is to determine the best hyperplane which represents the relationship between features and target variable. For regression, this would be a line in 2D, a plane in 3D, and so on.

Key Parameters

C (Regularization): This parameter trades off correct specification of the training examples against maximization of the decision function’s margin. A smaller value of C creates a wider margin, which may result in more training errors but better generalization to the test data.

ε (Epsilon): It defines the margin's width. Errors within this margin aren't penalized, while those outside are.

Kernel: Just like SVM, SVR can use various kernel functions to transform the input data into a higher-dimensional space. Common kernels include linear, polynomial, radial basis function (RBF), and sigmoid.

Advantages:

Flexibility: SVR can capture non-linear relationships by using non-linear kernels (like RBF).

Robustness: The ε-insensitive loss makes it robust to outliers, as errors within the ε margin aren't penalized.

Disadvantages: Sensitive to Hyperparameters: SVR's performance is sensitive to the selection of C, ε, and kernel parameters.

Computationally Intensive: SVR can be computationally expensive for larger datasets.

Usage:

SVR is particularly useful in scenarios where relationships between variables are complex and non-linear. Because of the kernel trick, SVR can capture intricate relationships without the need to add additional features or assume a specific polynomial degree.

4.2 Random Forest

A Random Forest is a collection of individual trees operating as an ensemble. To implement the Random Forest for a regression problem, we build new datasets from the original data, a process called bootstrapping. Bootstrapping ensures that we are not using the same data for every tree, making the model less sensitive to training data noise and outliers.

For each of the bootstrapped datasets, a tree is trained. During this training, random subsets of features are selected at each node of the tree. This random feature selection helps to reduce the correlation between the trees. If every feature were to be used at each node, most trees would end up being too similar, if not identical, especially if there are one or two very strong predictors in the dataset. This would make the ensemble too sensitive to the noise in these features' predictions, thus increasing the variance of the overall model.

For regression, when making a prediction using the Random Forest, we take a data point and pass it through each tree. Each tree provides a continuous prediction (e.g., a price or a measurement). The final prediction from the Random Forest is then the average of all these individual tree predictions. This averaging process helps to further reduce variance and provide a more stable and robust estimate than any single decision tree could offer[24].

Advantages:

Robustness to Noise and Outliers: Due to the bootstrapping and averaging/majority vote process, Random Forests are quite robust to noise in the data.

Handling Missing Data: Random Forest can handle missing data by using techniques like surrogate splits.

Less Overfitting: The bootstrapping and random feature selection help in ensuring that the forest doesn't overfit to the training data.

Importance Measurement: Another beneficial property of Random Forest is its ability to compute a heuristic for determining feature importance. Features used frequently at the top of the trees across the forest tend to be more crucial for the prediction task.

It's essential to understand that while Random Forests offer numerous advantages, they also come with challenges like increased computational load due to the need to train multiple trees. However, the benefits often outweigh these challenges, especially in complex datasets where non-linear relationships and interactions exist.

4.3 XGBoost (Extreme Gradient Boosting)

XGBoost stands for Extreme Gradient Boosting. It's an optimized and regularized version of the Gradient Boosting algorithm, designed to be more efficient and accurate.

Boosting Concept:

Boosting is an ensemble technique in which new models are added to correct the errors made by existing models. Models are added sequentially until no further improvements can be made. XGBoost is a tree-based ensemble algorithm that utilizes the gradient boosting framework.

Regularization:

One of the unique aspects of XGBoost is its ability to do L1 (Lasso regression) and L2 (Ridge regression) regularization. This reduces overfitting and often gives it an advantage in terms of performance[25].

Handling Missing Data:

XGBoost has an in-built routine to handle missing data. When the algorithm encounters a missing value at a node, it tries both the left and right child nodes and learns the direction to move for missing values in future.

Parallel Processing:

XGBoost is designed to be highly efficient. It can perform parallel processing across all cores and even across servers. This means it's significantly faster than other boosting algorithms.

Tree Pruning:

Unlike Gradient Boosting where trees are grown to a fixed size, XGBoost grows the tree and prunes it back using the max_depth parameter, reducing complexity and improving speed.

Cross-validation:

XGBoost has an efficient implementation of cross-validation at each iteration of the boosting process, allowing for efficient model tuning and selection.

4.4 Decision Tree Regressor

A Decision Tree Regressor is a powerful tool for regression and forecasting tasks in machine learning. Just like decision trees for classification tasks, the tree is constructed through an algorithmic approach that identifies ways to split a data set based on different conditions. It's structured as a tree where internal nodes represent feature splits, and the leaf nodes represent the outcome variable, which in this case, is a continuous variable.

Node Types: Root Node: It is the topmost node, where the tree splits first based on the most significant attribute.

Decision Node: The nodes where subsequent splits happen.

Leaf/Terminal Node: The terminal nodes that represent the final outcome, which in this case, will be a continuous value or an average of values.

Feature Selection: Decision Tree Regressors select features based on measures like reduction in variance, which helps to identify the feature that best splits the data into subsets with similar values.

Pruning: To prevent overfitting and to have a more generalizable tree, pruning techniques can be used, which essentially means reducing the size of the tree by cutting off some branches that might just add noise.

Handling Overfitting: Overfitting is a common problem in machine learning where the model learns the noise in the training data. For decision tree regressors, overfitting can be controlled using various strategies:

Limiting tree depth.

Pruning the tree.

Setting a minimum number of samples required to split an internal node.

Ease of Interpretation and Visualization:

Decision Tree Regressors, like their classification counterparts, are visually intuitive and can be more easily interpreted compared to some other regression models. This makes the model transparent and the decisions traceable.

Efficiency: Though decision trees can quickly grow to be complex, their initial computations are less expensive, making them relatively efficient. The complexity is generally logarithmic to the number of points used for training.

Applicability to Various Types of Data: They can handle both categorical and numerical data, allowing for a versatile approach to different regression problems.

Limitations: Sensitivity to small perturbations in the data: A small change can result in a drastically different tree.

The issue of overfitting: Especially if the tree grows too complex.

The problem of finding an optimal decision tree is NP-complete, thus heuristic methods are used which might not always find the most optimal tree.

By leveraging Decision Tree Regressors correctly, it can be a potent tool for solving regression problems, providing both good predictive performance and interpretability.

4.5 Chosen Methods

Having identified several relevant Machine Learning methods for regression tasks, it was decided that the following four algorithms would be most appropriate and strongest for our problem: Decision Tree, Random Forest, SVR and XGBoost[26].

5 Chapter 5 - Supervised Learning

In our recent series of experiments, we harnessed the power of supervised machine learning algorithms to unearth insights into various materials' properties. The dataset, a rich amalgamation of diverse materials, encapsulates the atomic compositions, elemental properties, and a plethora of derived statistical attributes of each material. In this chapter, we detailed the process of using the data for regression analysis, our approach to feature selection, the training methodology employed for our predictive models, and an evaluation of the resultant outcomes.

5.1 Feature Selection

Separate predictors and target: The dataset separated into predictors (X) and the target variable (y). In this case, numerical_Value seems to be the target variable we were trying to predict it, so it's separated from the features.

Train a Random Forest Regressor: A RandomForestRegressor from the scikit-learn library is being utilized to predict the numerical_Value. This algorithm is chosen because of its ability to handle large datasets with higher dimensionality. It can also rank features based on their importance, making it a good choice for both prediction and feature selection.

Feature Importances Extraction: After training, the model is used to retrieve the importance of each feature. Feature importance gives a score for each feature in the data, the higher the score the more important or relevant is the feature towards our output variable.

Selecting Important Features: The following line of code identifie the model using the training data represented by X_train (feature variables) and y_train (target variable). Extracting Feature Importances: After training, the feature_importances_ attribute of the model contains the relative importance of each feature. This line of code assigns these feature importances to the importances variable. The code identifies the top 50 features. The argsort function is called on the importances array, which returns the

indices that would sort the array in ascending order. The [::-1] reverses the array to get it in descending order, and [:50] selects the top 50 indices. These indices are then used to get the corresponding feature names from X.columns

```
#  Data

X = df_features1.drop("numerical_Value", axis=1)
y = df_features1["numerical_Value"]

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Feature selection using RandomForestRegressor
model = RandomForestRegressor()
model.fit(X_train, y_train)
importances = model.feature_importances_
important_features = [X.columns[i] for i in importances.argsort()[::-
1][:50]]
X_train_selected = X_train[important_features]
X_test_selected = X_test[important_features]
```

5.2 Creating The Test Set

The data was split such that 80% was used to train the models and the remaining 20% was used to test the regression models. To create these test and train sets we used the following commands:

```
from sklearn.model_selection import train_test_split
```

```
#  Data
X = df_features1.drop("numerical_Value", axis=1)
y = df_features1["numerical_Value"]

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Feature selection using RandomForestRegressor
model = RandomForestRegressor()
model.fit(X_train, y_train)
importances = model.feature_importances_
```

```
important_features = [X.columns[i] for i in importances.argsort()[::-
1][:50]]
X_train_selected = X_train[important_features]
X_test_selected = X_test[important_features]

# Scaling the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_selected)
X_test_scaled = scaler.transform(X_test_selected)
```

The data has been partitioned into training and testing sets to facilitate the effective training and evaluation of machine learning models. We have allocated 9245 instances, which constitute about 80% of the data, for training, and reserved 2312 instances, approximately 20% of the data, for testing. This setup helps in preventing overfitting and allows us to assess the model's performance on unseen data.

In the training phase, the machine learning algorithms will utilize the 9245 training instances to learn the patterns and relationships that influence the thermoelectric figure of merit, which is the focus of our regression analysis. The models will be evaluated using the 2312 test instances. Predictions derived from these test features will be cross-verified with the actual test labels to evaluate the performance and accuracy of the models. The data was now prepared for Machine Learning algorithms.

5.3 Training And Evaluating Support Vector Regression Model

A Support Vector Regression (SVR) model is initialized with a polynomial kernel, and specific values for the regularization parameter C and the epsilon-tube within which no penalty is associated in the training loss function with points predicted within a distance epsilon from the actual value. The scaler is fitted to the selected training data, and it is then used to transform both the training and test datasets. The SVR model is fitted with the scaled training data.

```
# Get the selected features
```

```
X_train_selected = selector.transform(X_train)

X_test_selected = selector.transform(X_test)

# Initialize the SVR model

svr = SVR(kernel='poly', C=10, epsilon=0.5)

# Scale the features using StandardScaler

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train_selected)

X_test_scaled = scaler.transform(X_test_selected)

# Fit the model to the training data

svr.fit(X_train_scaled, y_train)

# Calculate and print the train R² score

train_r2_score = svr.score(X_train_scaled, y_train)

print("Train R² Score:", train_r2_score)

# Calculate and print the test R² score

test_r2_score = svr.score(X_test_scaled, y_test)

print("Test R² Score:", test_r2_score)

# Make predictions on the training and testing datasets

y_train_pred = svr.predict(X_train_scaled)

y_test_pred = svr.predict(X_test_scaled) # Calculate and print the
MSE for the training dataset

train_mse = mean_squared_error(y_train, y_train_pred)

print("Train MSE:", train_mse)

# Calculate and print the MSE for the testing dataset

test_mse = mean_squared_error(y_test, y_test_pred)

print("Test MSE:", test_mse)

from sklearn.model_selection import cross_val_score
```

```
# Perform cross-validation
cv_scores = cross_val_score(svr, X_train_scaled, y_train, cv=10, scoring='r2')
# Print the cross-validation scores for each fold
print("Cross-Validation Scores:", cv_scores)
# Calculate and print the average R^2 score
average_r2_score = cv_scores.mean()
print("Average R^2 Score:", average_r2_score)
```

The performance of the model is then evaluated using the coefficient of determination (R^2 score) for both the training and testing datasets. The code as a whole sets up and executes a machine learning pipeline, which involves data splitting, feature selection, data scaling, model initialization and training, and performance evaluation using the R^2 score. The mean squared error is calculated for both the training and testing datasets using the mean_squared_error function. The function is applied to the true and predicted values for both the training and testing datasets.

We explored cross-validation in Section 5.5.1.

This SVR model was trained with the 9245 instance's train features and labels and predicted the labels of the remaining 2312 instances. The training and testing accuracy was obtained by almost 73% and 71% respectively which shows that the trained model can explain 73% of the variance in the dependent variable (y_train) using the independent variables (X_train) in the training dataset. Given that a higher R2 value (closer to 1) suggests a better fit, the model has a decent fit to the training set of data.

MSE is a risk metric corresponding to the expected value of the squared (quadratic) error or loss and is a common metric to evaluate the performance of regression models. Lower MSE values indicate a better fit of the model to the data, while higher values indicate a worse fit. The Mean Square Error for training data is (0.0663) and

testing data is 0.083 that signifies a better fit to the data, and in this case, the model seems to have a fairly low error on the training set. Similarly, the test MSE is quite close to the training MSE suggests that the model generalizes well to new, unseen data, without overfitting or underfitting. The model's performance is indicated by its low MSE values and high R^2 scores. However, further improvement could be achieved by experimenting with different features, tuning parameters, using other regression algorithms, and evaluating the model using metrics like MAE or RMSE. Overall, the model is well-suited for both training and unseen data.

5.4 Training And Evaluating Random Forest Regression Model

A high training R2 score of 0.8787 means that the model adequately accounts for 87.87% of the variation in the training set of data. It is a good score, indicating that the model closely matches the training set of data.

The model accounts for 87.67% of the variation in the test data, according to the testing R2 Score (0.8767), which is also high. The fact that the testing R2 score is relatively close to the training R2 score is a strong sign that the model generalises effectively to new data.

The average R^2 score is 0.8849, which means, on average, the model can explain approximately 88.49% of the variance in the dependent variable - which is a good score indicating a potentially highly accurate model.

Analysis

High Average R^2 Score: The high average R^2 score indicates that the model performs well across different subsets of data, indicating that it is not just fitting to a specific portion of your data but generalizing well across the data set.

Consistent Cross-Validation Scores: The cross-validation scores are consistently high, indicating that the model performs well across different subsets of the data, which signifies a well-fitted model.

Close Range of Scores: The scores range from 0.8814 to 0.8890, indicating that the model performance does not fluctuate much between different splits of the data, demonstrating robustness in the model.

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor

# Initialize the Random Forest Regressor
random_forest = RandomForestRegressor()

# Define the grid of parameters to search
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Initialize Grid Search with a 3-fold cross-validation
grid_search = GridSearchCV(random_forest, param_grid, cv=3, scoring='r2',
verbose=2, n_jobs=-1)

# Fit Grid Search to the scaled training data
grid_search.fit(X_train_scaled, y_train)

# Get the best parameters and the best R² score
best_params = grid_search.best_params_
best_score = grid_search.best_score_
print("Best Parameters:", best_params)
print("Best R² Score:", best_score)
# Use the best estimator for further predictions or analysis
best_random_forest = grid_search.best_estimator_
```

The text describes a grid search cross-validation process for optimizing hyperparameters in a random forest regression model. The model is trained and evaluated 324 times, testing every possible combination of hyperparameters. The optimal hyperparameters are 'max_depth','min_samples_leaf','min_samples_split', and 'n_estimators'. The grid search results in a model with a high predictive accuracy of approximately 0.8842, explaining about 88.42% of the variance in the dependent variable. The model is then trained using these optimal parameters and evaluated on a test set to ensure its robustness.

5.5 Feature Importance

The data analysis used Support Vector Regression (SVR) and Random Forest models and Decesion Tree regression models to identify key features influencing the thermoelectric figure of merit. They identified temperature_average and temperature_value as significant factors, while Random Forest emphasized temperature-related features and the mean Neodymium valence. The space group number also played a significant role in the analysis. Both models emphasized the importance of temperature parameters and material properties in predicting the figure of merit (Figure 5 and 6). But by following figure 7 ‘temperature parameters as well as ‘MagpieData avg_dev Electronegativity’ were the most significant features. Hence, overall, we can say that ‘Temperature_Average’ and ‘temperature_value’ identified as the most significant features in determining the thermoelectric figure of merit. ‘MagpieData mean

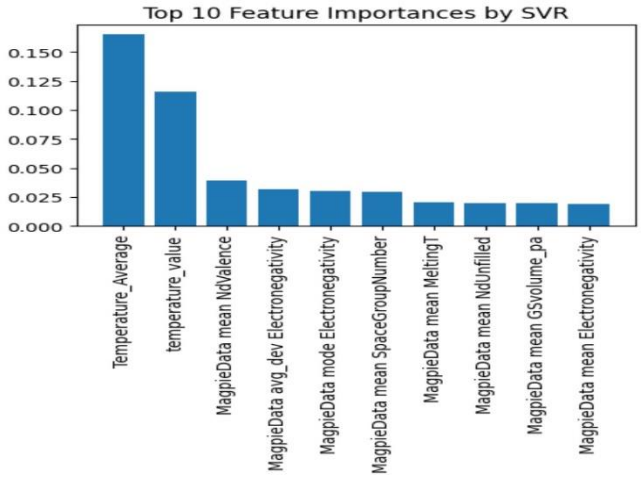


Figure 5: SVR Model Feature Importance Scores

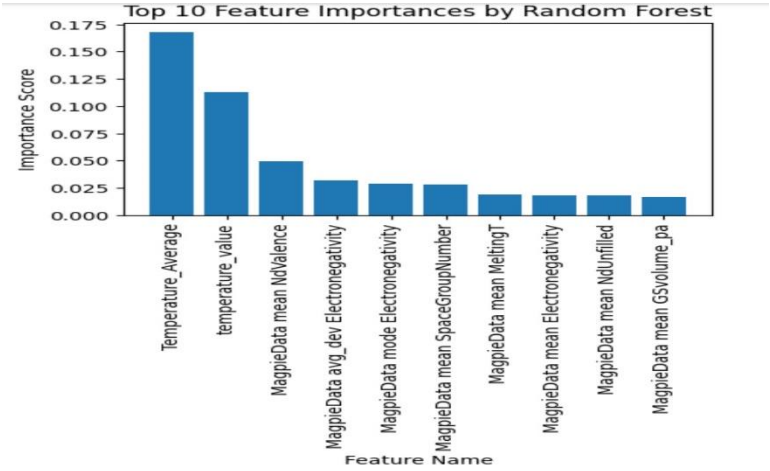


Figure 6: Random Forest Model Feature Importance Scores

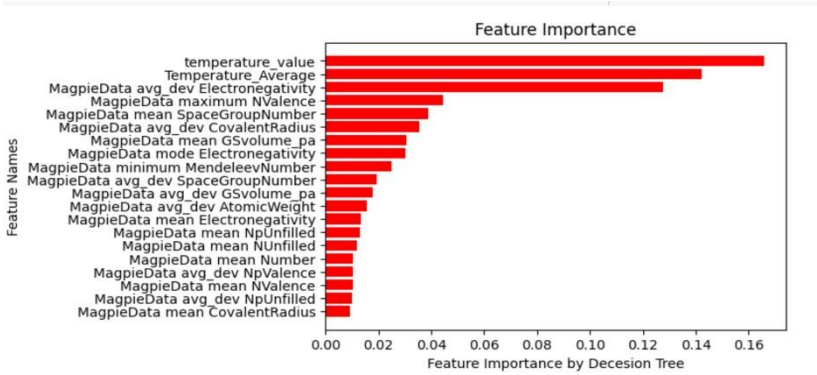


Figure 7: Decesion Tree Regression Model Feature Importance

Electronegativity' along with the 'MagpieData mean NdUnfilled' features were again on the lower end of the importance scale.

5.6 Feature Importance with The SHAP Method

To further understand the effect features had on these models, we looked at a technique of Explainable Machine Learning: the SHAP(SHapley Additive exPlanations) method. This was used as a more advanced method to explain how changes in feature values affected model output. In the previous subsection, we looked

at feature coefficients that demonstrated how important they were to the model, however they did not account for the local importance of the feature. The trained XGBoost model is used to initialise a SHAP object. Calculating SHAP values, which measure the influence of features on model predictions, was done using the explanation object.

```
# Train an XGBoost model
model = xgboost.train({"learning_rate": 0.01}, xgboost.DMatrix(X_train,
label=y_train), 100)
# Create a SHAP explainer object
explainer = shap.Explainer(model)
# Calculate SHAP values for X_test
shap_values = explainer.shap_values(X_test)
# If you want to set a custom figure size, you can do it like this
plt.figure(figsize=(8, 6))
# Plot the summary of SHAP values with max_display parameter set to 12
shap.summary_plot(shap_values, X_test, max_display=12)
```

Using the shap.summary_plot function, we were plotting the computed SHAP values in a summary manner. For each feature, the plot shows the average absolute SHAP values, which aids in understanding how each factor affects the model's predictions. The max_display=12 parameter restricts the plot to the top 12 features, helping to draw attention to the features that have the greatest influence.

Figure 8 demonstrates the influence each feature had on model prediction, according to SHAP values. In this example, the XGBoost model demonstrated that ‘Temperature_Average’ was the most important feature, followed by MagpieData mean NdValence , MagpieData avg_dev NdUnfilled , MagpieData mode Electronegativity. In Figure 9, Temperature_Average fluctuates with high positive and low negative impacts, implying a complex relationship with the target variable, and hence demands a detailed investigation for optimal utilization. MagpieData mean NdValence stands as a vital feature, predominantly steering the predictions in a positive direction, indicating it's a crucial element in predicting favorable outcomes.

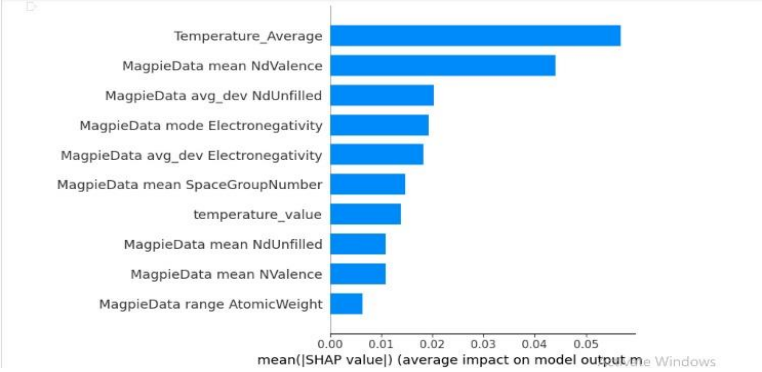


Figure 8: SHAP Method - XGBoost Feature Importance.

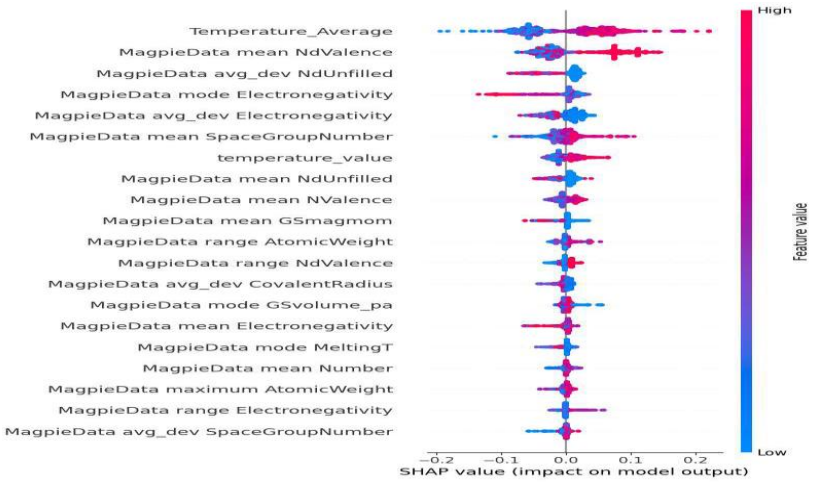


Figure 9: SHAP Method - XGBoost Feature effects On Model

MagpieData avg_dev NdUnfilled and MagpieData mode Electronegativity notably influence the model's predictions negatively. These features need to be managed carefully, potentially maintaining lower values to attain desirable results. Temperature_Value has a positive influence, suggesting that materials with higher values in this aspect are associated with favorable outcomes. MagpieData dev_avg SpaceGroupNumber, conversely, impacts the model negatively, indicating that a reduction in this feature's values might promote better results. Moving forward, a strategic approach focusing on manipulating these critical features can potentially enhance material performance.

5.7 Residual Analysis (Random Forest Regression Model)

The most of the residuals are clustered around the zero line, it indicates that the model is making very accurate predictions, as the differences between the actual and predicted values are very small.

Good Model Fit: Since most of the residuals are around the zero line, it indicates a good fit. The model seems to be capturing the underlying patterns in the data quite well.

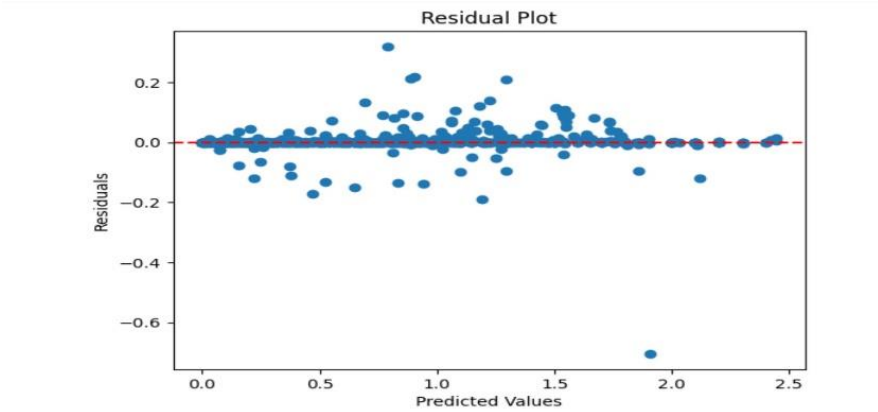


Figure 10: Residual Analysis for Random Forest Regression Model

5.8 Standard Eror of Random Forest Regression Model

```
# Initialize the RandomForestRegressor
rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)

# Perform 5-fold cross-validation
cross_val_scores = cross_val_score(rf_regressor, X, y, cv=5,
scoring='neg_mean_squared_error')

# Calculate the standard deviation of mean squared error across folds
std_mse = np.std(-cross_val_scores)

print(f"Standard Deviation of Mean Squared Error: {std_mse}")
```

The standard deviation of the mean squared error (MSE) being 0.07120510070801299 is quite small that seems the model's performance is relatively

consistent, indicates that the model is likely not overfitting and is generalizing well to different subsets of the data.

5.9 Model Evaluation

It was very important to evaluate the three algorithms. We could determine which algorithm was the best at modelling the data and best at prediction. Let us observe each model's train and test accuracy scores.

Algorithm	Train Accuracy(%)	Test Accuracy(%)
SVR	73%	71%
Random Forest	88%	87%
XGBoost	89%	65%
Decision Tree Regression model	86%	85%

Table 1: Train And Test Accuracy Scores For three Classification Algorithms.

Overall, all models except for XGBoost have shown commendable generalization capabilities with minimal gaps between training and testing accuracies. In contrast, the XGBoost model seemed to have fallen into the trap of overfitting, which might require further tuning or regularization to improve its performance on unseen data. The Random Forest model appeared to be the most promising choice for this specific task given its high testing accuracy, though the final decision should also consider other factors like computational efficiency and specific requirements of the task at hand.

Although there was a mention of a singular outlier (figure 10), indicating a scope for further refinement, the overarching narrative clearly highlights the Random Forest algorithm as a robust, reliable, and promising tool for predictive analysis, standing a notch above the rest in this assessment. The constructive feedback included hints at potential improvements, leaving room for even more refined results in the future.

Section 5.5.1: Accuracy With Cross-Validation

K-Fold Cross-Validation

In this section, we introduced the concept of K-fold cross-validation as a superior method for evaluating the models. This process involves splitting the dataset into K equal parts, or 'folds', and using K-1 of them to train the model, with the remaining part used for testing. This process is repeated K times so that every part is used as a testing set once. This method helps in making the most out of a limited dataset for both training and testing, thereby giving a more robust estimate of the model's performance on unseen data.

The K-fold cross-validation method was used during the investigation to evaluate each model's performance on unobserved data and provide a trustworthy estimate of that performance.

The SVR model demonstrated a decent stability and predictive power, with R^2 scores hovering between roughly 72.74% and 71.25% across the folds, culminating in an average R^2 score of about 72.51%. Despite not being the most potent among the models tested, it still maintains a respectable predictive efficacy, capable of explaining a considerable proportion of the variance in the dependent variable.

Meanwhile, the Decision Tree regression model emerged as a strong contender, showcasing both stability and high predictive power, reflected in its average R^2 score of about 0.8436. The tight grouping of individual R^2 scores (ranging from 0.84 to 0.868) across the folds indicates a model that's well-fitted, not being overly sensitive to variations in the training data subsets.

The Random Forest model, on the other hand, delivered an outstanding performance, with an impressive average R^2 score of nearly 0.8848. This high score, accompanied

by a narrow range of individual fold scores (approximately 0.8617 to 0.8991), highlights the model's robustness and excellent predictive accuracy, attesting to its ability to handle unseen data proficiently.

To sum up, all three models (without XGBoost) exhibited promising capabilities, with the Random Forest standing out as the most reliable choice for predictions on new data, owing to its superior accuracy and stability, followed closely by the Decision Tree regression model. The SVR, though not as potent, still offers a viable option, demonstrating considerable predictive power and stability across different data subsets.

Chapter 6 - Unsupervised Learning

In this chapter, we apply unsupervised learning to analyze the thermoelectric figure of merit (ZT) of various materials. Utilizing clustering algorithms, we aim to classify materials into distinct groups based on their thermoelectric properties, potentially uncovering patterns that indicate high ZT values.

6.1 Dimensionality Reduction

Given the high-dimensional nature of material science data, we initiated a dimensionality reduction process to simplify the dataset while retaining essential information about the thermoelectric figure of merit. This step aims to merge correlated features into single entities, facilitating a smoother and more insightful analysis. The refined dataset will then be used to deploy clustering algorithms, helping to unearth potential breakthroughs in the field of thermoelectric.

6.1.1 Principal Component Analysis (PCA)

PCA stands as a paramount algorithm in reducing dimensionality, adept in pinpointing hyperplanes that account for the majority of data variance. Subsequently, it finds

orthogonal hyperplanes to encapsulate the remaining variance, defining unit vectors along these hyperplanes as principal components.

6.1.2 Projecting Down to n Dimensions

The next step involves projecting the dataset onto a hyperplane defined by the selected principal components, a strategy to preserve the maximal amount of variance.

6.1.3 Choosing the Right Number of Dimensions

The following script uses Principal Component Analysis (PCA) to reduce dataset dimensionality, increasing interpretability and minimizing information loss. It initializes a PCA object with a 0.95 parameter, applies the transformation to the training dataset, and then applies the transformation to the testing dataset. This ensures that the essential characteristics and patterns in the data are preserved, facilitating efficient analyses and machine learning model applications. The script demonstrates the effectiveness of PCA in reducing dimensionality.

```
# Apply PCA to reduce dimensionality
pca = PCA(n_components=0.95) # Adjust n_components as needed
X_train_pca = pca.fit_transform(X_train_selected)
X_test_pca = pca.transform(X_test_selected)
Principal_Components_n = pca.fit_transform(X_train_scaled)

# Printing the number of principal components selected by PCA
print(pca.n_components_)
```

PCA was applied to reduce data dimensionality while retaining 95% of the original data's variance, which resulted in 23 principal components. Despite having 23 components, only the top 10 were selected for a deeper analysis to make the dataset less complex and more manageable. Two plots were generated: The first illustrates the variance explained by each of the top 10 components separately. The second showcases the cumulative variance captured as the number of components increases, aiding in determining the optimal number of components to use for further analysis without losing significant data detail.

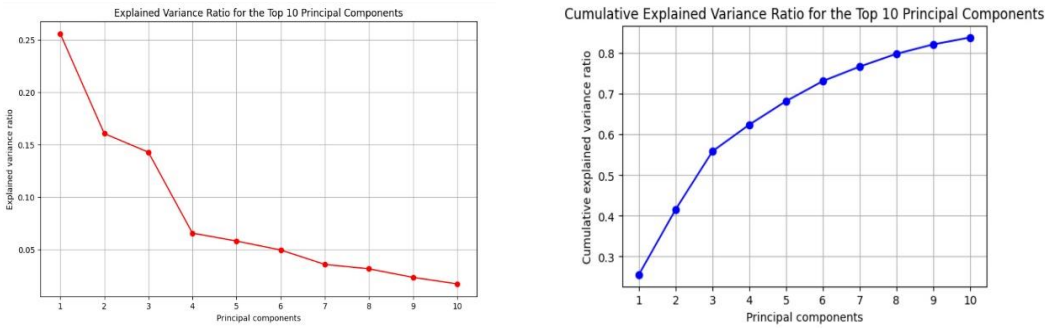


Figure 11: Explained and cumulative explained Variance ratio for the Top 10 Principal

The data revealed a significant decrease in the percentage of variance explained by each successive principal component. The first three components explained a large portion of the variance (56.88% combined), representing a substantial part of the dataset's original structure. From the fourth dimension onwards, the explained variance decreased markedly, with the tenth component accounting for just 1.7% of the variance. This suggests that the bulk of the important information is contained within the first few components, and utilizing these might be sufficient for most analyses without incorporating the latter components that contribute less to the explained variance.

This dimensionality reduction is vital as it simplifies the data structure, paving the way for more efficient data analysis and modeling in subsequent steps, without forfeiting essential information.

6.1.4 Feature Association

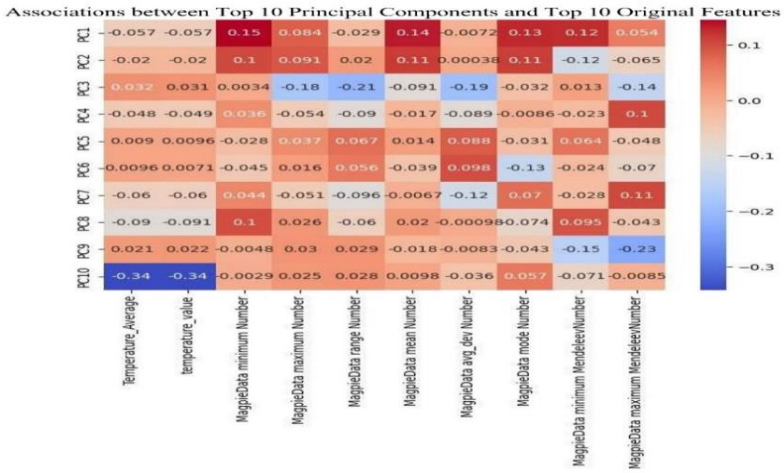


Figure 12: Feature Association of top 10 PC and top 10 original features

The composition of the first and second principal component was very interesting. These principal components had positive associations with all features (except two). PC1 had positive relationship with MagpieData minimum number(r=0.15), which was high among all. We therefore inferred that PC10 was most negatively associated with Temperature_Average and temperature value. Besides, MagpieData mean Number and MagpieData mode number were positively correlated with PC1. But all the associations were very weak.

6.2 K-Means Clustering

K-Means is an unsupervised machine learning algorithm used for clustering data points into groups or clusters based on their similarity. It is known for its simplicity and efficiency in dividing data into K clusters. The code performs K-Means clustering with two clusters (K=2) on the first 10 dimensions of the data. It calculates the inertia (within-cluster sum of squares) and silhouette score to assess the quality of the clustering. The KM_labels variable stores the cluster labels assigned to each data point.

In summary, this code applies K-Means clustering to the data and evaluates how well the data points are grouped into two clusters using the inertia and silhouette score

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
# Initialize the K-Means algorithm with K=2 clusters and set n_init
explicitly
KM = KMeans(n_clusters=2, n_init=10)
# Fit the K-Means model to your data and obtain cluster labels
KM_labels = KM.fit_predict(X_train.iloc[:, 0:10])  # Use the first 10
dimensions
# Calculate the inertia (within-cluster sum of squares) and silhouette
score
inertia = KM.inertia_
silhouette = silhouette_score(X_train.iloc[:, 0:10], KM_labels)
```

Cluster 1: Contains 3779 data points.

Cluster 2: Contains 1483 data points.

Cluster 3: Contains 2016 data points.

Cluster 4: Contains 976 data points

6.3 Finding The Optimal Number Of Clusters

Initially, the code executed K-Means with 2 clusters, calculated inertia (a metric indicating the sum of squared distances between points and their respective clusters' centroids) and silhouette score (a measure of how similar an object is to its own cluster compared to other clusters). Then, it iteratively performed K-Means clustering with a number of clusters ranging from 1 to 10, plotting the inertia values to help find the optimal number of clusters using the elbow method, which identified a point where adding more clusters doesn't provide much better fit to the data

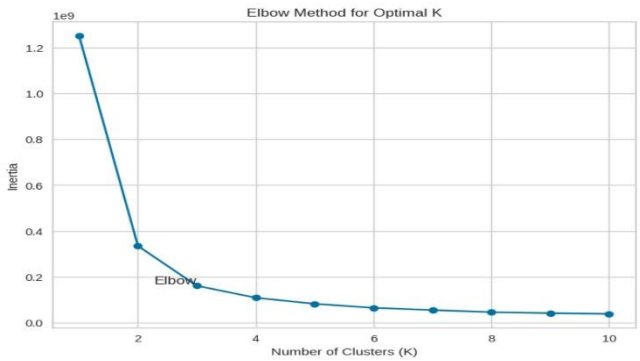


Figure 13: Selecting The Number Of Clusters *K* Using The 'elbow rule'.

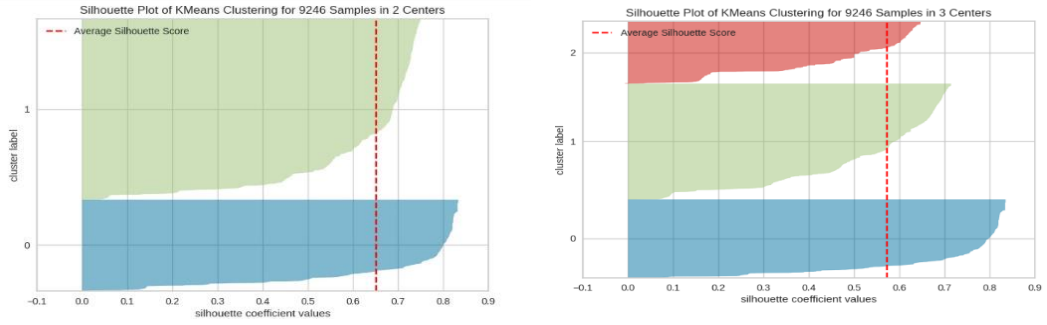


Figure 14: Silhouette Diagram: *K= 2 and k = 3*

This script segmented data into 2 and 3 clusters using the K-Means algorithm based on the first 10 principal components. It then visualizes the clusters and their centroids on a scatter plot, with clusters differentiated by colors and centroids marked in red. The visualization uses only the first two principal components for simplicity.

6.3.1 Results of The K-Means Algorithm

The script clustered data into three groups using the K-Means algorithm and visualizes them on a 2D scatter plot. The average silhouette score of 0.57(k=3) indicates good separation, with points in each cluster closer to each other and farther away from others, indicating a well-separated clustering structure. From the figure 15,

we can say that the clusters seem to have a reasonably good separation, indicating that the clustering might be meaningful.

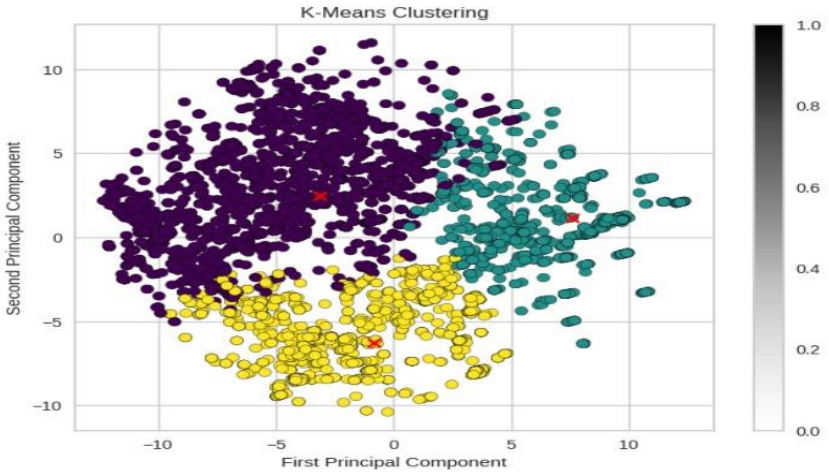


Figure 15: K-means clustering for PC1 and PC2

7 Conclusion

Overall, the machine learning experiment that we conducted revealed that among the three models tested - Random Forest, Decision Tree, and Support Vector Regression (SVR) - Random Forest emerged as the most reliable option for making predictions on new data due to its higher accuracy and stability. It was followed closely by the Decision Tree model, with SVR also proving to be a reasonable alternative, albeit with slightly less predictive capability. Besides, Temperature_Average And temperature_value were the most significant features.

The analysis of the principal components illustrated intriguing relationships and associations with various features. Particularly, the first principal component (PC1) showed a positive correlation with several attributes, notably having a strong association with MagpieData minimum number. Additionally, the tenth principal component (PC10) was found to have significant negative associations with Temperature_Average and temperature value, providing potentially meaningful

insights into the data's underlying structure. Moreover, certain attributes such as MagpieData mean number and MagpieData mode number were positively correlated with PC1.

Furthermore, the K-Means clustering implementation effectively grouped the data into well-segregated clusters, as evidenced by an impressive average silhouette score of 0.57 for k=3. This indicates that the data points within each cluster are considerably closer to each other as compared to points in different clusters, suggesting a potentially meaningful clustering structure. The visualization facilitated by the script, which represented clusters and their centroids on a 2D scatter plot (utilizing only the first two principal components for simplicity), substantiated the apparent good separation between clusters, hinting at a significant and structured segmentation within the data.

Therefore, based on the results obtained and visual insights drawn from the scatter plot in figure 15, it can be inferred that the machine learning models and clustering algorithm have succeeded in uncovering significant patterns and groupings within the data, which might pave the way for more focused and informed analysis in future studies.

8 Future Work

In the pursuit of developing a more robust and insightful analysis, future endeavors should significantly consider the following avenues of exploration and development:

- In-depth Exploration of Unused Data: Collaborate with domain experts to analyze the 111 elements that were not previously explored, to potentially unveil more comprehensive insights and augment the predictive accuracy of the models. Investigate the renewable or sustainable energy sector applications of the 111 elements, potentially unveiling new research and development avenues.
- Feature Engineering: Enhance the current feature set through the creation of new features, possibly by combining existing ones or leveraging domain knowledge to pinpoint impactful features, aiming to boost the R^2 value beyond 88%.

- Model Optimization: Undertake further optimization of the machine learning models by fine-tuning parameters and experimenting with advanced algorithms, to strive for higher predictive accuracy.
- Real-time Analysis and Predictions: Depending on the data's nature and the ultimate goal of the analysis, establish a system for real-time analysis and predictions, a beneficial trajectory for future work.
- Publication and Sharing of Findings: Once the analysis matures, publishing the findings in relevant forums or journals to gather feedback from the wider community, aiding in further refinement of the analysis and models.
- Energy Efficiency Correlations: Explore correlations between the analyzed features and energy efficiency parameters to discern the role these elements might play in sustainable energy systems.
- Lifecycle Analysis: Undertake lifecycle analyses to gauge the environmental impact of these elements, offering a comprehensive view of their sustainability.
- Integration with Renewable Energy Systems: Analyze potential integration avenues of these elements into existing or future renewable energy systems, possibly as components in renewable energy technologies.
- Policy and Regulation Study: Study the policy and regulatory landscape surrounding these elements concerning renewable energy development and sustainability initiatives.
- Techno-economic Analysis: Conduct a techno-economic analysis to understand the economic viability and technical feasibility of utilizing these elements in renewable energy applications.
- Collaborations and Partnerships: Collaborate with renewable energy focused entities for potential joint research initiatives fostering the development of sustainable energy solutions.
- Community Engagement and Education: Engage with the community to foster awareness and education about these elements' potential uses in renewable and sustainable energy sectors.

- Innovation and Technology Development: Foster innovation by exploring the development of new technologies or processes leveraging these elements to enhance renewable energy systems' efficiency and sustainability.
- Publication in Renewable Energy Forums: Publish our findings in renewable energy forums or journals to actively engage with the community in this sector and receive valuable feedback and insights.

By focusing on these aspects, the future work promises to unearth intricate details from the data, offering a richer and more accurate analysis, potentially contributing significantly towards fostering a green and sustainable future.

References

1. Bell, L. E. Cooling, Heating, Generating Power, and Recovering Waste Heat with Thermoelectric Systems. *Science* 2008, *321*, 1457− 1461.
2. Rowe, D. M. *CRC handbook of thermoelectrics* (CRC press, 2018).
3. Alam, H. & Ramakrishna, S. A review on the enhancement of figure of merit from bulk to nano-thermoelectric materials. *Nano Energy* **2**, 190–212 (2013).
4. Alpaydin, E. *Introduction to machine learning* (MIT press, 2020).
5. Gaultois, M. W. *et al*. Data-driven review of thermoelectric materials: performance and resource considerations. *Chemistry of Materials* **25**, 2911–2920 (2013).
6. Gaultois, M. W. *et al*. A recommendation engine for suggesting unexpected thermoelectric chemistries. *arXiv preprint arXiv:1502.07635* (2015).
7. Hautier, G. Prediction of new battery materials based on ab initio computations. In *AIP Conference Proceedings*, vol. 1765, 020009 (AIP Publishing LLC, 2016).
8. Ong, S. P. *et al*. Python materials genomics (pymatgen): A robust, open-source python library for materials analysis. *Computational Materials Science* **68**, 314–319 (2013).

9. Carrete, J., Mingo, N., Wang, S. & Curtarolo, S. Nanograined half-heusler semiconductors as advanced thermoelectrics: An ab initio high-throughput statistical study. *Advanced Functional Materials* **24**, 7427–7432 (2014).

10. Gorai, P. *et al*. Te design lab: A virtual laboratory for thermoelectric material design. *Computational Materials Science* **112**, 368–376 (2016).

11. Yan, J. *et al*. Material descriptors for predicting thermoelectric performance. *Energy & Environmental Science* **8**, 983–994 (2015).

12. Tshitoyan, V. *et al*. Unsupervised word embeddings capture latent knowledge from materials science literature. *Nature* **571**, 95–98 (2019).

13. Swain, M. C. & Cole, J. M. ChemDataDxtractor: a toolkit for automated extraction of chemical information from the scientific literature. *Journal of Chemical Information and Modeling* **56**, 1894–1904 (2016).

14. Mavračić, J., Court, C. J., Isazawa, T., Elliott, S. R. & Cole, J. M. ChemDataExtractor 2.0: Autopopulated ontologies for materials science. *Journal of Chemical Information and Modeling* **61**, 4280–4289 (2021).

15. Sierepeklis, O. & Cole, JM. A thermoelectric materials database auto-generated from the scientific literature using ChemDataExtractor, *figshare*, https://doi.org/10.6084/m9.figshare.19658787 (2022).

16. https://www.materialsproject.org/

17. W.H. Bragg: The significance of crystal structure. *J. Chem. Soc. Trans*. **121**, 2766 (1922).

18. A. Van De Walle: A complete representation of structure-property relationships in crystals. *Nat. Mater*. 7, 455–458 (2008).

19. Zhang, Z., Zhang, R., Qi, N., Wu, Y. & Chen, Z. Microscopic origin of the extremely low thermal conductivity and outstanding thermoelectric performance of $BiSbX_3$ (X = S, Se) revealed by first-principles study. *Physical Chemistry Chemical Physics* 22, 15559–15566
20. Sendek, A. D.; Cubuk, E. D.; Antoniuk, E. R.; Cheon, G.; Cui, Y.; Reed, E. J. Machine Learning-Assisted Discovery of Solid Li-Ion Conducting Materials. *Chem. Mater.* 2019, *31*, 342−352.
21. Geron, A. (2019). Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow. O'Reilly. Sebastopol, CA.
22. Hoerl, A.E., & Kennard, R.W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. Technometrics, 12(1), 55-67.
23. Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological), 267-288
24. Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.
25. Chen, T., & Guestrin, C. (2016, August). XGBoost: A scalable tree boosting system.
26. Ward, L., Dunn, A., Faghaninia, A., et al. "Matminer: An open source toolkit for materials data mining." Computational Materials Science 152 (2018): 60-69.
27. Ward, L., Agrawal, A., Choudhary, A., & Wolverton, C. "A general-purpose machine learning framework for predicting properties of inorganic materials." npj Computational Materials 2.1 (2016): 1-7.