

**DEVELOPMENT OF MACHINE LEARNING MODELS FOR
PREDICTING PROPERTIES OF SODIUM-ION BATTERY
MATERIALS**

By

MONARENG KELETSO MABEL

RESEARCH DISSERTATION

Submitted in fulfilment of the requirements for the degree of

MASTER OF SCIENCE

In

PHYSICS

In the

FACULTY OF SCIENCE AND AGRICULTURE

(School of Physical and Mineral Sciences)

at the

UNIVERSITY OF LIMPOPO

SUPERVISOR: Dr P. S. NTOAHAE

CO-SUPERVISOR: Prof R. R. MAPHANGA (CSIR)

2023

ABSTRACT

In this work, machine learning regression techniques are applied to a large amount of data from Materials Project Database, to develop machine learning models capable of accurately predicting the properties of sodium-ion battery cathode materials. Different machine learning models, namely, Bayesian ridge, gradient boosting regressor, light gradient boosting machine, extra trees regressor, random forest, and orthogonal matching pursuit are successfully developed and validated, using SIB materials' properties calculated from DFT as input dataset, with the models' efficiency based on elemental properties of materials constituents feature vectors.

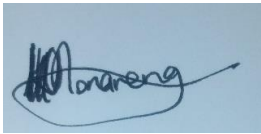
The target properties in this work include formation energy, final energy, Fermi energy, energy above hull, density, and band gap. The importance of feature vectors derived from the properties of materials' chemical compounds and elemental properties of their constituent is evaluated. The average covalent radius and the average single bond covalent radius were found to be the most important descriptors in predicting formation and final energies, whilst the estimated face centred cubic lattice parameter, the average electronegativity, and the average density to be the most important descriptors for predicting the Fermi energy. The optimal features in predicting energy above hull were found to be the sum of sound velocity, sum of total unfilled electron, and the average ground state energy. Furthermore, the results show that maximum mass specific heat capacity and variance of density functional theory energy per atom descriptors are the most essential in accurately predicting the materials density and valence electron in d shell, the average radius and the average electronegativity been the most important features for predicting band gap.

Amongst various algorithms that are evaluated, the Bayesian ridge model is found to be the best model in predicting the formation energy with an accuracy of 0.99 and 0.01 eV coefficient of determination and mean square error, respectively, and final energy of 0.98 and 0.03 eV accuracy for the coefficient of determination and mean square error, respectively. Light gradient boosting machine model is found to be the best model in predicting the Fermi energy with an accuracy of 0.82 and 0.52 eV coefficient of determination and mean square error, respectively, and energy above hull of 0.67 and 0.01 eV, for the coefficient of determination and mean square error, respectively.

Extra trees regressor is found to be the best model in predicting the density with an accuracy of 0.95 and 0.09 g/cm³ coefficient of determination and mean square error, respectively, and band gap of 0.78 and 0.66 eV, for the coefficient of determination and mean square error, respectively. The models demonstrate an improvement accuracy in predicting the sodium-ion battery materials properties as demonstrated by the regression scores.

DECLARATION

I, Keletso Mabel Monareng, hereby declare that this research project titled **‘Development of machine learning models for predicting the properties of sodium-ion battery materials’** submitted for the degree Master of Science in Physics at the University of Limpopo, is my original research work. Wherever I used someone’s work I acknowledged in full references. This work has not been submitted for a degree or examination at any other university. The work was done under the guidance of Dr. PS Ntoahae [supervisor] at the University of Limpopo Physics Department and Prof. RR Maphanga [co-supervisor], at the Council for Scientific and Industrial Research.

A blue rectangular box containing a handwritten signature in black ink. The signature appears to be 'Monareng' with a stylized flourish at the end.

Monareng K. M

19/06/2023

Date

DEDICATION

I dedicate this work to my parents “Mr. K.D who inspired the bookworm in me and Mrs. K.N Monareng who have always encouraged me to pursue my dreams”.

ACKNOWLEDGEMENTS

The study is funded by the Department of Science and Innovation-Interbursary Support (IBS) Programme of the Council for Scientific and Industrial Research, and the funding support is duly acknowledged. Thanksgiving to the University of Limpopo, department of physics for allowing me to undertake this study. My sincere gratitude goes to my supervisor Dr P.S Ntoahae and co-supervisor Prof. R.R Maphanga for helping me throughout the project, for their positive criticism and all the guidance, the completion of this study could have been impossible without their expertise. Data used for this project was provided by Materials Project database, this project would not have been possible without materials project data. I would also like to thank my parents “Mr. K.D. and Mrs. K.N. Monareng” for encouraging, invaluable assistance to me, without you none of this would indeed be impossible. Last “but not least” I am thankful to all my siblings “Batobeleng, Babotsanne, Lesenyang, Kamogelo, Motopeng, and Monareng” for your support financially and physically, my source of happiness Thabiso for your support and guidance throughout the journey, and Ms. Ramakoloi for her support and encouragement throughout my study. Above all, to the shepherd and bishop of our soul who has given me knowledge, and opportunities that have permitted me to finally finish this dissertation. The almighty!!!!

LIST OF ABBREVIATIONS

AFLOW	Automatic Flow for Materials Discovery
AI	Artificial Intelligence
ANN	Artificial Neural Network
Ave	Weighted Average
AutoML	Automated machine learning
BCC	Body Centred Cubic
BCV	Bootstrap Cross Validation
BR	Bayesian Ridge
CA	Cluster Analysis
CBFVs	Composition-Based Feature Vectors
CSIR	Council for Scientific and Industrial Research
CV	Cross Validation
DFT	Density Functional Theory
DNN	Deep Neural Network
DT	Decision Tree
ET	Extra Trees
ETR	Extra Trees Regressor
FCC	Face Centred Cubic
FNN	Feedforward Neural Network
GA	Genetic Algorithm
GBR	Gradient Boosting Regressor
Gmean	Geometric mean

HHI	Herfindahl-Hirschman index
Hmean	Harmonic Mean
ICSD	Inorganic Crystal Structure Database
KNN	K Nearest Neighbor
KRR	Kernel Ridge Regression
LOO	Leave one Out
LOOCV	Leave one Out Cross Validation
LPOCV	Leave p Out Cross Validation
LGBM	Light Gradient Boosting Machine
LIB	Lithium-Ion Battery
LIBs	Lithium-Ion Batteries
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
Max	Maximum-pooling
Min	Minimum-pooling
ML	Machine Learning
MSE	Mean Square Error
MPD	Materials Project Database
NaN	Not a Number
NN	Neural Network
NOMAD	NOvel MAterials Discovery
OMP	Orthogonal Matching Pursuit
OQMD	Open Quantum Materials Database
Pymatgen	Python Materials Genomes
RF	Random Forest

RFR	Random Forest Regressor
RMSE	Root Mean Square Error
SAA	Simulated Annealing Algorithms
SIB	Sodium-Ion Battery
SIBs	Sodium-Ion Batteries
SOC	State of Charge
Std	Standard (Deviation)
SVM	Support Vector Machine
SVR	Support Vector Regression
Sum	Weighted Summation
Var	Weighted Variance
VASP	Vienna Ab initio Simulation Package

LIST OF SYMBOLS

Al	Aluminium
Atm	Atmospheric pressure
$A_wA_wB_wB$	Binary compound
a_m	Parameter of the m^{th} subtree
Ca	Calcium
Co	Cobalt
d_{min}	Tree depth
E	Experience
eV	Electron volt
E_f	Formation energy
$f_{A,i}$	Element feature level of a binary compound
f_{ave}	Weighted average
$f_{B,i}$	Element feature level of a binary compound
Fe	Iron
f_{gmean}	Geometric mean
f_{hmean}	Harmonic mean
f_{max}	Maximum pooling
f_{min}	Minimum pooling
f_{sum}	Weighted summation
f_{var}	Weighted Variance
$F(x)$	Prediction function
$f(\theta)$	Random vector
g/cm^3	Gram per cubic centimetre

GPa	GigaPascal
$h(x; a_m)$	M^{th} subtree
l_j	Sample set of leaf j
l_l	Samples of the left branch
l_r	Samples of the right branch
K	Potassium
K	Kelvin
$L(y, F(x))$	Loss function
Li	Lithium
Mg	Magnesium
Mn	Manganese
μ	Mean
N	Sample space
Na	Sodium
Ni	Nickel
n_{\min}	Number of samples split from a node
O	Oxygen
P	Performance
α	Random variable
R^2	Regression score
σ	Standard deviation
σ^2	Variance
T	Tasks
$\text{Var}(x)$	Variance

v	Step size
w_A^* and w_B^*	Normalized composition
$X_{i,j}$	Input variables
y_{im}	Negative gradient of the loss function
y_i	Mean y test value
\hat{y}_i	Actual y test value
\bar{y}_i	Testing set sample size
\tilde{y}_{im}	Training target

ELEMENTS LEVEL FEATURES

Feature	Description
atomic_number	Number of protons found in the nucleus of an atom
atomic_radius	Atomic radius
atomic_radius_rahm	Atomic radius by Rahm <i>et al</i>
atomic_volume	Atomic volume
atomic_weight	Mass of an atom
boiling_point	Boiling temperature
bulk_modulus	Bulk modulus
covalent_radius_cordero	Covalent radius by Cordero <i>et al</i>
covalent_radius_pyykko	Single bond covalent radius by Pyykko <i>et al</i>
covalent_radius_pyykko_double	Double covalent radius by Pyykko <i>et al</i>
covalent_radius_slater	Covalent radius by Slater
c6_gb	C ₆ dispersion coefficient in a.u
density	Density at 295 K
dipole_polarizability	Dipole polarizability
electron_affinity	Electron affinity
electron_negativity	Tendency of an atom to attract a shared pair of electrons
en_allen	Allen's scale of electronegativity
en_ghosh	Ghosh's scale of electronegativity
en_pauling	Pauling scale of electronegativity
evaporation_heat	Evaporation heat
first_ion_en	First ionisation energy
fusion_enthalpy	Fusion heat
gs_energy	DFT energy per atom (raw VASP value) of T=0 K
ground state	
gs_est_bcc_latcnt	Estimated BCC lattice parameter based on the DFT volume
gs_est_fcc_latcnt	Estimated FCC lattice parameter based on the

DFT volume	
gs_volume_per	DFT volume per atom of T=0 K ground state
heat_of_formation	Heat of formation
heat_capacity_mass	Mass specific heat capacity
heat_capacity_molar	Molar specific heat capacity
hhi_p	Herfindahl-Hirschman index (HHI) production
values	
hhi_r	Herfindahl-Hirschman index (HHI) reserves values
icsd_volume	Atom volume in ICSD database
lattice_constant	Physical dimension of unit cells in a crystal lattice
melting_point	Melting point
mendeleviev_number	Atom number in mendeleviev's periodic table
molar_volume	Molar volume
num_valence	Total valence electron
num_d_valence	Valence electron in d shell
num_f_valence	Valance electron in f shell
num_p_valence	Valence electron in p shell
num_unfilled	Total unfilled electron
num_f_unfilled	Unfilled electron in f shell
num_p_unfilled	Unfilled electron in p shell
period	Period in the periodic table
Polarizability	Ability to form instantaneous dipoles
sound_velocity	Speed of sound
specific_heat	Specific heat at 20 °C
thermal_conductivity	Thermal conductivity at 25 C
vdw_radius	Van der Waals radius

TABLE OF CONTENTS

CHAPTER 1	1
1 INTRODUCTION.....	1
1.1 General Introduction.....	1
1.2 Basic Components of Sodium-ion Batteries	2
1.3 Literature Review	4
1.4 The Basics of Machine Learning	6
1.4.1 The Definition.....	6
1.4.2 Machine Learning Methods.....	7
1.5 Research Problem	8
1.5.1 Problem Statement	8
1.5.2 Rationale.....	9
1.6 Purpose of the Study	9
1.6.1 Aim.....	9
1.6.2 Objectives.....	9
1.6.3 Research Questions	10
1.7 Structure of the Dissertation.....	10
CHAPTER 2.....	12
METHODOLOGY	12
2 MACHINE LEARNING STEPS	12
2.1 Sample Construction.....	12
2.1.1 Data Collection and Curation.....	12
2.1.2 Feature Engineering	13
2.1.2.1 Features/ Vector	13
2.1.2.2 Features Extraction	15
2.1.2.3 Feature Selection.....	15
2.1.2.4 Feature Selection Methods.....	16
2.1.2.5 Feature Importance Plot	17
2.1.2.6 Features Construction/Input Features Development	18
2.1.2.7 Procedure for Feature Development.....	18
2.1.2.8 Feature Learning	19
2.2 Hyperparameter Tuning.....	19
2.3 Machine Learning Formulation.....	22

2.4 Machine Learning Algorithms.....	23
2.4.1 Gradient Boosting Regressor.....	24
2.4.2 Light Gradient Boosting Machine	26
2.4.3 Random Forest Regressor.....	27
2.4.4 Extra Trees Regressor	28
2.4.5 Bayesian Ridge.....	29
2.4.6 Orthogonal Matching Pursuit	30
2.5 Evaluation Methods	30
2.5.1 Mean	30
2.5.2 Variance.....	31
2.5.3 Standard Deviation	31
2.5.4 Mean Square Error.....	32
2.5.5 Root Mean Square Error.....	32
2.5.6 Mean Absolute Error	33
2.5.7 Mean Absolute Percentage Error	33
2.5.8 Coefficient of Determination.....	33
CHAPTER 3.....	35
MODEL DEVELOPMENT	35
3.1 Building the model.....	35
3.1 Machine Learning Workflow	36
3.2 Dataset.....	37
3.2.1 Dataset for Selected Sodium Containing Materials	38
3.2.2 Dataset Split.....	41
3.2.3 Cross Validation	42
3.3 Model Selection.....	45
3.4 Model Tuning.....	47
3.5 Model Validation and Evaluation	48
CHAPTER 4.....	49
4 RESULTS AND DISCUSSION	49
4.1 Feature Engineering	49
4.1.1 Formation and Final Energy	49
4.1.2 Energy above Hull	51
4.1.3 Fermi Energy.....	53
4.1.4 Density.....	55
4.1.5 Band Gap.....	57

4.2 Model Selection.....	59
4.3 Model Tuning/Hyperparameter Optimization.....	64
4.3.1 Bayesian Ridge	65
4.3.2 Light Gradient Boosting Machine	66
4.3.3 Extra Trees Regressor	67
4.4 Model Performance.....	67
CHAPTER 5	73
CONCLUSION.....	73
5.1 Recommendations and Future Work.....	75
REFERENCES.....	76
APPENDIX.....	84
A.1 Papers Presented at Conferences	84
A.2 Code Details.....	85

LIST OF FIGURES

Figure 1.1: Schematic diagram of the Na-ion battery [6].	3
Figure 1.2: Two main categories of machine learning.	8
Figure 2.1: Steps considered for hyperparameter tuning [75].	21
Figure 3.1: Machine learning approach for property prediction [89].	35
Figure 3.2: Machine learning workflow [96].	37
Figure 3.3: Dataset split [96].	42
Figure 3.4: Illustration of 5-fold cross-validation process [103].	45
Figure 4.1: Correlation heatmap for the critical feature vectors selected by the Bayesian ridge model.	51
Figure 4.2: Important features selected by the light gradient boosting machine for energy above hull.	52
Figure 4.3: Optimal features selected by the extra trees regressor for energy above hull.	53
Figure 4.4: Important features for Fermi energy selected by light gradient boosting machine model.	54
Figure 4.5: Optimal features selected by the extra trees regressor for Fermi energy.	55
Figure 4.6: Important feature vectors selected by the extra tree regressor for density.	56
Figure 4.7: Optimal features selected by the light gradient boosting machine for density.	57
Figure 4.8: Important features selected by the light gradient boosting machine for band gap.	58
Figure 4.9: Key features selected by the extra trees regressor for predicting the band gap.	58
Figure 4.10: Measures of predicted formation energy (a) coefficient of determination and (b) mean square error as determined by various models.	59
Figure 4.11: Measures of predicted final energy (a) coefficient of determination and (b) mean square error as determined by various models.	60

Figure 4.12: Measures of predicted energy above hull (a) coefficient of determination and (b) mean square error as determined by various models...	61
Figure 4.13: Measures of predicted Fermi energy (a) coefficient of determination and (b) mean square error as determined by various models.	62
Figure 4.14: Measures of predicted band gap (a) coefficient of determination and (b) mean square error as determined by various models.....	63
Figure 4.15: Measures of predicted density (a) coefficient of determination and (b) mean square error as determined by various models.....	64
Figure 4.16: Parity plot of Bayesian ridge model predicted formation energy versus DFT formation energy showing model performance for training set (left) and test set (right).	68
Figure 4.17: The parity plot compares the predicted final energy with DFT final energy for the training set (left) and test set (right) model performance by the Bayesian ridge.....	69
Figure 4.18: Parity plot showing the performance of the extra trees regressor model predicted density versus the DFT density in the training set (left) and test set (right).....	69
Figure 4.19: The parity plot of LGBM predicted Fermi energy versus DFT Fermi energy model performance for training set (left) and test set (right).	70
Figure 4.20: The parity plot of LGBM predicted band gap versus DFT band gap model performance for training set (left) and test set (right).	71
Figure 4.21: The parity plot of LGBM predicted energy above hull versus DFT energy above hull model performance for training set (left) and test set (right).	72

APPENDIX

Figure A.1: Dataset extraction.	85
Figure A.2: Descriptor calculations.	86
Figure A.3: Pre-processed data.	87
Figure A.4: Performance of the models.	87
Figure A.5: Model building.	88
Figure A.6: Correlation heatmap for important features.....	88

Figure A.7: Bayesian ridge performance for the testing data.	89
Figure A.8: DFT and Machine learning formation energy comparison.....	89
Figure A.9: Scatter plot for testing data.	89
Figure A.10: Model predictions on the training data.....	90
Figure A.11: Scatter plot for the training data.	90
Figure A.12: Model tuning.	90
Figure A.13: The dataframe for model results.	91
Figure A.14: Performance of the model based on regression score.	91
Figure A.15: Performance of the model based on mean square error.	92
Figure A.16: Feature importance for light gradient boosting machine and extra trees regressor.	93

LIST OF TABLES

Table 3.1: Dataset for some of the selected sodium containing materials.	40
Table 4.1: Tuned Bayesian ridge model parameters from training set for formation energy and final energy.....	65
Table 4.2: Tuned light gradient boosting machine model parameters from training set for Fermi and energy above hull.....	66
Table 4.3: Tuned extra trees regressor model parameters from training set for density and band gap.	67

CHAPTER 1

1 INTRODUCTION

In this section, we discuss the transition from lithium-ion battery to sodium-ion batteries, machine learning basics, which include the definition of machine learning and the two main classes of machine learning which is supervised and unsupervised learning. The literature review, explaining how machine learning is applied in materials science. Research problem is also included with subsections problem statement and rationale. The purpose of the study, stating the aim, objectives, research questions and the research question approach. Lastly the dissertation structure is given.

1.1 General Introduction

The development of energy storage and conversion technologies is essential to mitigate renewable energy generation discontinuities and instability [1]. Presently, fossil fuels such as coal and oil are the main sources of energy world-wide, but burning of these fossil fuels emits carbon dioxide and other greenhouse gases, which are harmful to the atmosphere and subsequently causing global warming and climate changes. The current era necessitates the use of new, environmentally friendly sources of energy to reduce greenhouse gas emissions and ultimately benefit human health.

Batteries are among the important energy storage technologies required to overcome the world's dependent on fossil fuels while moving towards more efficient and environmentally friendly renewable energy sources. However, to successfully achieve this goal, a reliable energy storage technology, particularly battery sources system is required. In the past two decades, there has been tremendous advancements in lithium-ion batteries (LIBs) and solid-state electrochemistry research for application in portable electronics industry [2].

Lithium-ion batteries are utilized in a broad range of applications because of their life cycle, safety, high efficiency compared to other energy storage technologies. Despite their success, LIBs are expensive to produce due to limited lithium resources in the

Earth's crust and their relatively low power and energy densities. Moreover, large-scale application of LIB energy storage is not possible with their technology.

In recent decades, research interest on alkaline-ion batteries has developed rapidly because of their high energy density and environmental friendliness [1]. These batteries have gained a good reputation as alternatives to LIBs due to the high abundance of Na- and K-ions in the Earth's crust and seawater [3]–[5].

Sodium-ion battery (SIB) technology has gained the privilege of enabling advanced and more demanding applications for large-scale energy storage systems. However, compared to lithium-ion, sodium-ion has a larger radius and heavier mass, which causes the SIB to have a lower specific energy and shorter cycle life. These factors impact the storage reaction mechanism. Therefore, the structural difference between lithium- and sodium-ions storage reaction is insufficient to devise SIB electrode materials by simply duplicating LIB electrode materials. Hence, recent research efforts have been directed towards discovering new material and plausible reaction mechanism for SIB-electrodes with enhance overall battery performance on specific energy, cycling life, good cycling stability and high energy density.

The study aims to employ data-driven modelling to develop machine learning models that are capable of predicting the properties of sodium-ion battery cathode materials.

1.2 Basic Components of Sodium-ion Batteries

Sodium-ion battery consists of anode, cathode, electrolyte (non-aqueous/aqueous) and a separator. The sodium-ion is shuttle between positive cathode and negative anode during charging/discharging, operating the same way as lithium-ion battery. Typical components and working principle of sodium-ion battery are shown in Figure 1.1.

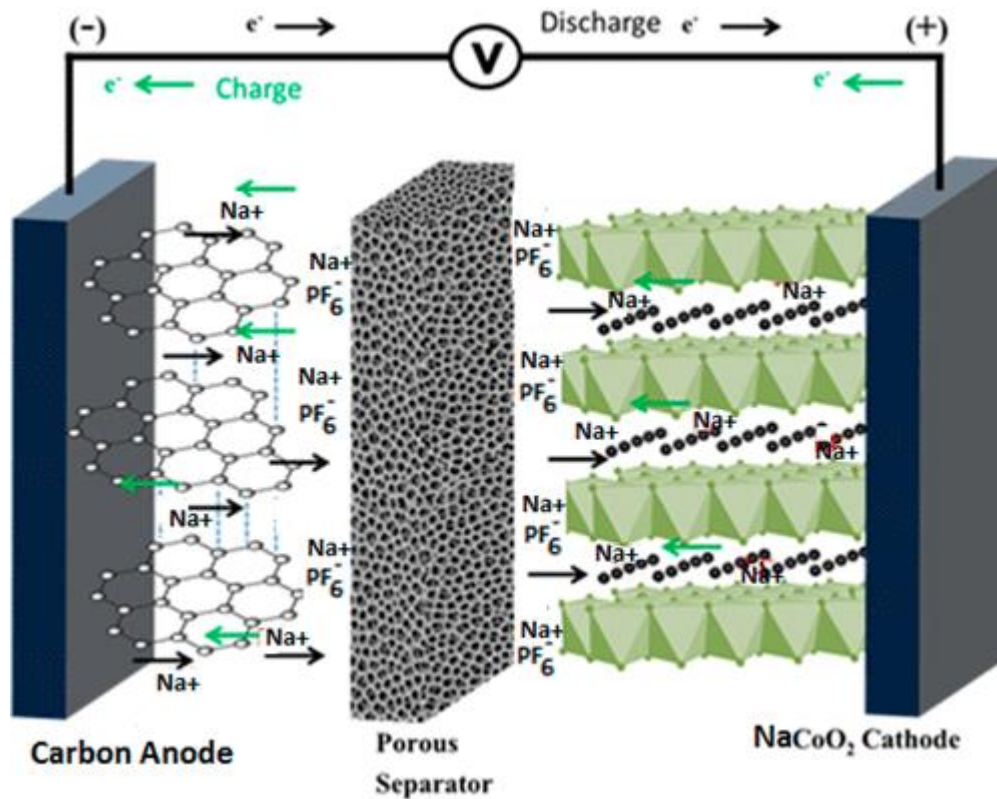


Figure 1.1: Schematic diagram of the Na-ion battery [6].

There are four main segments in a standard sodium-ion battery.

- i. Anode - negatively charged or reducing electrode, responsible for releasing electrons into an external circuit and oxidizing during an electrochemical reaction [7].
- ii. Cathode - a positive electrode, drawing electrons from an external circuit and reducing them through an electrochemical reaction [7].
- iii. Separator - a kind of polymeric membrane situated between the anode with a positive charge and the cathode with a negative charge [8].
- iv. Electrolyte - a transport for ions to move between the anode and cathode in a cell. Generally, they are assumed to be liquids, consisting of acid, salt, and a solvent such as water, enabling the transfer of ions. However, some batteries include solid electrolytes, which can conduct ions even when operating at room temperature [7].

1.3 Literature Review

The sodium-ion battery has shown promise in increasing energy storage capacity and safety; however, due to the high-voltage cathodes, its long-term cycling performance is limited, impacting the overall performance of SIBs. The biggest challenge is to develop new electrode materials that can easily promote intercalation/de-intercalation of sodium-ions in the electroactive substrate, as well as other components of the cell. In recent years, great efforts have been carried out to search for functional SIB electrode materials [9], [10], amongst those that are investigated are metal alloys, oxides, chalcogenides, phosphorus, and carbonaceous materials [11]–[16].

Quantum mechanical methods such as density functional theory (DFT) have been proven to be effective in predicting and discovering functional novel materials. Although DFT has proved to be useful in materials design and discovery, it is computationally expensive and difficult for the techniques to handle complicated material and their associated scientific challenges. With data-driven machine learning (ML) approaches, material discovery is now possible at an accelerated rate and with fewer computational resources.

Recent studies proved that combining density functional theory and machine learning approaches can speed up structure-property prediction and the discovery of new materials [17], [18]. Notably, the use of ML methods requires accessibility to a structured data, hence concerted efforts on developing materials databases. Among those that were recently developed, are DFT based electronic databases such as Automatic Flow for Materials Discovery (AFLOW) [19], Materials Project Database (MPD) [20], NOvel MAterials Discovery (NOMAD) [21] and Open Quantum Materials Database (OQMD) [22]. The databases were developed to mainly accelerate the application of ML in materials science. The goal is to develop new materials with enhanced or novel properties by employing advanced statistical methods on collected data sets. Stefano *et al.* [23] designed the high-throughput framework AFLOW which has been developing over the past decade. Ye *et al.* [24] developed the Materials Project Database, which contains DFT calculated results for most of the known inorganic materials. A DFT database called OQMD was developed by Saal *et al.* [25], to accelerate materials discovery and design rule extraction using informatics techniques. Oses *et al.* [26] illustrated that a combination of databases with ML tools

can be used predict thermodynamic formability modelling. Seko *et al.* [27] reported that nanomaterials can be characterized and designed by using a data-centric approach that integrates machine learning and DFT calculated data.

Relevant to this study, machine learning models and algorithms are increasingly applied in battery materials research, with superior time efficiency and high accuracy in property prediction [28]–[32]. Recent studies demonstrated how ML provide insights into the battery's operation and guide the rational design of electrodes and electrolytes [33]–[35]. For example, Kang *et al.* [36] used Neural Network (NN) algorithm to develop a black box battery model. In another study, Haq *et al.* [37] used Support Vector Regression (SVR) algorithm to improve the black box battery model accuracy and performance. Darbar *et al.* [38] used the Feedback Neural Network (FNN) to accurately determine the state of charge (SOC) value of the highly nonlinear nature of Na-ion batteries, using a higher cut-off voltage of -4.5 V Na⁺ with different current rates and cycle data, and achieved R² values of 0.97 to 0.99, respectively. Furthermore, Joshi *et al.* [32] used feature vectors derived from chemical properties and their basic components to develop machine learning algorithms. Using Deep Neural Networks (DNN), support vector machines (SVM), and Kernel Ridge Regression (KRR) algorithms, new 5000 electrode materials for Li, Na- and K-ion batteries were identified [32].

NaNi_{1/3}Mn_{1/3}Co_{1/3}O₂ cathode materials were used to model and optimize the manufacturing process of sodium-ion battery materials, resulting in a value better than that of conventional batteries [39]. Machine learning was recently used to predict the mechanical properties of sodium-ion solid state electrolyte, and the developed model performed well with a prediction accuracy (R² score) of 0.72 and 0.87, with mean absolute errors of 11.8 and 15.3 GPa for the shear and bulk modulus, respectively [40]. To date, there is no literature on using machine learning techniques to predict sodium-ion properties such as formation energy, final energy, Fermi energy, energy above hull, density and band gap.

1.4 The Basics of Machine Learning

1.4.1 The Definition

Arthur Samuel first proposed machine learning in 1959, defining it as a field of study that enables computers to learn and even refine their abilities without being explicitly programmed [1]. Machine learning algorithms are usually expressed as computer programs that can learn from experience (E) in terms of classes of tasks (T) and measures of performance (P) [1]. Therefore, ML is simply denoted as a function $\langle P, T, E \rangle$. Thus, performance on tasks in T, as measured by P, is expected to improve with experience E. In general terms, machine learning is a branch of artificial intelligence (AI) that demonstrates its ability to be applied accurately to classification, regression, and other activities involving large-scale, high-dimensional data [41].

The fundamental idea behind machine learning is to develop an algorithm that can take in input data and predict its output using statistical analysis. Machine learning is mainly useful where large amount (thousands) of data is available. Depending on the algorithm used for a particular task, machine learning can change from being simple to extremely complex. The ML algorithms can easily recognize trends and patterns, examine extensive amounts of data, and recognize particular patterns and trends which are not detectable by humans [42].

In relation to materials research, when ML models are used the structure-property relationships and materials discovery can be made quickly using a simple model, which allows the analysis or prediction [43]. By predicting novel materials or properties using existing data, ML offers a solution to the materials search challenges. Some of the data generated from machine learning, e.g. new materials and their associated properties be used to further enhance the ML models. The aim of the models that are built using machine learning techniques is to provide fairly precise predictions that are more economical and more productive than computational, experimental, or human-driven approaches.

1.4.2 Machine Learning Methods

The main classes of machine learning include supervised and unsupervised learning. In the case of supervised learning, an artificial intelligence system is presented with labelled and categorized data. The aim is to estimate the mapping function so well such that when the new input data X is accessible, one may predict the output data. Training data consist of a set of input values and a corresponding set of output values. Examples of supervised learning include regression and classification. During the monitored learning process, the training input x_i is fed to the learning system, which produces the output, y_i . Training is called the learning process and estimates the learner's parameters based on the observed ground-truth data. The test is used to evaluate the learning predictions about the data [44]. Supervised regression, clustering, and classification algorithms are used for the prediction of materials properties on the macro- and micro-levels.

In the case of unsupervised learning, an artificial intelligence system is presented with unlabelled, and uncategorized data. The output values are absent in the training data, and the aim is to recognize patterns in the input values it analyzes (without desired outputs). Examples of unsupervised learning include clustering and anomaly detection. In materials science, unsupervised probability estimation algorithms are mainly used for the discovery of new materials. Also, unsupervised learning can be applied to analyze compositional variations from combinatorial experiments, analyze micrographs, identify descriptors and dataset noise reduction. Shown in Figure 1.2 is a summarized description of supervised learning and unsupervised learning.

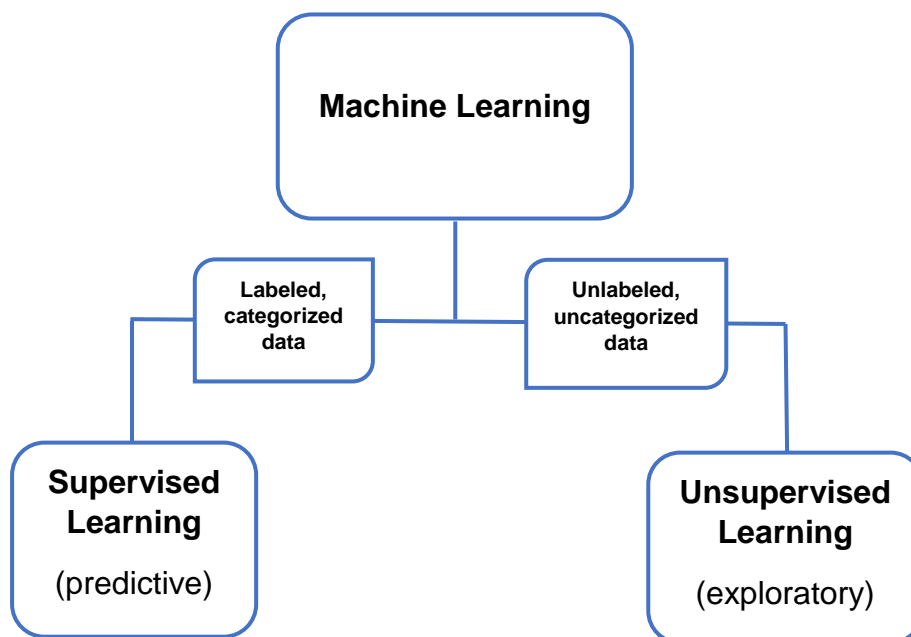


Figure 1.2: Two main categories of machine learning.

1.5 Research Problem

1.5.1 Problem Statement

Despite concerted efforts to develop novel materials for energy storage technologies, there is a continuous need for technologies that can push the limits on material properties. Lithium-ion batteries revolutionised the development of energy storage technologies and opened unprecedented solutions for portable electronic devices [45], [46]. Despite their success, LIBs have relatively low power and energy density, raising the challenge of transferability to large-scale applications [46], [47]. Furthermore, the scarcity of Li reserves in the Earth's crust, cost, and safety concerns raise uncertainties about their large-scale application [48]–[51]. All of these issues call for alternative technologies with better performance. Various metal-ion batteries, such as Na, Mg, Ca, K, and Al- ions, have been proposed. However, their technological developments are limited due to the lack of suitable electrode and electrolyte materials [47]–[49], [52]–[54] as well as difficulties in accurately screening their chemical and structural spaces [55]. Therefore, the study aims to employ data-driven modelling to

develop machine learning (ML) models that are capable of predicting the properties of sodium-ion battery (SIB) cathode materials.

1.5.2 Rationale

The discovery of new materials brings about immense progress in technological developments needed for human well-being. However, commonly used empirical trial-and-error and DFT-based methods cannot meet the current need and demand for new materials [56], they cannot be applied to systems with a large number of electrons. Hence, big-data and machine learning methods have recently emerged as a driving force for the materials research revolution because of their low computational cost and shorter development cycle. The era of big data and information is upon us. Daily, there is an unprecedented amount of data generated, shared, processed, and stored on the planet and machine learning methods can be used to assist in decision-making. ML is efficient in such a way that it can take only a few minutes to build a model and only a fraction of a second to make predictions [57]. It identifies patterns from large datasets, quickly discovers hidden laws, and extracts useful information faster compared to conventional computational simulations. As such, it is well suited for the discovery of new materials, and the prediction processes for the properties of materials are accelerated.

1.6 Purpose of the Study

1.6.1 Aim

The aim of this study is to apply machine learning regression techniques to a large amount of data to develop ML models capable of accurately predicting the properties of sodium-ion battery (SIB) cathode materials.

1.6.2 Objectives

The objectives of the study were the following.

- i. extract, process, and clean data provided from the Materials Project Database (data curation).

- ii. identify optimal features for property prediction (feature selection and engineering).
- iii. develop machine learning models (model development).
- iv. validate the developed models (model evaluation).
- v. predict properties of SIB materials (property prediction).

1.6.3 Research Questions

This study seeks to answer the following questions.

- i. Can machine learning models be used to accurately predict the properties of SIB materials?
- ii. What feature vectors (also known as descriptors) are suited for a particular property?
- iii. Which machine learning algorithms are optimal for predicting the properties of SIB materials?

1.7 Structure of the Dissertation

In this dissertation, machine learning models are used to predict sodium-ion battery cathode properties such as formation energy, Fermi energy, energy above hull, final energy, band gap, and density. The dissertation is partitioned into five chapters as follows:

Chapter 1: Presents the general introduction, followed by the fundamentals of machine learning, and literature review. Lastly, research problem which includes problem statement, rationale, aim, objectives, and research questions of this work is presented.

Chapter 2: Presents methodology covering the following topics, main classes of machine learning, machine learning flowchart, hyperparameter tuning, machine learning formulation, machine learning algorithms and evaluation methods.

Chapter 3: Discusses the model development, i.e., the procedure followed to build, develop, select, and validate the models.

Chapter 4: Discusses results for the target properties (i.e., formation energy, final energy, energy above hull, Fermi energy, density, and band gap), model performance and optimization.

Chapter 5: Presents summary of the findings, concluding remarks, suggestions on future research, and appendix. Papers presented in conferences and the code that was developed for this work are given in the Appendices.

CHAPTER 2

METHODOLOGY

In this chapter methodology on machine learning model development is discussed. The discussion covers; sample construction, hyperparameter tuning, machine learning algorithms and evaluation methods.

The most relevant question new researchers need to ask is whether their problems are likely to lend themselves to data-driven methods or not. There is no doubt that having available, reliable historical data, or at least efforts to generate it uniformly and systematically for a subset of critical cases, is essential to the adoption of machine learning. To be effective, data-driven methods should tackle (1) properties that are extremely difficult to compute or measure using traditional methods, (2) complex phenomena that cannot be solved directly by solving fundamental equations, or (3) questions that have no known solutions, so that surrogate models can be built [58]. As discussed in chapter 1, machine learning algorithms can be divided into two main categories based on their purpose.

2 MACHINE LEARNING STEPS

There are four key steps involved in machine learning process and are discussed in the next subsections.

2.1 Sample Construction

2.1.1 Data Collection and Curation

Sample construction is the first step in the machine learning model development process, it explains how the data is collected and curated. The original data is obtained and in this case from computational modelling simulations, and in other cases from experimental measurements. The data is normally incomplete and noisy; therefore, it is important to perform data cleaning during data processing of samples from the raw data. Errors must be identified and removed to prevent machine learning algorithms from being misled. Each sample obtained can be affected by several conditioning

factors, some of which are unrelated to the choice attributes. In this study the data was collected from the Materials Project Database, the data was cleaned by removing NaN (Not a Number) values and removing columns not adding value to the task at hand.

2.1.2 Feature Engineering

A featurization or feature engineering process read and transform raw data into the format that algorithms can understand [59]. It is more accurate for an algorithm to map input data to output data when the input data is more suitable. In feature engineering, raw data is used as input to algorithms for application. The performance of machine learning models is often constrained by it, as it is crucial for its functioning. Despite the fact that raw scientific data is often numerical, how the data is presented is often a key factor in machine learning.

2.1.2.1 Features/Vectors

An individual feature is a distinctive characteristic that is being considered by machine learning algorithms. Finding useful, discriminating, and independent features is a vital step in recognizing patterns, classifying data, and predicting outcomes. Features in material science must be capable of capturing all relevant information needed to distinguish between different atomic states [60]. Machine learning algorithms have trade-offs between the size of their feature vectors and their classification accuracy. Predicting and classifying with a large feature vector is significantly more complicated. On the other hand, in terms of classified objects/events, small feature vectors do not provide sufficient information about the objects/events. It is important to ensure that features are medium sized to capture the significance of the object/event and not overwhelm the user with too much information.

A feature vector is just as important as an algorithm when it comes to using machine learning and must at least be unique to predict a certain property. It is important to understand which of the relevant and highly correlated properties in the features are relevant for predicting the target property. It is vital to note that small feature vectors, in which the objects/events are not sufficiently described, can result in poor classification accuracy. It is just as important to choose the optimal algorithm as to choose the feature vector that represents the problem [61].

Here, we discuss how feature vectors are generated based on details of the chemical formula. The chemical formula is converted so that machine learning algorithms can be able to read it. A vector describing the formula in a meaningful way is required to provide information to the computer. This can be expressed as a vector in which each component represents a different chemical formula, for example (Na, Fe, O). This allows formulas to be expressed more easily, and in this case the compound Na_2FeO_3 is encoded as (2, 1, 3). Using the atomic weight of all elements within a compound, we generate a vector known as average atomic weight. Variance, geometric mean, and so on are calculated in the same manner. Alternatively, instead of relying on the elements, Composition-Based Feature Vectors (CBFVs) can be built by mixing atomic and elemental properties [62].

Chemical descriptors are used to construct machine learning features based on fundamental atomic properties, such as the chemical formula and atomic number. Chemical descriptors are available from the Xenonpy package [62] in 74 element-level properties that consist of 118 elements. Considering a binary compound $A_{w_A}B_{w_B}$, whose element-level features are denoted by $f_{A,i}$ and $f_{B,i}$ ($i = 1, \dots, 58$), the 290 compositional descriptors for $i = 1, \dots, 58$ are calculated as follows:

- i. Weighted average: $f_{ave,i} = w_A^* f_{A,i} + w_B^* f_{B,i}$ (2.1)

- ii. Weighted variance: $f_{var,i} = w_A^* (f_{A,i} - f_{ave,i})^2 + w_B^* (f_{B,i} - f_{ave,i})^2$ (2.2)

- iii. Geometric mean: $f_{gmean,i} = \frac{w_A + w_B}{\sqrt{[w_A + w_B] f_{A,i}^{w_A} * f_{B,i}^{w_B}}}$ (2.3)

- iv. Harmonic mean: $f_{hmean,i} = \frac{w_A + w_B}{\frac{1}{f_{A,i}^{w_A}} + \frac{1}{f_{B,i}^{w_B}}}$ (2.4)

- v. Max- pooling: $f_{max,i} = \max f_{A,i}, f_{B,i}$ (2.5)

- vi. Min- pooling: $f_{min,i} = \min f_{A,i}, f_{B,i}$ (2.6)

vii. Weighted sum: $f_{sum,i} = w_A f_{A,i} + w_B f_{B,i}$ (2.7)

where w_A^* and w_B^* represent normalized composition and sum up to one.

The calculated features can be obtained by obtaining a pandas.DataFrame object [63], [64].

2.1.2.2 Features Extraction

The initial set of raw data is reduced by feature extraction to more manageable groups for further processing by reducing the dimensionality. Among the characteristics of these large data sets is the large number of variables, which can require large amounts of computing power. Feature extraction techniques select and combine variables into features to represent the original dataset while reducing the amount of data that needs to be processed accurately and completely. When it comes to reducing processing resources, feature extraction can be very useful, since it doesn't dilute or omit any important information.

The purpose of feature extraction is to transform material space into descriptor space, i.e., input variables $X_{i,j}$. According to a particular application scenario, the number of $X_{i,j}$ is different. However, as the number of independent variables increases, the selection of features and computational load will be more complicated. Most existing technologies for extracting features for energy materials rely on human judgment. Their primary aim is to assess the importance and correlation of the extracted features.

2.1.2.3 Feature Selection

A feature selection technique involves selecting the most relevant input features for predicting a specific target variable. It is important to avoid irrelevant and redundant input variables since they can distract and mislead learning algorithms, possibly resulting in poor predictions [65]. A feature attribute is used in order to find the least influential features. Scikit learn [66] machine learning module was used for analysis of the regressor classes. The calculation of out of bag errors calculated the importance of features. The least important features were removed, the regressor was retrained

and it turned out that most of the features could be eliminated without significantly affecting the predictive power of the model.

Machine learning algorithms train faster when features are adequately selected, it simplifies the model and makes it easier to interpret. If the right subset is selected, the accuracy of the model is improved, in addition overfitting is minimized. It is important to use the correct feature selection method to see the subset of attributes used in the final simulation [67]. In this study, correlations and importance of selected features were determined by visible mapping [68]. A broader field of assessing the relative importance of input features is called feature importance. Many model-based methods exist, and their results can be used to aid model interpretation, dataset interpretation, or feature selection for modeling. The features should be uncorrelated because many correlated features can hinder the efficiency and accuracy of the model. In such cases, it is necessary to further select features to avoid the curse of dimensionality.

2.1.2.4 Feature Selection Methods

There are several methods used in feature selection and are briefly explained below:

- i. **Filter methods** - each feature is scored based on a statistical measure. A feature is ranked by its score and is either kept or removed from the dataset based on the score. It is common for methods to be univariate, describing features independently or with respect to the dependent variable. The filter method is used as a preprocessing step and the features are individually selected by machine learning algorithms; instead, feature selection is based on the results of various statistical tests that relate them to outcome variables. Chi-square test, information gain, and correlation coefficient score are some of the filter methods that are commonly used [69].
- ii. **Wrapper Methods** - trains the model with a subset of features. A model is trained using subsets of data. As a result of the model's output, features are added to retrain the model. Subsets are formed using a greedy approach, and accuracy is evaluated for all combinations of features. The computational cost of these methods is generally high. A wrapper method can be used to select

features forward, eliminate features backward, or recursively eliminate features.

- iii. **Forward Selection** - involves starting with no features and iterating through them one by one. In each iteration, the feature that best improves the model is added.
- iv. **Backward Elimination** - all features are analyzed at the outset, with the least significant features being removed at each iteration so that the model performs better. This procedure is repeated until the model no longer improves.
- v. **Recursive Elimination** - involves finding the optimal subset of features [70]. As a first step, the model is built based on all features and the importance of each feature is calculated. On the basis of model evaluation metrics (for example, RMSE, accuracy, and Kappa), the features are then ranked-ordered and the ones with the least importance are removed.
- vi. **Embedded/ intrinsic methods** - combine the benefits of both wrapper and filter methods, while at the same time maintaining a reasonable computational cost. As part of the embedded method, each iteration of the training process is taken care of, and the features that contribute most to the training process are selected [71].

In this study the filter method was used as a pre-processing step to select optimal features for each target property.

2.1.2.5 Feature Importance Plot

Feature importance plot is a graphical representation showing the ranking of features on how each feature contributes to the model. Features are plotted against their relative importance, that is the percentage importance of the most important feature. The term 'Feature Importance' describes a technique for computing scores for all input features for a given model. These values denote the "importance" of each feature. A high score indicates that the particular feature has a greater impact on the model used

to predict the particular variable/property. Descriptor imports can be obtained from model importance so that it can be determined which descriptors are most important for effective predictions. There are several reasons why feature importance is so vital:

- i. Data understanding
- ii. Model improvement
- iii. Model interpretability

2.1.2.6 Features Construction/Input Features Development

From the composition of the electrode materials, descriptive variables were created. These descriptors should be easily accessible, or easily measured, without requiring computer simulation. We create a mathematical description of the composition based on the atomic properties of the constituent elements. Sum, average, and variance of atomic weight, ionic radius, electronegativity, etc. obtained in advance from atomic properties [72]. For input feature development compositional descriptors were used from Xenonpy [62].

- Weighted average (abbr: ave)
- Weighted variance (abbr: var)
- Geometric mean (abbr: gmean)
- Harmonic mean (abbr: hmean)
- Max-pooling (abbr: max)
- Min-pooling (abbr: min)
- Weighted sum (abbr: sum)

2.1.2.7 Procedure for Feature Development

Based on formula $X_x Y_y Z_z$ in which X, Y, and Z share the j property. Average, sum, and variance are computed using the formulas below:

$$Average = \frac{xX_j}{x+y+z} + \frac{yY_j}{x+y+z} + \frac{zZ_j}{x+y+z} \quad (2.8)$$

$$Sum = xXj + yYj + zZj \quad (2.9)$$

$$Variance = \frac{(X - V)^2 + (Y - V)^2 + (Z - V)^2}{N} \quad (2.10)$$

where V is the number of elements.

$$V = \frac{Xj + Yj + Zj}{N} \quad (2.11)$$

2.1.2.8 Feature Learning

In machine learning, a system that automatically discovers the representations needed to detect or classify features from raw data is known as feature learning or representation learning [73]. This technique involves transforming raw data into representations that can be effectively used for machine learning tasks or features. It eliminates the manual feature engineering that would otherwise be required and allows machines to learn both specific tasks (using features) and the features themselves. The main reason for feature learning is that ML tasks often require inputs that is both mathematically and computationally convenient. Raw data must be transformed into useful features or representations. In order to automate and generalize these processes, efficient feature learning techniques are required. Features are learned with labelled input data. In this work, to give the model a useful angle about the important properties of the data, we created a mathematical representation of the data.

2.2 Hyperparameter Tuning

The tuning process is an effort to improve a model's performance without generating high variance or overfitting. All machine learning systems have hyperparameters, and the main goal of automated machine learning (AutoML) is to automatically set these hyperparameters as much as possible to maximize performance [74]. By choosing appropriate hyperparameters in machine learning, algorithms can learn models with significant differences in performance from the same training dataset.

It is imperative to explore an optimal combination of hyperparameters that minimizes the loss function for optimizing the model. The objective is to obtain better results by minimizing the loss function. Having enough resources, an optimized learning algorithm should achieve performance that is arbitrarily close to optimal. The interplay between random search and more complex optimization techniques, therefore, is an effective way to ensure a minimal rate of convergence, as well as to enhance model-based search [75]. It is also useful to perform a random search to start the search process, as it explores the entire configuration space and can often find settings that are reasonably performant. Figure 2.1 depicts the steps for hyperparameter tuning which is explained in detail in the following discussion.

As first step, data is loaded through the data loading process, which is the process by which source data is loaded from a file, folder, or application to a database or similar application. The relevant data is pre-processed, then hyperparameters are optimized using cross-validation. Accordingly, the tuned algorithm is fit to the training data, which comprised 70% of the data in this study, and finally learned model is applied to the test set which comprised of 30% of the data. The second step is hyperparameterization. As part of the hyperparameter optimization procedure, grid search cross validation is performed to determine the optimal parameters for each model. Performing a grid search is the standard method of tuning hyperparameters. Grid search validates the model for each hyperparameter value specified in the grid. The parameters are initially defined in terms of their range, and this is done to identify the minimum and maximum values.

There are certain parameters whose values are optimal within certain ranges. The grid is rendered more detailed according to the range. For the split to be optimized, two parameters need to be adjusted, number of trees (`n_estimator`) and number of features (`max_features`).

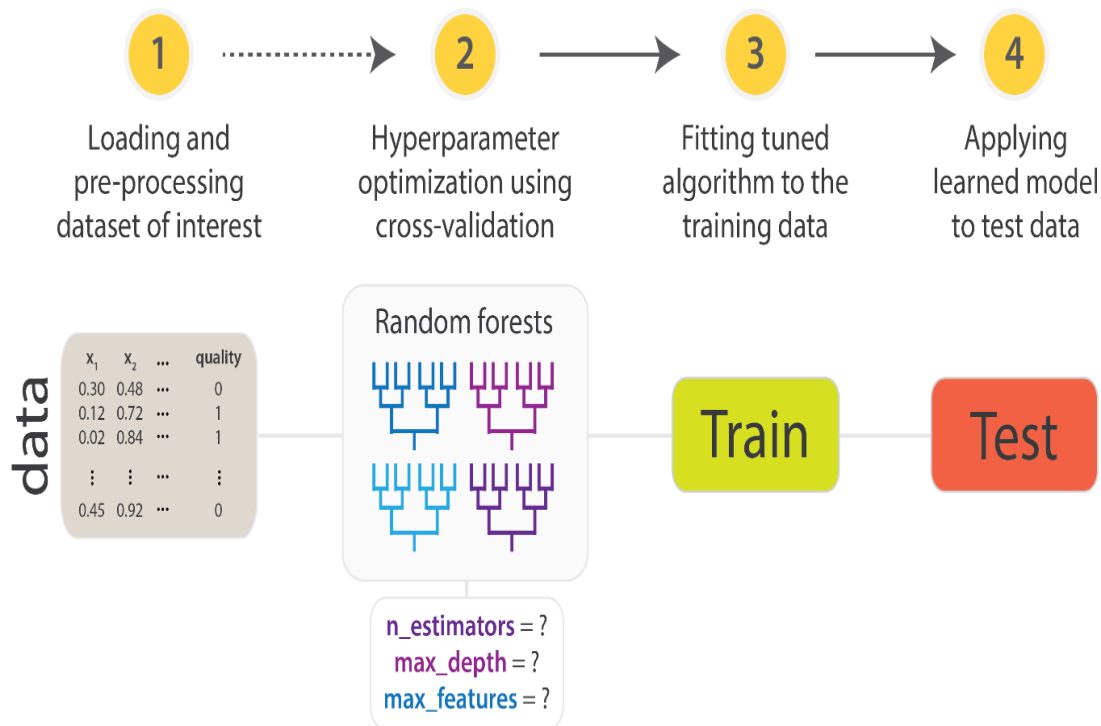


Figure 2.1: Steps considered for hyperparameter tuning [76].

The grid can be built by adjusting two parameters: $n_estimator$, which determines how many trees are estimated, and $max_features$, which specifies how many features are predicted. The grid could be constructed as follows in the first instance:

- $max_features$: [1, 0.8, 0.4, 0.3, 0.2]
- $n_estimators$: [100,150, 200, 250, 300,1000]

Scikit-learn [66] was used to train each model in Python [77]. Scikit-learn is a Python library used to build state-of-the-art machine learning algorithms. Each regression algorithm is designed to build an automatic ML model for each dataset that selects the best regression algorithm based on the prediction accuracy. Coding involves importing, reading datasets, and understanding target variables. Output variables are analyzed according to their distribution [78].

The determination of hyperparameters is often carried out via cross validation (CV) [79]. A popular cross-validation method is known as K-fold cross-validation. Suppose we have a hyperparameter, where the training dataset is divided into K-folds. As a training dataset, the remaining K-1 folds is used to learn the model parameters and test the model's performance on that fold. After averaging the results, the remaining

K-fold is used to test the model's accuracy. In a cross-validation process, the hyperparameter is determined in such a way that the average performance is maximized [80].

Decision Tree (DT) and K -Nearest Neighbour (KNN) are optimized using 10-fold cross validation. Initially, it divides the dataset into 10 sections of similar size and variable distribution, by choosing one section as the test set and the rest as the training set for 10 rounds. So, 10 sets of training and 10 test sets are taken and used 10 times for training and validation. The final evaluation results are based on the average performance of the test set over 10 rounds of experiments. Optimal hyperparameters are those that yield the highest performance [81].

Model development and model evaluation, evaluation is step 3 and 4 respectively. These two steps are explained under chapter 3.

2.3 Machine Learning Formulation

For materials science problems that require ML, it is important to build a machine learning system. In general, these machine learning systems follow this paradigm:

$$\textit{Goal} + \textit{Sample} + \textit{Algorithm} = \textit{Model} \quad (2.12)$$

The goal is the problem at hand, it usually takes the form of an objective function. Typically, a sample is a subset of the population which is selected according to some predetermined rules [82]. The raw data is transformed into the sample through data pre-processing, such as data cleaning and feature engineering. An algorithm, which includes machine learning algorithms and model optimization algorithms. In machine learning, Support Vector Machines (SVM), Decision Trees (DT), and Artificial Neural Networks (ANN) algorithms are most commonly used. The most important model optimization algorithms include Genetic Algorithms (GA), Simulated Annealing Algorithms (SAA), and Particle Swarm Optimization algorithms (PSO). Model is a mathematical formulation of the system and referred to as the algorithm built upon the sample.

The machine learning formulation is imperative in materials science, it helps identifying the task at hand, data type and availability, it also helps in determining which algorithms will be optimal for the task.

2.4 Machine Learning Algorithms

A key component of ML is its algorithms, and these can generally be grouped into traditional ML algorithms primarily based on statistics and Neural Networks (NN). Most classical machine learning algorithms include Bayesian, Decision Trees, Support Vector Machines, Cluster Analysis (CA), and Random Forests (RF). In Python, Scikit-learn contains most of the classical ML algorithms, which can be accessed easily [83]. Model selection depends on the problem to solve [84], as such there is no single algorithm that fits all. Every time a new situation arises, cognitive system reaches for the past experience, for guidance. Considering past scenarios, decisions, and experience, better decision making in the future can be done.

A machine-learning algorithm creates a relationship between a dependent attribute and an independent one and then predicts new input data outcomes based on that relationship [85]. ML regression algorithms are used to predict the value of the target variable based on a set of independent variables, also called features. Basically, the algorithm is trained to predict over time, using examples to verify the predictions. After that, the algorithm modifies its structure to minimize errors [86].

The term clustering algorithm refers to algorithms that cluster data sets without any prior knowledge of them. When data is clustered based on density, high-density areas are surrounded by low-density areas. Data points are classified into clusters based on their probability of belonging to a particular cluster with distribution-based clustering.

ML methods work by building a model (which can be seen as a function) that transforms inputs (also called descriptors, describing the materials) into outputs (usually a material property, such as formation energy, final energy, Fermi energy, energy above hull, density, and band gap as selected in this study). The descriptors should be as close to the targets as possible (the actual values of the material properties). To analyze and predict data, machine learning algorithms create statistical models by learning from data. Without being specifically programmed. Using

regression module *pycaret.regression import ** [87], 17 machine learning regression models are imported to set up the pycaret environment and target properties are estimated, in this study the target properties were formation energy, final energy, Fermi energy, energy above hull, density and band gap.

To analyze our data, six regression models: were considered, namely light gradient boosting machine (LGBM), gradient boosting regressor (GBR), extra trees regressor (ETR), random forest regressor (RFR), Bayesian ridge (BR), and orthogonal matching pursuit (OMP). Below is the command that generates the regressor algorithm for our ML models.

```
from pycaret import *  
from pycaret.regression import *
```

The models considered in our study are briefly explained in the next subsections.

2.4.1 Gradient Boosting Regressor

Gradient Boosting Regressor (GBR) is a supervised learning method used for regression problems. Gradient boosting was developed, by Jerome H. Friedman, based on Leo Breiman's observation that boosting may be interpreted as an optimization algorithm [88]–[90]. In practice, it is a powerful machine learning tool. This is a method that uses the loss function of the base models, usually decision trees to give a predictive model in the form of an ensemble of weak predictive models. Training data overfits quickly with this technique. Through regularization techniques, it penalizes different parts of the algorithm, reducing overfitting and improving performance.

Using the gradient boosting method with regression trees as weak learners, the following is the basic principle:

For the sample space $N = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$. The aim is to find the prediction function $F(x)$ minimizing the loss function $L(y, F(x))$ among all x-y mappings.

The prediction function is then represented as:

$$F(X) = \sum_{m=1}^M \beta_m h(x; a_m) \quad (2.13)$$

where $h(x; a_m)$ is the m^{th} subtree of the weak learner, $m = 1, 2, \dots, M$; a_m is the parameter for the m^{th} subtree, β_m is the subtree weight. If the prediction function produced by the first m training weak learners is $F_m(x)$, the optimization problem is equivalent to finding the parameters of a new subtree (β_m, a_m) ,

$$(\beta_m, a_m) = \operatorname{argmin} \sum_{i=1}^N L(y_i, F_{m-1}(X_i) + \beta_m h(x; a_m)) \quad (2.14)$$

For the above conditions (2.13) and (2.14), gradient boosting is updated as follows:

The first step is to initialize the regression tree:

$$F_0 = \operatorname{argmin} \sum_{i=1}^N L(y_i, h_0(x; \alpha_m)) \quad (2.15)$$

The second step: for $m = 1, 2, 3 \dots M$, the negative gradient of the loss function is:

$$y_{im} = - \left[\frac{\partial L(y_i, F(X_i))}{\partial F(X_i)} \right]_{F(X)=F_{m-1}(X)} \quad (2.16)$$

Fit a new subtree with y_{im} as the training target, and determine the leaf node area by calculating the subtree parameters:

$$a_m = \operatorname{argmin} \sum_{i=1}^N [Y_{im} - \beta_m h(x_i, a_m)]^2 \quad (2.17)$$

where,

$$\beta_m = \operatorname{argmin} \sum_{i=1}^N L(y_i, F_{m-1}(X_i) + \beta_m h(x; a_m)) \quad (2.18)$$

Updating the prediction function takes the form:

$$F_m(x) = F_{m-1}(x) + \nu \beta_m h(x; a_m) \quad (2.19)$$

where ν is the step size used to control the learning rate. To achieve the required prediction accuracy, m must be set to a smaller value; because, if the m is set too large, it may be more difficult to achieve the required prediction accuracy.

2.4.2 Light Gradient Boosting Machine

Light Gradient Boosting Machine (LGBM) is an open-source library that gives effective and compelling uses of the algorithm. It takes gradient boosting one step further by adding an automatic feature selection method as well as boosting examples with large gradients. In some cases, this can result in a remarkable improvement in coaching performance and predictability. Due to this, light gradient boosting machine has become one of the most popular algorithms in machine learning competitions for the analysis of tabular data for regression and classification prediction modelling tasks.

The LGBM method has high prediction accuracy, a fast computation rate and a capability of minimizing overfitting issues relative to other methods, it has been widely used in numerous fields [91]. Two novel techniques used in the LGBM algorithm, are of gradient-based one-side sampling and exclusive feature bundling.

Given the supervised training dataset: $X = \{(x_i, y_i)\}_{i=1}^n$, LGBM aims to find an approximation $\hat{f}(x)$ to a certain function $f^*(x)$ that minimizes the expected values of a specific loss function $L(y, f(x))$ as follows:

$$f(x) = \operatorname{argmin}_{f \in E_{y,x}} L(y, f(x)) \quad (2.20)$$

The LGBM integrates several T-regression trees $\sum_{t=1}^T f_t(X)$ to estimate the final model, which is;

$$f_T(X) = \sum_{t=1}^T f_t(X) \quad (2.21)$$

if J , q , and w represent the number of leaves, the decision rules of the tree, and the sample weight of leaf nodes, respectively, the regression trees can be expressed as $w_q(x)$, $q \in \{1, 2, \dots, J\}$ and it is possible to train the LGBM in an additive form at t .

$$\Gamma_t \cong \sum_{t=1}^N L(y_i, f_{t-1}(x_i) + f_t(x_i)) \quad (2.22)$$

Newton's method is used to rapidly approximate the objective function in LGBM. Removing the constant expression from the above equation, the objective function is reduced to:

$$\Gamma_t \cong \sum_{i=1}^N \left((g_i f_i(x_i) + \frac{1}{2} h_i f_t^2(x_i)) \right) \quad (2.23)$$

where g_i and h_i represent the first- and second-order gradient statistical results of the loss function, respectively. The equation (1.23) could then be transformed to:

$$\Gamma_t = \sum_{j=1}^J \left(\left(\sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right) \quad (2.24)$$

where, I_j represents the sample set of leaf j .

With respect to the tree structure $q(x)$, the optimum leaf weights of the leaf nodes w_j^* and extreme values of Γ_T are determined by equations (2.25) and (2.26), respectively:

$$w_j^* = - \frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \quad (2.25)$$

$$\Gamma_T^* = - \frac{1}{2} \sum_{j=1}^J \frac{\left(\sum_{i \in I_j} g_i \right)^2}{\sum_{i \in I_j} h_i + \lambda} \quad (2.26)$$

where is the weight function that measures the quality of the tree structure $q(x)$. Finally, the objective function is obtained by integrating the split:

$$G = \frac{1}{2} \left(\frac{\left(\sum_{i \in I_l} g_i \right)^2}{\sum_{i \in I_l} h_i + \lambda} + \frac{\left(\sum_{i \in I_r} g_i \right)^2}{\sum_{i \in I_r} h_i + \lambda} - \frac{\left(\sum_{i \in I} g_i \right)^2}{\sum_{i \in I} h_i + \lambda} \right) \quad (2.27)$$

where I_l and I_r are samples of the left and right branch, respectively.

2.4.3 Random Forest Regressor

Random Forests (RF) are ensembles of multiple decision trees that are trained in a random manner. We used RF in this study as a predictor i.e., random forest regressor (RFR), but it can also be applied as a classifier. In contrast to using a single decision tree, RF addresses the problem of biased variance using an ensemble of decision trees. Based on the features and the data points, each decision tree in the RF is trained on its own random features. The overall regression value is obtained by averaging the results of each tree [92]. This technique operates quickly over large datasets due to its high computational efficiency. Through feature bagging and tree bagging, the random forest technique performs regression tasks. Each node in the decision tree is split using the feature bagging technique.

For regression tasks, random forests are usually defined by growing trees based on a random vector $f(\theta)$, so that the predictor (i.e., a decision tree) $h(x, \theta)$ take the values instead of labels. In this case, the regression values will be real. Training samples are randomly selected from a distribution (Y, X) . Typically, the prediction task is usually defined as a Mean-Square Error (MSE) as follows:

$$E_{X,Y}(Y - h(X))^2 \quad (2.28)$$

where X and Y are the training samples and labels (values), respectively. After solving this equation, the random forest $h(x, \theta)$ can be used to predict each test sample x to get regression results [93].

2.4.4 Extra Trees Regressor

The term "Extra-Tree (ET) method" is short for extremely randomized trees. A great deal of variance in induced trees arises from choosing the optimal cut-point when considering input features (numerical). Therefore, randomizing tree building is important. A tree-based ensemble, the extra tree regression (ETR) algorithm is an extension of the random forest algorithm. A random subset of features is used to train each base estimator, like how random forests work [94].

Splitting occurs in each node of the decision tree with the feature bagging technique. Using the entire training dataset instead of the bagging step, the extra tree regressor trains on the decision trees. ETR algorithm increases the performance of the model since it is less susceptible to overfitting. In order to generate output features, input feature pairs are generated as shown by the equations (2.28) and (2.29):

$$F = \{f_n\}_{n=1}^N \quad (2.29)$$

$$B = \{B_n\}_{n=1}^N \quad (2.30)$$

In view of the fact that extra trees regressor performance is affected by factors such as tree number (N), number of samples split from a node (n_{min}), tree depth (d_{min}) and selection of attributes (K) [94]. Then, it is worth mentioning that the number of trees is directly related to computational time, and therefore a reasonable number of trees needs to be selected to optimise prediction performance and computational time. With

the increase in the number of training data sample, it is expected that the prediction accuracy of the model will increase. Since K is the number of randomly selected features at each node during the tree growing process, it determines the strength of variable selection process and for most regression problems.

2.4.5 Bayesian Ridge

Bayesian Ridge (BR) estimates a probabilistic model for regression problem. The prior for the coefficient w is given by a spherical Gaussian:

$$p(w|\lambda) = \mathcal{N}(w|\lambda^{-1}I_p) \quad (2.31)$$

where, w is the regularization parameter, and λ and I_p are the hyperparameters of the gamma prior distribution.

The priors over α and λ are chosen to be gamma distributions, which are conjugate priors to the Gaussian. The resulting model is known as Bayesian ridge regression, which is similar to classical Ridge Regression. In this work Bayesian ridge regression was used to predict the formation and final energy. The parameters w , α , and λ are jointly estimated during model fitting, and the regularization parameters α and λ are estimated by maximizing the log-marginal likelihood [95], [96].

To estimate regularization parameters, Bayesian ridge regression techniques can be applied. The regularization parameter is not determined in a hard-and-fast manner but is rather tailored to the data frame. The model's hyperparameters can be reconstructed using uninformative priors. Regularization with L2 in Ridge Regression (RR) and classification, it is equivalent to exactly finding the largest posterior estimate under a Gaussian prior with coefficient w with precision λ^{-1} . To generate a full probabilistic model, the output y is assumed to be Gaussian with respect to Xw :

$$p(y|X, w, \alpha) = \mathcal{N}(y|Xw, \alpha) \quad (2.32)$$

where α is treated as a random variable and is estimated from the data. As a Bayesian regression method, it is able to adjust to the data at hand and can be used to determine regularization parameters in the estimation process. Unfortunately, inference of the model can be slow [97].

2.4.6 Orthogonal Matching Pursuit

The goal of the Orthogonal Matching Pursuit (OMP) algorithm is to approximate the solution of one of two problems.

The sparsity constrained sparse coding problem given by:

$$\hat{\underline{\gamma}} = \underset{\underline{\gamma}}{\text{Argmin}} \left\| \underline{x} - D\underline{\gamma} \right\|_2^2 \text{ Subject To } \left\| \underline{\gamma} \right\|_0 \leq K \quad (2.33)$$

and the error- constrained sparse coding problem, given by:

$$\hat{\underline{\gamma}} = \underset{\underline{\gamma}}{\text{Argmin}} \left\| \underline{\gamma} \right\|_0 \text{ Subject To } \left\| \underline{x} - D\underline{\gamma} \right\|_2^2 \leq \epsilon \quad (2.34)$$

We assume, for simplicity, the columns of D are normalized to unit 2 lengths (although this constraint can be easily lifted). The greedy OMP algorithm selects the atom with the highest correlation to the current residue at each step. The signal is projected orthogonally to the selected span of atoms, the rest are recalculated, and the process is repeated [98].

2.5 Evaluation Methods

Metrics of evaluation explain the performance of a machine learning model. They are used to determine how well the model performs. Among the most important aspects of evaluation metrics is their ability to discriminate between results. Since the problem at hand is a regression then evaluation indices for regression are briefly explained in the next subsections.

2.5.1 Mean

An arithmetic mean or arithmetic average is a central value of a finite number set: specifically, the sum of the values divided by the number of values. A distribution's mean, or expected value, is one of its most recognized properties. The mean is denoted by the symbol μ , and is defined as follows:

$$\mu = \frac{\sum x}{n} \quad (2.35)$$

where $\sum x$ is the sum of all the observations and n is the total number of elements in the dataset.

2.5.2 Variance

The variance of a random variable is defined as the squared variation of the mean value, in probability and statistics. It is simply the average taken out of the standard deviation. Variance measures how spread out a set of numbers (randomly) are from their mean. Variance can be symbolized by σ^2 , s^2 , or $Var(X)$. The mathematical formulation is given by:

$$Var(X^2) = E[(X - \mu)^2] \quad (2.36)$$

from which

$$E[X] = \mu^2 + \sigma^2 \quad (2.37)$$

2.5.3 Standard Deviation

The standard deviation (std) provides a measure of the spread of values, it is simply stated as the observation that are measured through a given dataset. Essentially, standard deviation is calculated by square rooting the variance. Standard deviations for the sample and population are represented by the symbols σ and s , respectively. The standard deviation is expressed as:

$$\sigma = \sqrt{Var(X)} = \sqrt{\frac{\sum_{i=1}^N (x_i - \bar{x})^2}{N-1}} \quad (2.38)$$

where x_i is the terms given in the data, \bar{x} is the mean of the data and N is the total number of terms.

2.5.4 Mean Square Error

The Mean Squared Error (MSE) quantifies the difference between what is predicted and the actual result by calculating the average of the squares of the errors generated by an estimator. MSE can provide an indication of the quality of the model, and it is defined as:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (2.39)$$

where n is the number of data points, y_i the observed true value, and \hat{y}_i predicted value of the i -th sample. Zero mean square error shows that the model predicted 100% correct actual values, the model must achieve the mean square error closer to zero. The mean square error can never have negative values.

2.5.5 Root Mean Square Error

The root mean squared error (RMSE) is a commonly employed metric for quantifying the discrepancies between the values that are predicted by a model or estimator and the observed values. It is also referred to as the root mean square deviation and is expressed by the square root of the mean square error:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}} \quad (2.40)$$

In training the model, we want to minimize the RMSE as much as possible. Therefore, the smaller the RMSE, the better the model's performance. RMSE and standard deviation may appear similar, but they are different. Std measures the spread of data around a mean, whereas RMSE measures the difference between some values and the predictions associated with them. It can be interpreted as the degree to which data is concentrated around the line of best fit. In general, when the RMSE value is lower, the better a model fits a dataset. Std and RMSE converge as the mean error approaches 0 and n approaches infinity.

2.5.6 Mean Absolute Error

The mean absolute error (MAE) is a measure of the discrepancy between two identical observations, which signify a specific phenomenon. This metric evaluates the magnitude of the errors in a set of forecasts, disregarding the direction of the errors. MAE is used to gauge accuracy for continuous variables since it is a linear score, where the individual differences are equally taken into account for the overall mean. The mean absolute error is formulated as:

$$MAE = \frac{\sum_{i=1}^n |y_i - x_i|}{n} \quad (2.41)$$

where y_i is the prediction, x_i the true value and n is the total number of data points.

2.5.7 Mean Absolute Percentage Error

Mean Absolute Percentage Error (MAPE) is the most used error forecasting measure and is most effective when there are no extremes in the data (no zeros). A regression machine learning model's performance can be measured by the MAPE. Data scientists use this metric to assess the model's accuracy for a variety of use cases and datasets since it represents the error as a percentage. It is an easy metric for the end user to understand and enables a clear comparison between the model's accuracy. It is considered acceptable if the MAPE is less than 5%. MAPE is given by:

$$MAPE = \frac{1}{n} \sum_{t=1}^n \left| \frac{A_t - F_t}{A_t} \right| \quad (2.42)$$

where A_t is the actual value, F_t is the forecast value, n is the number of fitted points.

2.5.8 Coefficient of Determination

The coefficient of determination, also referred to as the regression score (R^2), is a statistical indicator that reflects the degree to which a dependent variable can be predicted based on the independent variable. It is measured using the following formula:

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y}_i)^2} \quad (2.43)$$

where y_i , \hat{y}_i , and \bar{y}_i are predicted values of machine learning mean y test value, actual y test value and testing set sample size. It is recommended that an R^2 reading of 0.6 or higher is adequate for a good reading, it indicates the model is fitting well and a measure below 0.4 shows a low level of correlation. When the coefficient of determination is equals to 1, this suggests that the model predicted 100% actual values that are correct. Therefore, the best model should have regression score closer to 1. To measure the accuracy of the models in this study, mean square error (MSE), and regression score square (R^2) will be used. The models are ranked according to their ability to predict the validation data that was not used during the training phase [78].

CHAPTER 3

MODEL DEVELOPMENT

In this section, the procedure for developing regression models based on the supervised machine learning method is discussed. This involves data collection and curation, model selection, model evaluation and validation.

3.1 Building the model

Model development is an iterative process in which many models are build and tested. It is basically a black box that links input and output data using either linear or non-linear functions. This method allows us to employ a sample of a desired function to search for the factors where a specific mapping function will best replicate the desired function. Models are built until they satisfy the desired criteria; this is often due to the fact that the model already in use may be ill-suited for reuse and not be fully understood.

Figure 3.1 shows the model development process, explaining how the machine learning algorithms work. The models work by performing a pre-processing step to generate a set of descriptive attributes as input features (X) and use known atomic properties to generate chemical and physical descriptors [92]. The true labels (Y) of the model during training are the properties computed by DFT, namely formation energy, final energy, Fermi energy, energy above hull, density and band gap are the true labels computed by density functional theory.

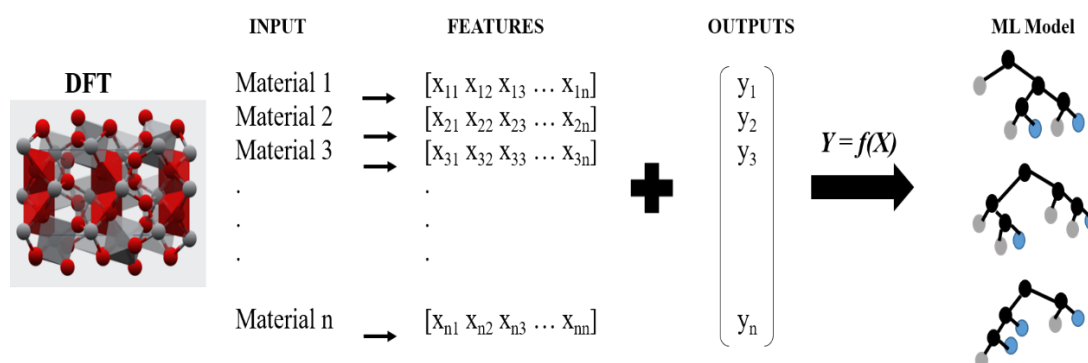


Figure 3.1: Machine learning approach for property prediction [92].

3.1 Machine Learning Workflow

Figure 3.2 shows the workflow that was followed in this study.

- Data collection and curation - the dataset containing 7397 SIB electrode materials was collected from the Materials Project Database. The data was cleaned (curated) by removing the duplicates of some materials and NaN not existing numbers. Also, for band gap prediction materials that have a band gap of zero, were eliminated, for better model performance.
- Feature engineering - important features were selected to train our machine learning models. Feature engineering involves creating new features from the original raw data by utilizing mathematical models. By using this mathematical representation, the relationship among features can be refined, and create few new features that accurately describe the sample data. The correlation heatmap and feature importance plot were used to select optimal features.
- Dataset split and model training - after the ML algorithm uses the training data set to create a model between the features and the objective function, we the data was randomly split into a training data set and a test data set. In this work the dataset was split into 70% train and 30% test set. The models were built and trained.
- Model selection - the best models were selected based on the performance accuracy. The evaluation indices selected for performance evaluation includes regression score and mean square error.
- Hyperparameter tuning - the selected models were fine-tuned. To boost performance and maintain an acceptable accuracy and computation cost of the model, the hyperparameters of the algorithm were adjusted.
- Model validation and evaluation - the models were validated by checking the performance accuracy on the train and test set. The validated model was then used to predict data properties [83]. For model evaluation the DFT calculated properties were compared with the corresponding machine learning predicted values.

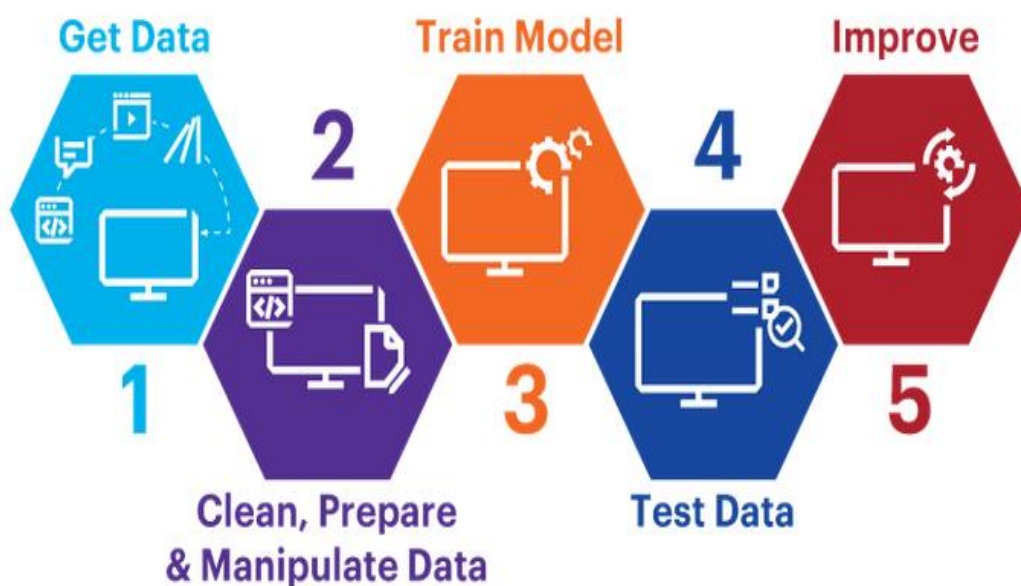


Figure 3.2: Machine learning workflow [99].

3.2 Dataset

Generally, a data set is a collection of information that corresponds to one or more database tables, with each column representing a variable, and each row resembles the given record for the information set in question. Basically, the data were organized in a certain model that helps to process the needed information. The training data was extracted from the Materials Project Database, containing a total of 7397 sodium-ion battery materials properties calculated using DFT, which was our input dataset. The DFT materials properties stored in materials project database was estimated and optimized by the Vienna Ab initio Simulation Package (VASP). Chemical formula, formation energy per atom, final energy per atom, Fermi energy, energy above hull, band gap, and density for every material are included in the extracted dataset. The meanings of these properties were detailed according to the explanation in the MPD glossary.

- i. **Pretty formula (chemical formula):** An expression in which the element set is normalized.
- ii. **Density:** Final relaxed density of the material, calculated bulk crystalline density.

- iii. **Formation energy per atom:** calculated formation energy from the elements normalized to per atom in the unit cell, computed formation energy at 0 K, 0 atm using a reference state of zero for the pure elements.
- iv. **Energy above hull (e above hull):** calculated energy at the convex hull of the structure. The energy in eV/atom at which this material decomposes into the most stable group of substances with this chemical composition. Stability is normally tested for all possible chemical combinations resulting in the composition of the material. A positive energy above the hull indicates that this material is unstable to decomposition. A negative energy above hull indicates that this material is stable. Zero energy above the hull indicates that this is the most stable material in its composition.
- v. **Band gap:** the band gap is defined as the energy difference in (eV) between the upper valence band and lower conduction band of insulator and semiconductor materials.
- vi. **Fermi energy:** energy difference between the highest and lowest occupied states of a non-interacting fermion system at absolute zero.
- vii. **Final energy per atom:** the total energy of the materials after structural relaxation.

3.2.1 Dataset for Selected Sodium Containing Materials

The data were accessed from the database via the Python Materials Genome (Pymatgen) application programming interface for Materials Project [20]. Our dataset is composed of predicted formation energy, Fermi energy, final energy, density, energy above hull, and band gap for a variety of sodium-ion batteries calculated using DFT. The following command was used to extract the materials properties from the Materials Project:

```
import json
import requests
data = {'criteria': { 'elements': { '$all': ['Na']}},
        'properties':
        ['pretty_formula', 'final_energy_per_atom', 'efermi', 'formation_energy_per_atom',
        'density', 'e_above_hull', 'band_gap']}
r = requests.post('https://materialsproject.org/rest/v2/query',
                  headers={'X-API-KEY': 'tXXXXXXXXXJ'},
                  data={k: json.dumps(v) for k,v in data.items()})
response_content = r.json()
train=pd.DataFrame(response_content['response'])
```

As shown in Table 3.1, the following dataset was obtained from Materials Database Project: chemical formula, formation energy per atom, Fermi energy, final energy per atom, density, energy above hull, and band gap.

After data cleaning, the original data reduced from 7397 data samples to 4063 sodium-ion data samples.

Table 3.1: Dataset for some of the selected sodium containing materials.

index	pretty_formula	formation_energy_per_atom	density	e_above_hull	band_gap	efermi	final_energy_per_atom
2475	NaCO3	-1.8031397877499997	2.0880278770726197	0.0	0.6661	-1.56350021	-6.458337634
2476	NaNiO2	-1.2335706431250006	4.609605859495142	0.018897787500000263	0.18389999999999995	3.70664806	-4.58841457875
2477	NaGaB4	-1.1261215920833336	3.207539219438662	0.0	3.48880000000000003	-1.78886251	-2.9407914395833337
2478	Na3Fe5O9	-1.8455208933088225	3.601101400129514	0.008413755882353158	0.0	1.23352658	-6.0829902082352945
2479	NaClO4	-1.0070405855433329	2.0232594149880088	0.111481495833333	4.59	-0.91410903	-4.265163228333333
2480	NaHoF4	-4.016335403293333	5.451083001750114	0.0	6.8599	-1.5661145	-5.945807471666666
2481	Na5V36O88	-2.3264682322093018	3.4248498420533604	0.03363404980619045	0.0	1.82183989	-7.338813181007752

3.2.2 Dataset Split

The dataset split is the process of dividing the original dataset into two sets (training set and test set) in order to coach ML models regardless of the data type used. The training set is used for model training, while the test set is used to validate the model. The low training data results in higher variances for parameter estimates and therefore it is essential that the data is separated in such a way that neither is simply too high nor too low, which is determined by the large amount of information generated.

For several problems, Joseph suggested a ratio of 70/30 or 80/20 (training/testing set) [100]. Previous studies reported that increasing the training dataset from 70% to 80% improved model performance and stability, and the testing performance could also be improved. However, testing performance exhibited an opposite trend when training size was increased from 80% to 90%. Overall, the size of the training set influenced the prediction ability of the ML models [101].

Initially, 90% of the data was used for training and 10% for testing, the models did not perform well during model selection process, so the split of the data was adjusted for the models to better fit the data. Second case involved 80% train set and 20% test set, the models performed better as compared to the first case of 90/10 splitting. In order to maximise and improve performance, we further checked how the data may fit our models with 70/30 train/test split and it was found to fit the data perfectly, hence 70/30 splitting used throughout the study for the predictions of the properties.

Below is the command for train/test split.

```
train_test_split = setup (data = X, target = 'property of interest', session_id=123, train_size = 0.70)
```

In splitting the 4063 dataset which was in a good format, readable by ML packages, for the target properties the transformed train set was about 2844.

As illustrated in Figure 3.3 dataset splitting shows that 70% of the data was used for the learning process or training of the models. The remaining 30% was used for the validation process.

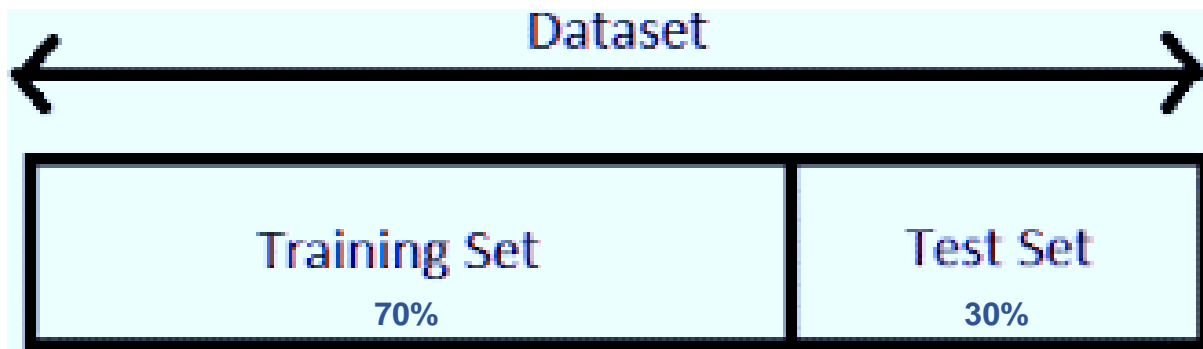


Figure 3.3: Dataset split [99].

3.2.3 Cross Validation

Cross-validation (CV) is a resampling technique for evaluating ML models on a restricted data. Statistical models such as regressions and classifications have been extensively validated using cross-validation. This method provides unbiased estimation and is easy to implement, as CV is well-known. There are several cross-validation techniques.

- i. Leave-p-out cross-validation - the leave-p-out cross-validation approach (LpOCV) involves utilizing p -observations for validation purposes and the rest of the available data for training. This procedure is repeated to divide the original sample into a collection of p observations and a collection of training observations in all possible ways.
- ii. Leave-one-out cross validation - in cross-validation, leave-one-out cross-validation (LOOCV) is an exhaustive technique. It is a type of LpOCV with the case of $p=1$. An n -row dataset is divided into the validation row and the training row ($n+1$). The second row will be selected for validation in the next iteration, and the rest for training. Similarly, the process is repeated until n steps or the desired number of operations have been completed. Both of these cross-validation techniques are types of exhaustive cross-validation. When a cross-validation method is exhaustive, it learns and tests in all possible ways. As a result, model biases may be low, and the computational time may be excessive. Yet, this technique is simple, straightforward, and can be implemented quickly [102].

- iii. Holdout cross-validation - with holdout cross-validation, a dataset is randomly split into a training and a validation data. The training data is usually bigger than the validation data. The training data is used to generate the model and the validation part is used to assess its accuracy. Generally, the more data used for the training, the better the model is. This kind of cross-validation method isolates a large amount of data from the training data.
- iv. Monte Carlo cross-validation - the Monte Carlo cross-validation technique, also known as random subsampling validation, splits the dataset into training and validation samples. The dataset was split, not in groups or folds, but in random ways in this case, for cross-validation. Analyzing the data determines the number of iterations. The average is then calculated over the splits. Monte Carlo cross-validation does not depend on the number of iterations or partitions how many trains and validations are done. There are also limitations to using this technique, for instance, the training or validation of some samples may not be possible, and it is not suitable for an imbalanced dataset.
- v. K-fold cross validation - the technique contains a parameter called K, which indicates how many groups the provided sample should be split into. Thus, the technique is commonly referred to as K-fold cross-validation. It is important to conduct cross validation because it gives more information about algorithm performance. As an alternative to Leave one out (LOO), K-fold cross-validation [103] has been proposed. Now, it is the simplest and most popular method of estimating generalization error. This method has the obvious advantage of requiring only K times calculations, which is far cheaper than (LOO) or Leave p out (LPO). It is imperative to keep in mind that this approach could be more prone to errors if the K-number is not large [104]. 5-fold or 10-fold cross-validation may be used for large-size datasets because the computational burden of leave-one out cross-validation is too heavy. The advantage of this technique includes low model bias, low time complexity and the entire dataset is used for both training and testing, however this method is not suitable for an imbalanced dataset.
- vi. Bootstrap cross-validation - due to the great variability in K-fold cross-validation when the scale of the sample data is small, researchers proposed bootstrap cross-validation (BCV). BCV has lower variability and fewer biases when the

scale of the samples is small compared to traditional validation methods [104], [105]. Although, it is important to note that BCV will cause the computation amount to rise sharply when the sample size is large.

There are a variety of cross validation methods, each with their own unique characteristics [105].

In this work we employed K-fold and hold-out cross validation. During model comparison, firstly the models were compared using cross validation 10-fold by default, and later 5-fold was used to compare the models. Based on this analysis, the 5-fold cross validation was effective in comparing the models. The reasons for changing the 10-to-5-fold cross validation were to improve the training time and the model accuracy. Under model comparison the score grid shows metrics MAE, MSE, RMSE, R^2 , and MAPE by default, which are already discussed in chapter 2. The hold-out cross validation was used during dataset splitting, where a larger number of the data was divided as the training set, which was used to train the models and the remaining dataset as the testing set used to validate the models.

For each fold, in Figure 3.4; the following tests were performed: In split 1, the first fold was used as validation (or holdout), while everything else was used as training data. A 30% holdout provides a model quality measurement. Besides the second fold, we take the data from split 2 (and train the model excluding the second fold). A second gauge of model quality was then obtained using the holdout set. This process was repeated one more time, with each fold being considered a holdout set.

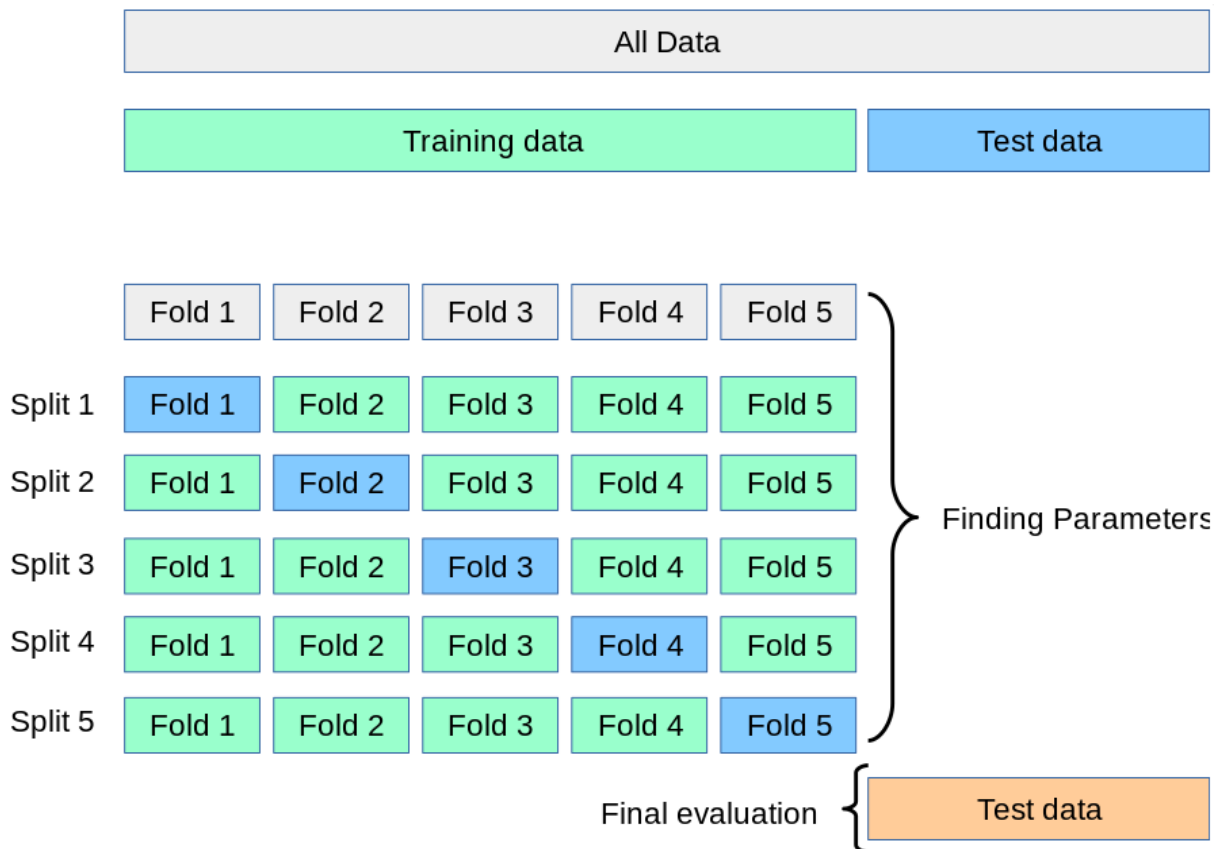


Figure 3.4: Illustration of 5-fold cross-validation process [106].

3.3 Model Selection

The process of selecting a scientific model from a collection of several models and data is called model selection [107]. It is always a good idea to consider prior data sets. Regardless of the method, the task may include arranging tests so that the data collected is well-suited to the matter of model selection. When inferring and learning from data, there are two main objectives. The first is for scientific discovery, understanding of the underlying mechanism for data generation, and interpretation of the data itself. Predicting future or unknown observations is another objective of data mining. A data scientist might not be specifically focused on obtaining an exact probabilistic representation of the data for their second objective. Alternatively, both directions may be of interest.

It is important to identify whether processed data have labels, i.e., target variables, before any further analysis. If labels exist in a dataset, supervised learning algorithms should be implemented, otherwise the issue should be classified as an unsupervised learning problem. The type of labels (discrete or continuous) can also influence the

choice of classification or regression algorithms. Afterwards, the data can be trained using a variety of algorithms and then select the best model depending on the prediction's accuracy. It is usually more difficult to interpret and hinders further understanding in the application domain to determine the model that can make the most accurate prediction from the more complicated feature space and decision rules. Selecting the right model and parameters for a particular task is the essence of model selection. Furthermore, it is generally accepted that the selection of an optimal algorithm should ideally consider the known physical properties of the descriptors and target [108].

Consequently, selecting a model can be categorized into two distinct paths: model selection for inference and model selection for prediction. It is essential to identify the most suitable model for the data, which will most likely give a reliable interpretation of the sources of uncertainty for scientific understanding [109]. In order to achieve this goal, it is crucial that the selected model does not depend too much on the sample size. Accordingly, the concept of selection consistency can be used to evaluate model selection, meaning that, if sufficiently many data samples are available, the most robust candidate will be selected consistently.

The second direction is to use a model that provides excellent prediction. Even when a model is chosen based on luck among a few close competitors, it can still perform well in predicting outcomes. If that's the case, selecting a model would be satisfactory for the second objective (forecasting), but the utilization of the chosen model for insight and interpretation would be completely unreliable and misleading. Moreover, for extremely complex models selected this way, even predictions based on data are just slightly different from those on which the selection was made may be unreasonable. [109].

The Bayesian Ridge (BR), Light Gradient Boosting Machine (LGBM), Random Forest Regressor (RFR), Gradient Boosting Regressor (GBR), Extra Random Tree (ETR), and Orthogonal Matching Pursuit (OMP) were tested to select the most suitable one using grid search strategies to increase efficiency. The Scikit library machine learning module and Python programming language were utilized to develop the models.

As part of this project, resampling methods were applied. Resampling techniques were used to measure the performance of a model (or, more precisely, the model

development process) by using data from outside the original sample. To achieve this, the dataset was divided into two parts - train and test - and a model was trained based on the train set, then evaluated on the test set. This procedure was repeated multiple times and the average performance over all of the trials was reported. In this method, out-of-sample data was used to estimate model performance under varying resampling methods, although each trial is not necessarily independent since the same data may appear in multiple training datasets or test datasets, depending on the chosen resampling method. Three frequent resampling model selection techniques include: Random train/test splits, Cross-Validation (K-fold, LOOCV, etc), and Bootstrap. The data was split randomly and cross validated using K folds.

3.4 Model Tuning

Model tuning is also known as hyperparameter optimization, it is the process of finding the optimal values of hyperparameters to maximize model performance. As soon as a model has been trained, the model is validated using unseen data that differ from the data in the training set. This validates the accuracy of the model. Bayesian ridge, light gradient boosting machine, and extra trees regressor were used to model the dataset. Modern classification and regression models can model complex relationships due to their high adaptability. Nevertheless, they are capable of exaggerating patterns that are not reproducible. When evaluating a model, without a methodological approach, the modeler will not discover the problem until the next set of samples is predicted. Several tests were run on improving the regression models by tuning hyperparameters until the best model producing the results reported. We eliminated the models that did not give accurate results to achieve better results. A number of hyperparameters were tuned to accomplish this and are further discussed in chapter 4. Choosing the best regularization is important, as small regularizations lead to complex models. It is also not effective to use larger regularization, since it makes the model less useful. A grid search technique was used to tune the models.

It is not necessary to understand the physical principles of material properties when performing machine learning for predicting material properties. The algorithm manages conversion of element-derived features into predictions using strictly statistical methods.

3.5 Model Validation and Evaluation

Estimating the quality of the model (model evaluation) is the step in the data-driven modeling process wherein the model's performance accuracy is assessed for both existing and unseen data. From the perspective of the intended uses of a model, validation is often defined as the process of determining whether the model accurately represents the real world. ML models must be evaluated in order to be effective. A model's performance metric refers to how well it performs on an unseen dataset. A machine learning model is evaluated by using different evaluation metrics discussed in section 2.5 to understand its performance as well as its strengths and weaknesses. As a part of the initial research phases, model evaluation is important for assessing a model's efficacy, as well as for monitoring it.

As a result, the model learned by the algorithm will not cover all situations, resulting in an actual predicted output that differs from the actual value of the sample. On the training and test sets, the ML model error is referred to as the training error, and on the new sample, the error is called the generalization error. ML models should have a small generalization error. However, the generalization error is difficult to calculate. Typically, the dataset is divided into three parts: a training set, a validation set, and a test set, and each is used to train the model, adjust the parameters, and calculate test error. To evaluating the accuracy of a machine learning model, the test error represents an approximate evaluation of the generalization error.

In order to make a model useful, it must address the correct problem, provide accurate information about the system, and be useful to users. In this study, the predictive accuracy of the model was evaluated by comparing the calculated DFT properties with the corresponding predicted ML values which is discussed in detail under chapter 4. The following accuracy measures were used to evaluate model performance: coefficient of determination/regression score (R^2) and mean squared error (MSE). These accuracy measures were described briefly in chapter 2.

CHAPTER 4

4 RESULTS AND DISCUSSION

This section presents the results covering the following key steps, feature engineering, model selection, model tuning and model performance. Model selection involves selecting the best models based on their performance, the best selected models are then fine-tuned to improve the performance score through a process called model tuning. Lastly model performance is discussed, where the density functional calculated properties are compared with the predicted machine learning target properties using parity plots.

4.1 Feature Engineering

As discussed in chapter 2, feature engineering is the transformation of raw data into a form that is more appropriate for use by a machine learning algorithm. In this study the optimal features were engineered using correlation heatmap and feature importance plots for various properties. It is an essential part of predicting material properties, as it allows for the optimization of the features used to train the model, which can increase the accuracy of the predictions. Feature selection is an important part of materials science which is informed by data. In particular, the features used to model a particular energy material must not only include the material's structural parameters, but also its performance characteristics. There are two ways of engineering features, which is correlation heatmap and feature importance. Features to calculate different properties are discussed below.

4.1.1 Formation and Final Energy

For formation and final energies, the correlation heatmap was used to engineer the features. A correlation heatmap is a graphical representation which displays the strength of the relationship between numerical variables. In correlation plots, variables are mapped against each other to determine the strength of their relationship.

Figure 4.1 shows 18 x18 matrix correlation heatmap ranging from -1 to 1, with squares representing the relationship between variables to predict both the formation and final energy of sodium containing materials. When the correlation is close to 1 or -1, it implies that the variables have a strong relationship. In addition, a value closer to zero indicates that the two variables are not linearly related. Since all the diagonals are 1 (fawn colour), there is a perfect correlation. A larger number and darker or lighter colour indicates a stronger correlation between the two variables. In this study, 18 descriptors were considered and evaluated to determine the important descriptors in predicting the energies. The average covalent radius (ave: covalent_radius_cordero, ave: covalent_radius_pyykko) and average single bond covalent radius (ave:c6_gb) were found to be the most important features, with feature correlation ranging between 0.82 and 0.99, respectively, as can be seen on the heatmap. Mphaka J. [110], established a feature set containing atomic and elemental properties and predicted formation energy of lithium-ion battery materials with 18 elemental descriptors. The Catboost model selected the maximum electron negativity, electron negativity Pauling, and average d valence as the major descriptors. In both cases, 18 key elemental descriptors were selected for the prediction of LIB and SIB formation energy. It follows that different descriptors predict similar properties differently based on the type of material.

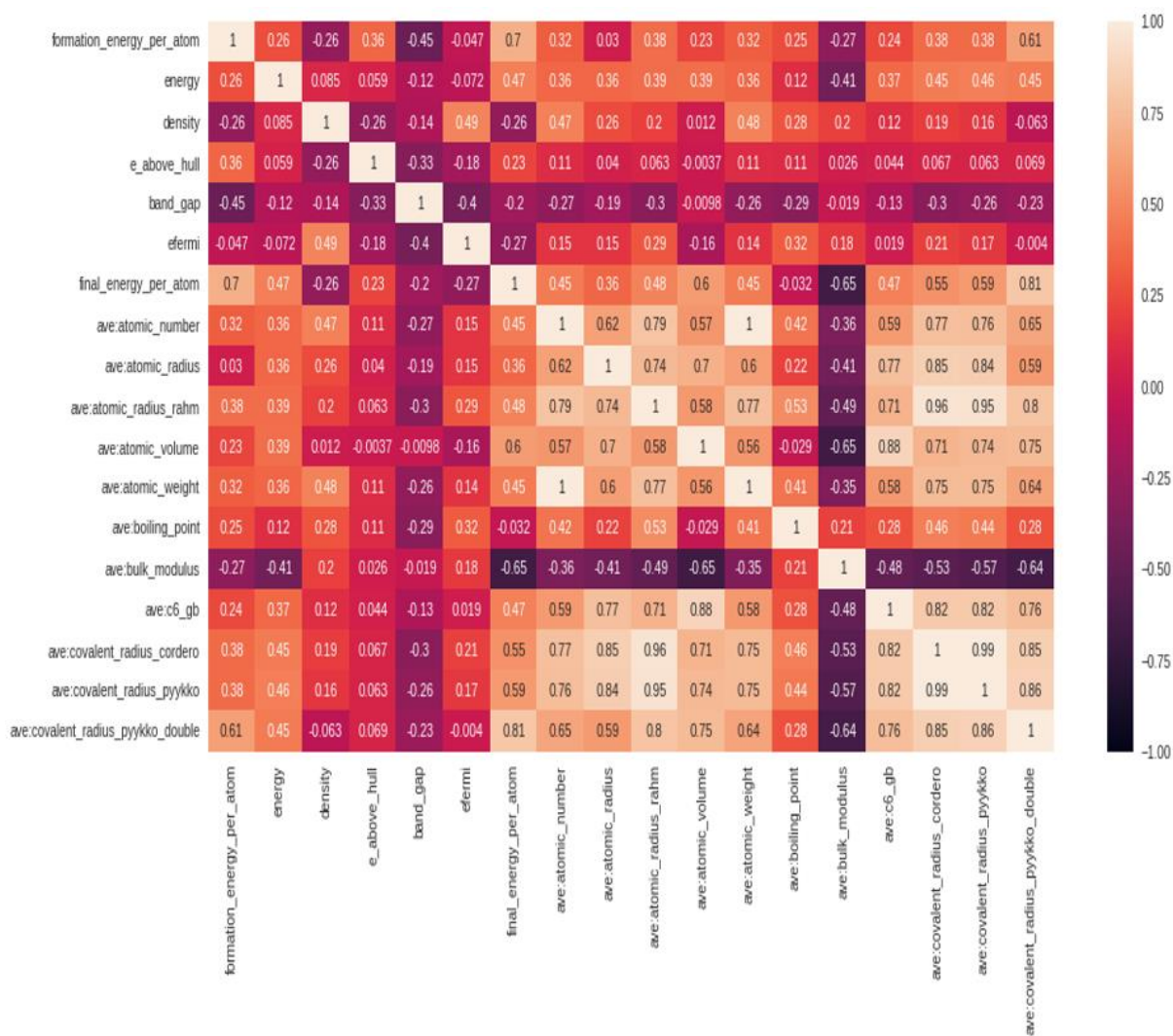


Figure 4.1: Correlation heatmap for the critical feature vectors selected by the Bayesian ridge model.

4.1.2 Energy above Hull

As discussed in chapter 2, feature importance refers to the technique of calculating scores for all input features for a given model, the scores simply represent the "importance" of the input features. The higher the score, the greater the impact of that specific feature on the model. The process of selection allows for easy interpretation of the features. In tree-based machine learning algorithms such as random forest, extra trees regressor and boosting algorithms, the feature importance attribute provides a value between 0 and 100 to represent how useful each feature is in predicting a target property. Thus, it enables us to determine what features contributed to model accuracy and what features are not that important. Using this information,

the model can be tested as to whether it is working as expected and discard those features that do not add value. 18 features are considered, and the features are ranked from the least to more important in calculating the target properties (bottom-up). In this study light gradient boosting machine and extra trees regressor models are used to determine important features required to predict the properties.

In order to engineer the features to accurately predict the energy above hull, feature importance plot was considered. The feature importance plot was utilized for the Fermi energy, energy above hull, band gap and density. Figure 4.2 illustrates the key feature vectors selected by the light gradient boosting machine for predicting the energy above hull. Average Mendeleev number (ave:mendeleev_number) and average atomic weight (ave:atomic_weight) are the top important features in calculating the energy above hull. The least important features are average Herfindal-Hirschman index (HHI) reserves values (ave:hhi_r), sum covalent radius (sum:covalent_radius_cordero), average atom volume in inorganic crystal structure database (ave:icsd_volume), and average density functional theory energy per atom (ave:gs_energy).

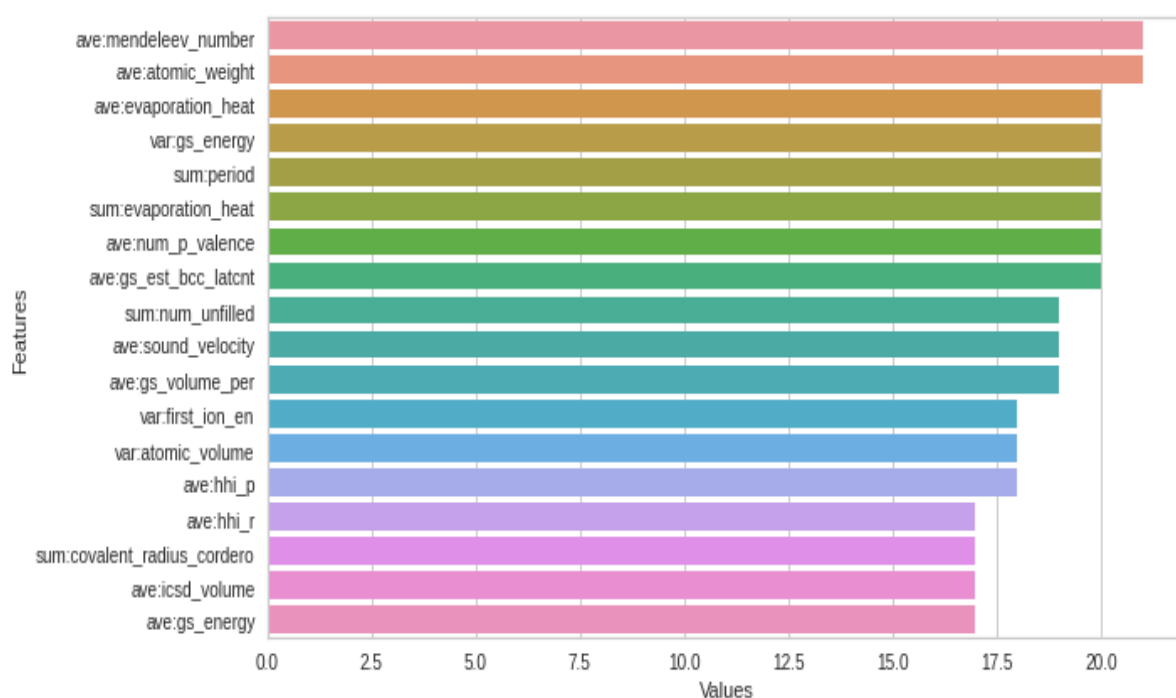


Figure 4.2: Important features selected by the light gradient boosting machine for energy above hull.

Figure 4.3 depicts the key descriptors selected by the extra trees regressor for predicting the energy above hull. Variance atom volume in inorganic crystal structure

database (var:icsd_volume) and average density functional theory energy per atom (ave:gs_energy) are top key features amongst others. Minimum density (min:density) and average covalent radius (ave:covalent_radius_pyykko_triple) are the least important features. The feature vectors selected by the light gradient boosting machine and extra trees regressor predicts the different features vectors, with only two similar features varying in their values. The worst two features selected by LGBM is seen as the top two important features by the ETR. For LGBM model feature values are 20.5 out of 100 whereas, for ETR is 0.00200 out of 100, hence the LGBM was selected in predicting the energy above hull, since the LGBM model features are highly useful compared to the ETR feature vectors.

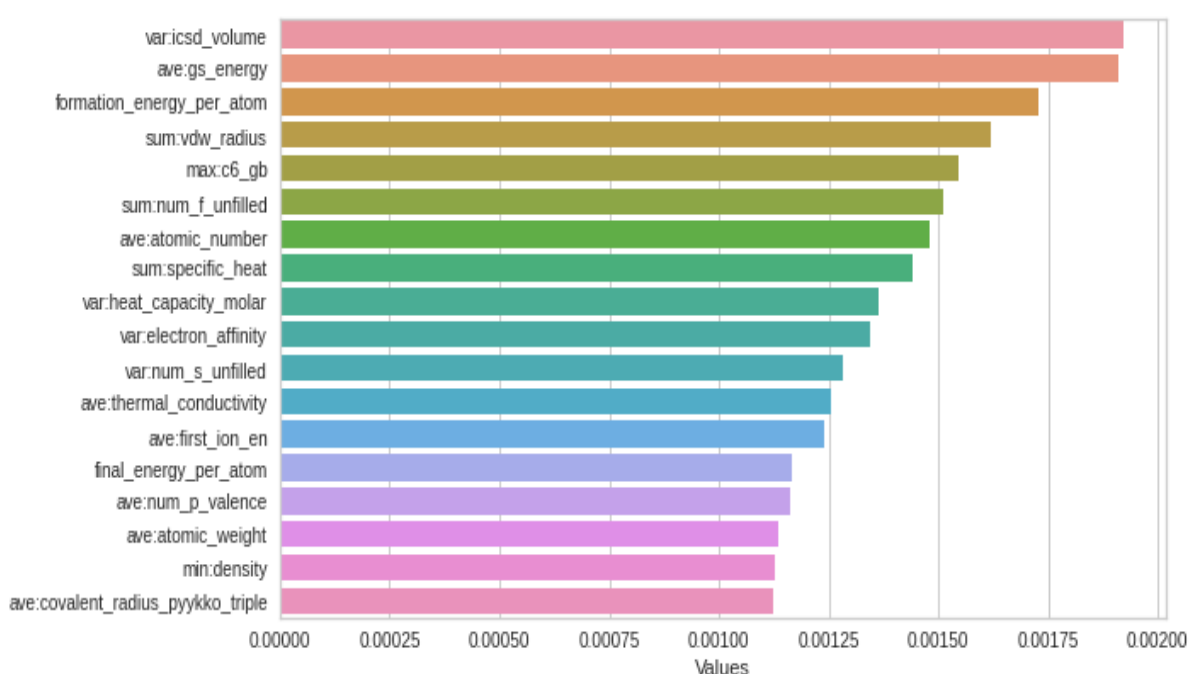


Figure 4.3: Optimal features selected by the extra trees regressor for energy above hull.

4.1.3 Fermi Energy

Shown in Figure 4.4 are the key feature vectors selected by the light gradient boosting machine for predicting the Fermi energy. Average estimated face centered cubic lattice parameter based on density functional theory volume (ave:gs_est_fcc_latcnt), average Ghosh's scale of electronegativity (ave:en_ghosh) and average density (ave:density) are the top three important features. Variance estimated body centred

cubic lattice parameter based on the density functional theory volume (var_gs_est_bcc_latcnt) is the least important feature. The key feature vector shows the contribution of body centred cubic and face centred cubic, and the results implies that most of the materials used are cubic in nature.

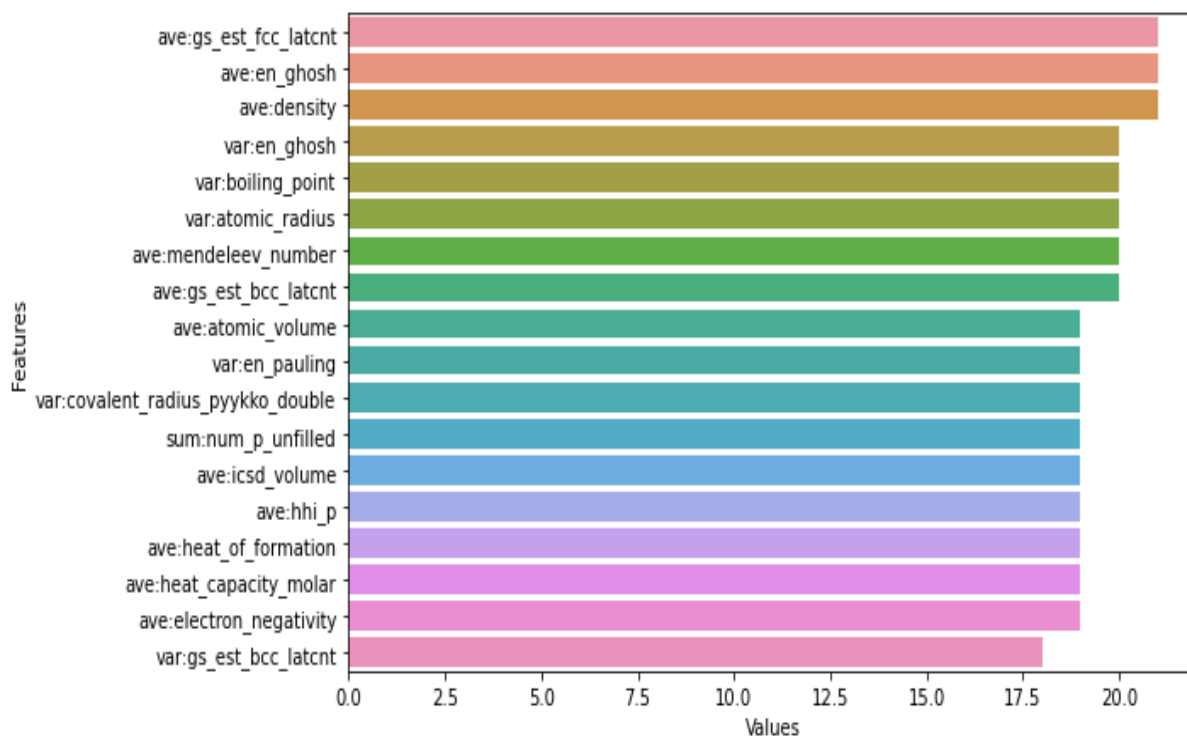


Figure 4.4: Important features for Fermi energy selected by light gradient boosting machine model.

The key features selected by the extra trees regressor for predicting the Fermi energy are depicted in Figure 4.5. The maximum lattice constant (max:lattice_constant), sum period (sum:period) and average atomic volume (ave:atomic_volume) are the most important features. Both the light gradient boosting machine and extra trees regressor predicted similar most feature vectors, namely, average Ghosh scale electronegativity (ave:en_ghosh), average density (ave:density), average atomic volume (ave:atomic_volume), variance covalent radius (var:covalent_radius_pyykko_double), and average heat capacity molar. It is worth noting that the feature vectors selected by the models vary by their importance, those that perform better at LGBM perform worst for ETR.

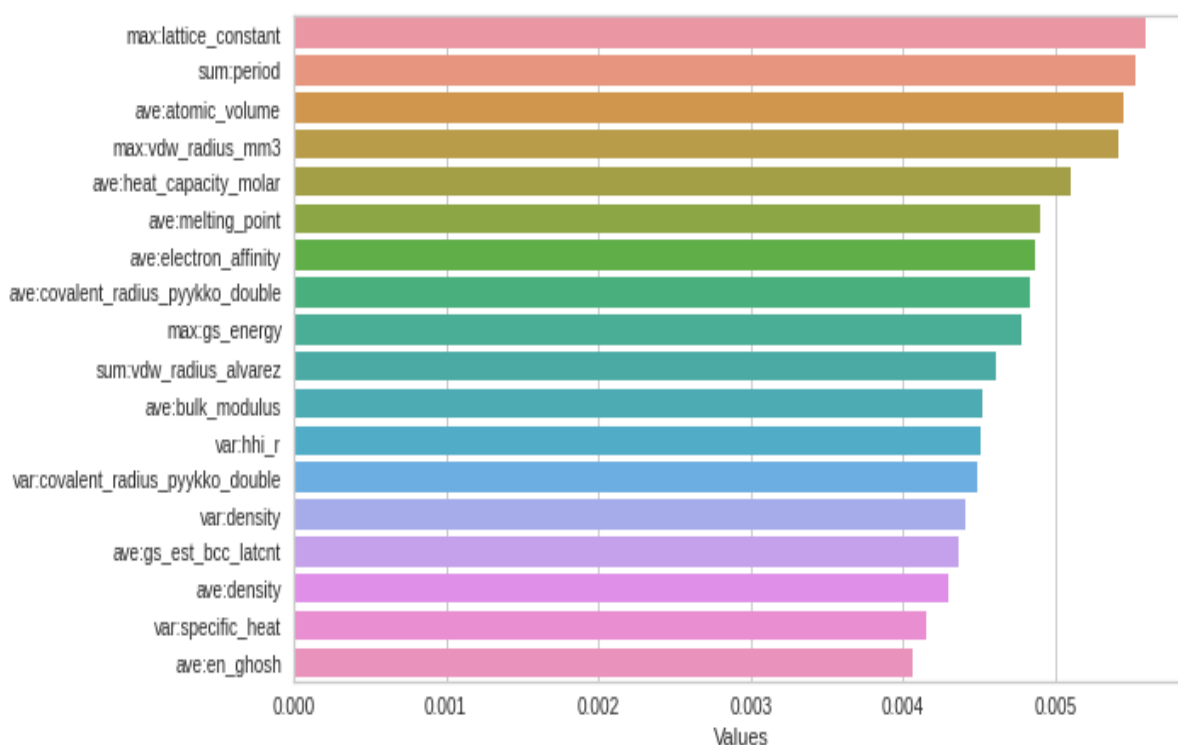


Figure 4.5: Optimal features selected by the extra trees regressor for Fermi energy.

4.1.4 Density

Figure 4.6 illustrates the key features selected by the extra tree regressor for density prediction, and it depicts that maximum mass specific heat capacity (max:heat_capacity_mass) is the most important feature for calculating the density of sodium-ion battery materials. The second important feature is variance density functional theory energy per atom energy (var:gs_energy), followed by maximum dipole polarizability (max:dipole_polarizability). Average electron negativity (ave:electron_negativity) and variance bulk modulus (var:bulk_modulus) are the least important features amongst the features selected.

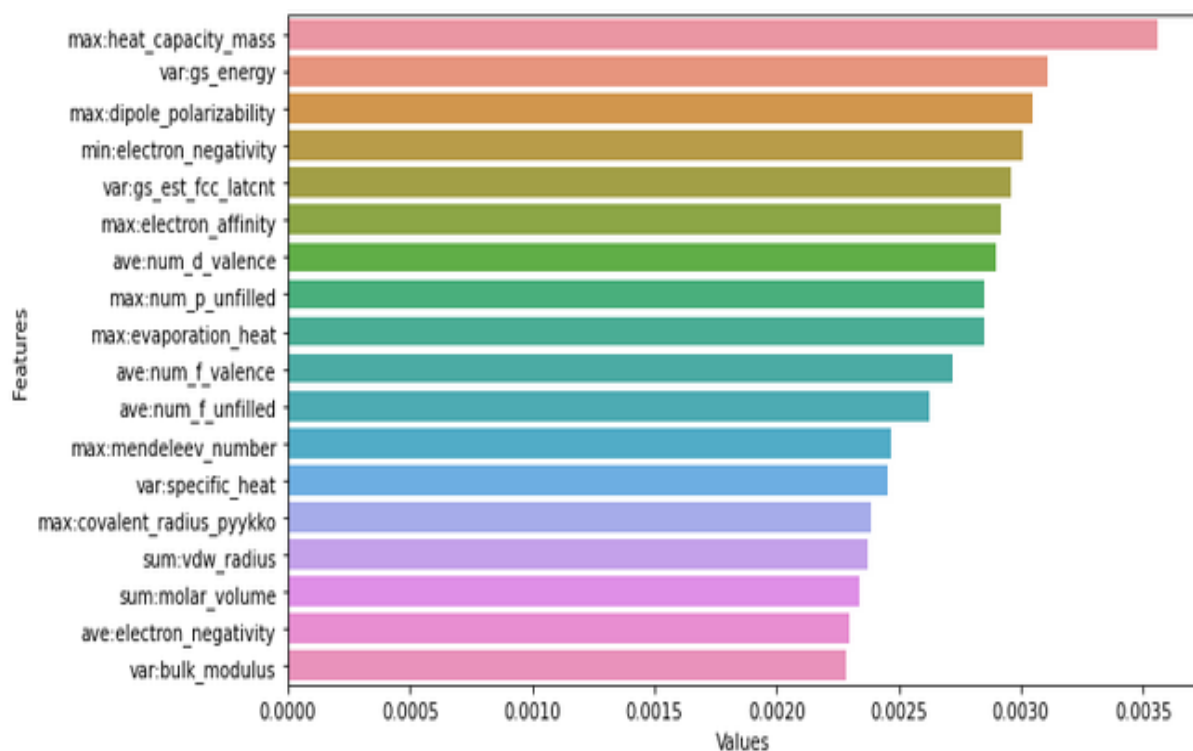


Figure 4.6: Important feature vectors selected by the extra tree regressor for density.

Figure 4.7 depicts the key feature variables selected by the light gradient boosting machine for predicting the density. Average density (ave:density), maximum Van der Waals radius (max:vdw_radius_mm3), average lattice constant (ave:lattice_constant) and average electronegativity Pauling (ave:en_pauling) feature vectors plays a critical role in predicting the density by the LGBM model. The models have a common feature vector, which is sum van der Waals radius (sum:vdw_radius), suggesting its importance in predicting the density of a material using ML. The common feature vector was the most important under the light gradient boosting machine, whereas of least importance under the extra trees regressor. Afzal et al. [111], predicted organic molecules packing density using Deep Neural Networks (DNNs). Approximately 197 descriptors were used, of which constitutional indices and functional group counts were the most critical. The descriptors were reasonably accurate [111].

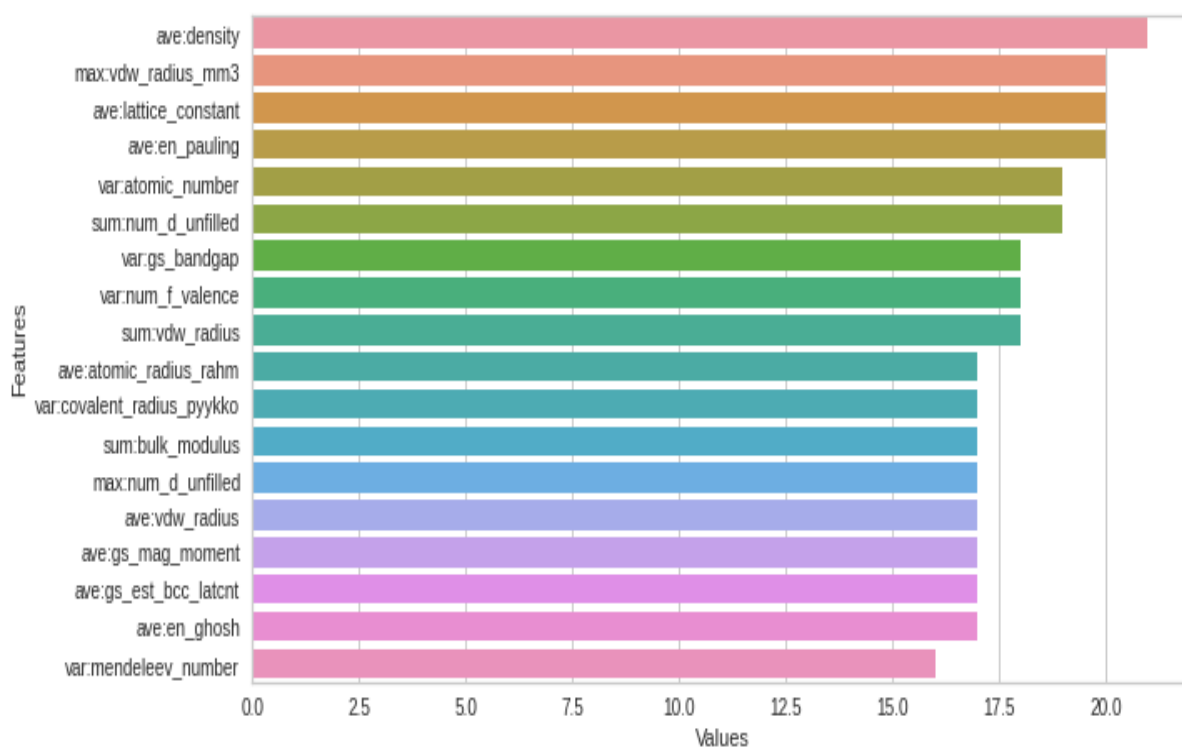


Figure 4.7: Optimal features selected by the light gradient boosting machine for density.

4.1.5 Band Gap

Figure 4.8 illustrates the key features selected by the light gradient boosting machine for predicting band gap. The features such as sum valence electron in d shell (sum:num_d_valence), average van der Waals radius (ave:vdw_radius) and average Ghosh's scale of electronegativity (ave:en_ghosh) are the top three features attributing the most to the predictive power of LGBM model for calculating the band gap. The variance first ionisation energy (var:first_ion_en) was found to be the least important feature.

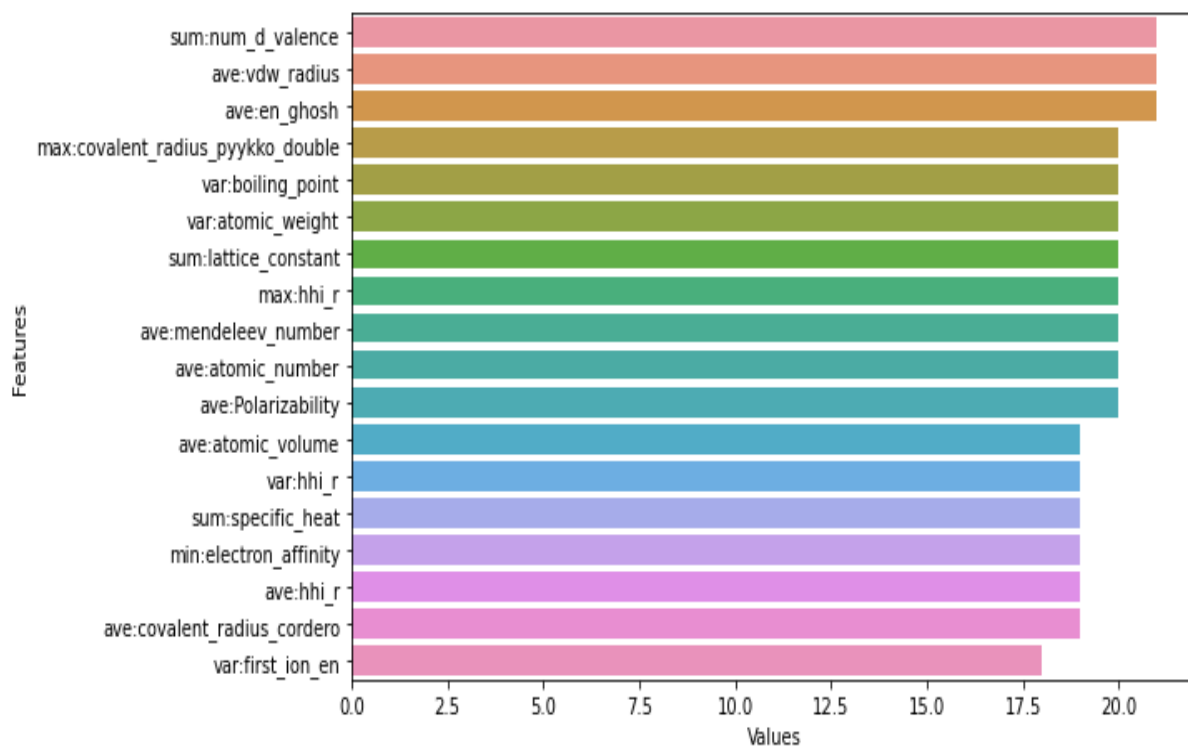


Figure 4.8: Important features selected by the light gradient boosting machine for band gap.

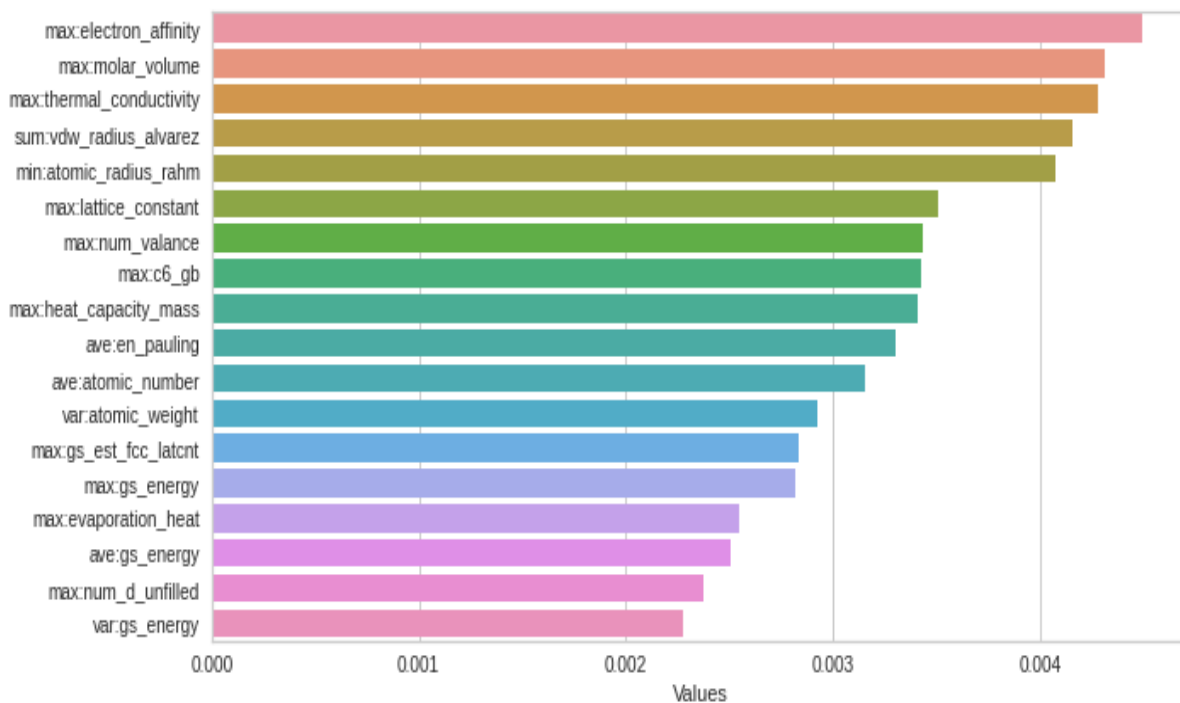


Figure 4.9: Optimal features selected by the extra trees regressor for predicting the band gap.

Figure 4.9 depicts the key features selected by the extra trees regressor for band gap. The important features in predicting the band gap are maximum electron negativity (max:electron_affinity), maximum molar volume (max:molar_volume) and maximum thermal conductivity (max:thermal_conductivity). The LGBM and ETR models have a common feature vector, which is average atomic volume (ave:atomic_volume) however, the other important features are different. The features selected by the LGBM model have high predictive capability as compared to ETR features, this is justified by the score of the values on the feature importance plot.

4.2 Model Selection

Model selection is the task of selecting statistical model from a candidate models, given the data to be analysed. Model selection is used to determine which model is most suitable for the data that has been collected by comparing their relative advantages. The best model is selected based on its capability to predict the target property, in this case the target properties included the formation energy, final energy, Fermi energy, energy above hull, band gap, and density. The following models were evaluated: Bayesian ridge (BR), extra trees regressor (ETR), light gradient boosting machine (LGBM), orthogonal matching pursuit (OMP), random forest regressor (RFR) and gradient boosting regressor (GBR).

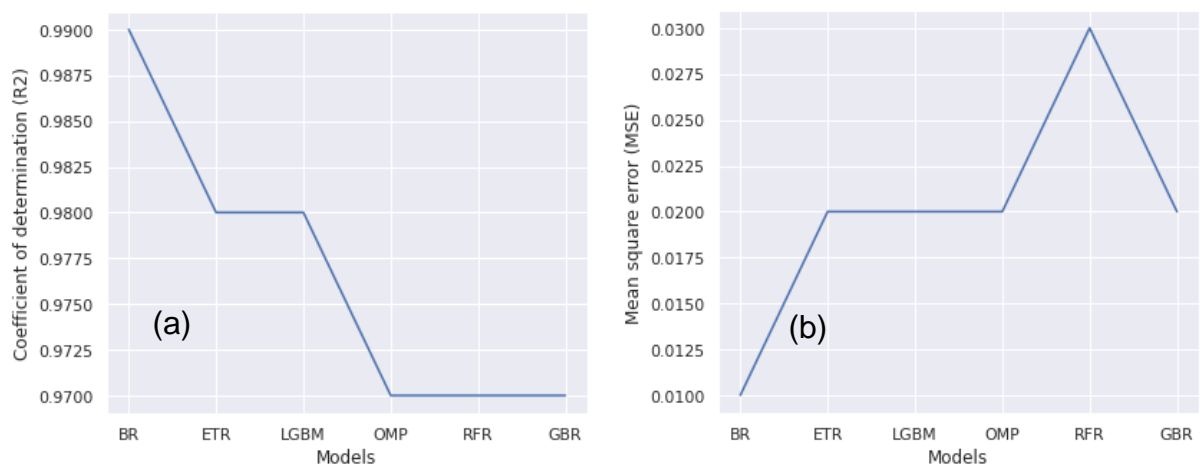


Figure 4.10: Measures of predicted formation energy (a) coefficient of determination and (b) mean square error as determined by various models.

Figure 4.10 depicts the measures of the predicted coefficient of determination (a) and mean square error (b) for formation energy as determined by various models. The models in category from poor to best performing are the gradient boosting regressor, random forest regressor and orthogonal matching pursuit with the same regression score of 0.97 and MSE of 0.02, 0.03 and 0.02 eV, respectively. Both light gradient boosting machine and extra trees regressor have a regression score of 0.98 and mean square error of 0.02 eV. The Bayesian ridge with regression score of 0.99 and MSE of 0.01 eV, predicts the formation energy with highest accuracy of 0.99 and a mean square error closest to zero. Bayesian ridge is easy, fast to implement and can handle large dataset in comparison to other regression models, it is also not sensitive to unrelated features hence is more accurate than the other selected models. Mphaka [110] used the catboost model to predict the formation energy of lithium-ion battery materials, the model was reported to perform well, however it was not included in this study. The catboost achieved a regression score of 0.95 and mean square error of 0.06 eV, which is good but not better than the Bayesian ridge used in this study.

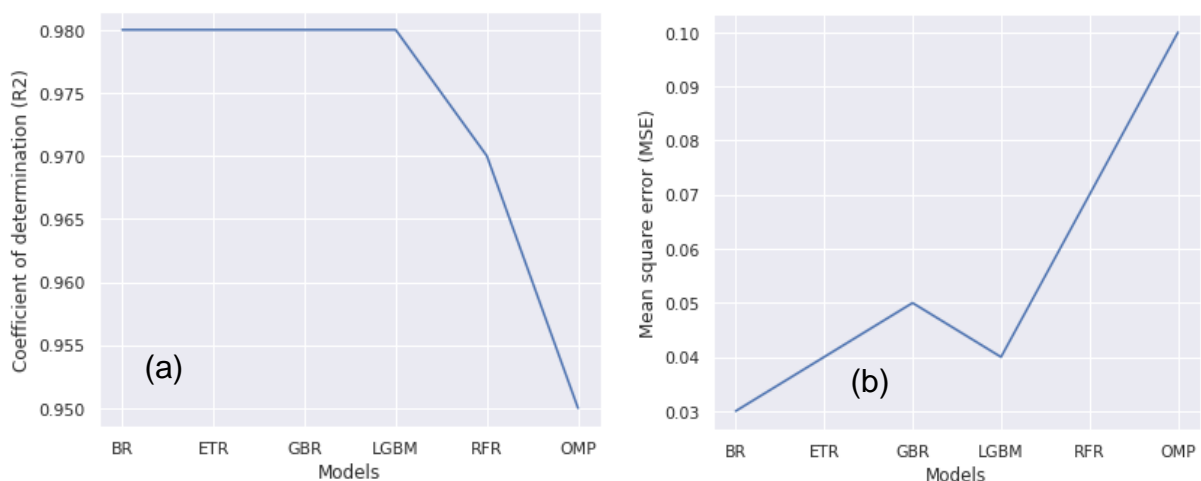


Figure 4.11: Measures of predicted final energy (a) coefficient of determination and (b) mean square error as determined by various models.

Figure 4.11 depicts the measures of predicted coefficient of determination (a) and mean square error (b) for final energy. According to their performance, the following models are categorized from poor to best: the orthogonal matching pursuit with regression score of 0.95 and MSE of 0.10 eV was the poor performing model; followed by random forest regressor with regression score of 0.97 and MSE of 0.07 eV. Light gradient boosting machine, gradient boosting regressor, extra trees regressor, and

Bayesian ridge, have the same regression score of 0.98, and MSE of 0.04, 0.05, 0.04 and 0.03 eV, respectively. The Bayesian ridge model may be regarded as the best in predicting the final energy, since it has the least MSE compared to the other three models with the same regression score. Bayesian ridge is straight forward to fit complex datasets, and its results are highly interpretable and easy to understand.

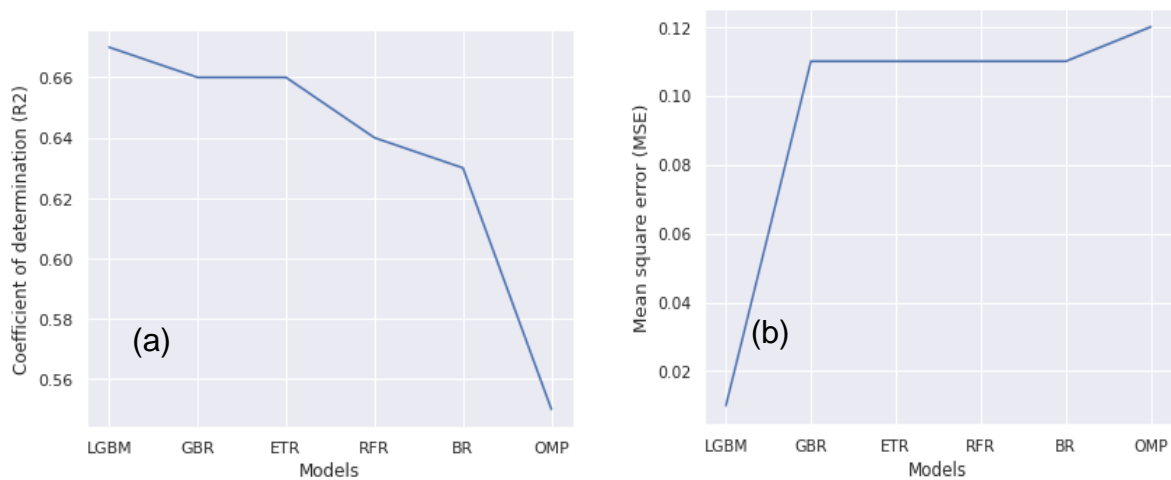


Figure 4.12: Measures of predicted energy above hull (a) coefficient of determination and (b) mean square error as determined by various models.

Figure 4.12 shows the measures of the predicted coefficient of determination (a) and mean square error (b) for the energy above hull. Again, orthogonal matching pursuit is the worst performing model with regression score of 0.55 and MSE of 0.12 eV, followed by the Bayesian ridge with regression score of 0.63 and MSE of 0.11 eV and random forest regressor with regression score of 0.64 and MSE of 0.11 eV. The three aforementioned models performed poorly as compared to the extra trees regressor with regression score of 0.66 and MSE of 0.11 eV, gradient boosting regressor with regression score of 0.66 and MSE of 0.11 eV and light gradient boosting machine with regression score of 0.67 and MSE of 0.01 eV. For LGBM model it is possible to reduce memory usage and increase efficiency, it is capable of capturing complex patterns in the data, this substantiate how best this model is in predicting energy above hull. Overall, all the models did not predict the energy above hull well.

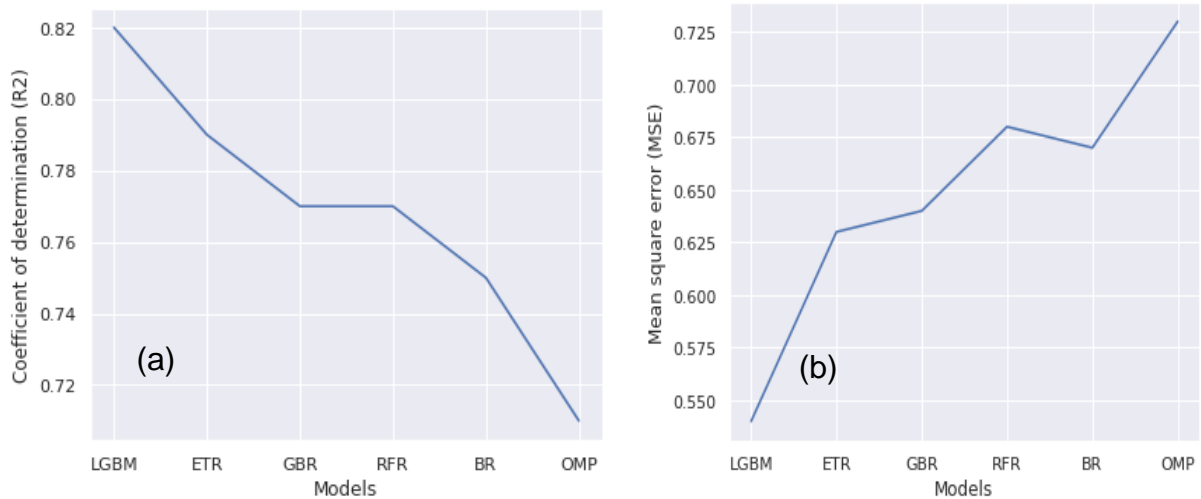


Figure 4.13: Measures of predicted Fermi energy (a) coefficient of determination and (b) mean square error as determined by various models.

Figure 4.13 depicts the measures of the predicted coefficient of determination (a) and mean square error (b) for Fermi energy. The orthogonal matching pursuit with regression score of 0.71 and MSE of 0.73 eV, followed by Bayesian ridge with regression score of 0.75 and MSE of 0.67 eV and random forest regressor with regression score of 0.77 and MSE of 0.68 eV. The three models mentioned above were slightly outperformed by the gradient boosting regressor with regression score of 0.77 and MSE of 0.64 eV, extra trees regressor with regression score of 0.79 and MSE of 0.63 eV, and light gradient boosting machine with regression score of 0.82 and MSE of 0.54 eV. The MSE are far from zero, suggesting that, overall the Fermi energy is not well predicted compared to the other properties discussed in the previous subsections. Despite that all the models did not perform well, LGBM algorithm was selected due to its exceptional results in a multitude of machine learning endeavours. It wins in terms of performance and speed compared to the tested models.

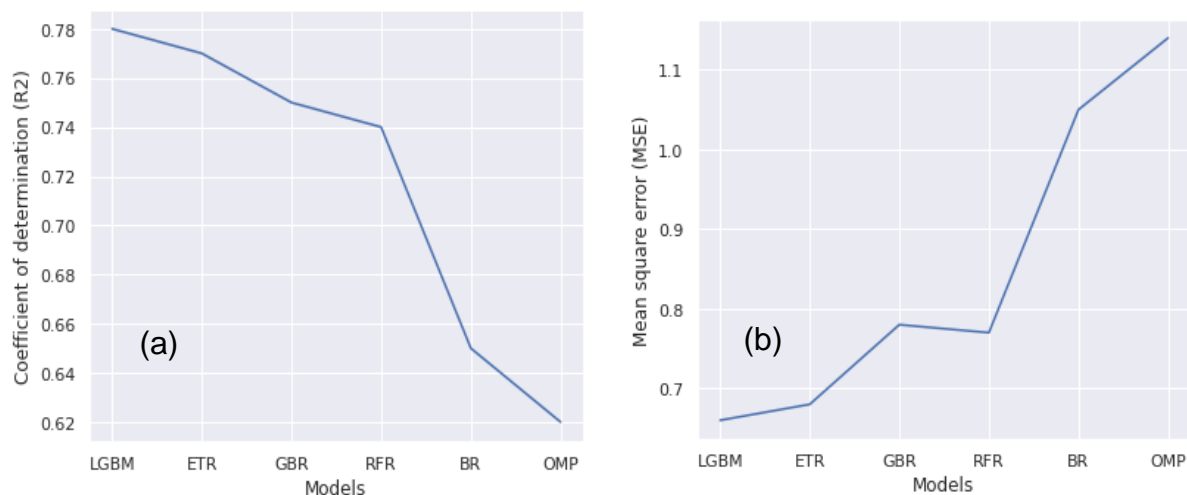


Figure 4.14: Measures of predicted band gap (a) coefficient of determination and (b) mean square error as determined by various models.

The measures of the predicted coefficient of determination (a) and mean square error (b) for band gap are shown in Figure 4.14. The orthogonal matching pursuit with regression score of 0.62 and MSE of 1.14 eV and Bayesian ridge with regression score of 0.65 and MSE of 1.05 eV are the poor performing models. The mean square error of the Bayesian ridge model is very high suggesting worst performance. Random forest regressor with regression score of 0.74 and MSE of 0.76 eV, gradient boosting regressor with regression score of 0.75 and MSE of 0.77 eV, extra trees regressor with regression score of 0.77 and MSE of 0.68 eV and light gradient boosting machine with regression score of 0.78 and MSE of 0.66 eV. The extra trees regressor was considered the best model since it had a regression score closest to 1 and mean square error close to 0 as compared to other evaluated models. LGBM is capable of capturing complex patterns in the data and increase efficiency hence it performed better than other selected models. Li et al. [112], reported gradient boosting regressor (GBR) model to be performing well in predicting the band gap of perovskites, achieving a regression score of 0.87 and MSE of 0.21 eV. However, in this study GBR was outperformed by light gradient boosting machine and ETR. These differences are attributed by factors such as data used, model parameters, types of materials, etc.

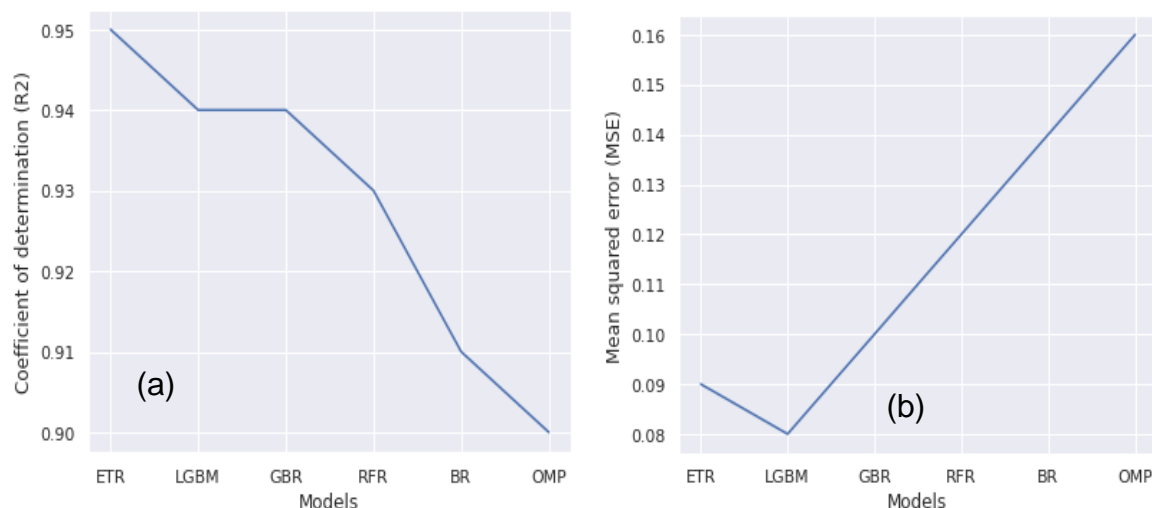


Figure 4.15: Measures of predicted density (a) coefficient of determination and (b) mean square error as determined by various models.

Figure 4.15 illustrates the measures of the predicted coefficient of determination (a) and mean square error (b) for density. Below are the models ranked from poor to best performance: orthogonal matching pursuit with regression score of 0.90 and MSE of 0.16 eV, Bayesian ridge with regression score of 0.91 and MSE of 0.14 eV, random forest regressor with regression score of 0.93 and MSE of 0.12 eV. Both gradient boosting regressor and light gradient boosting machine have a regression score of 0.94, MSE of 0.10 eV and 0.08 eV, respectively. The extra trees regressor model predicted the density with unprecedented high regression score of 0.95 and low MSE of 0.09 eV. Unlike in other predicted properties discussed in the previous sections, all the models achieved a performance score of above 0.9. Afzal et al. [111], used DNN methods to predict organic molecular density and achieved a regression score of 0.98 and mean absolute error of 10.8 kg/m³. The DNN model predicted the density with high accuracy [111].

4.3 Model Tuning/Hyperparameter Optimization

As discussed in chapter 3, model tuning is the process of finding the optimal values of hyperparameters to maximize model performance. Choosing the best regularization is critical, as small regularizations lead to complex models, while large regularizations are not effective and make the model less useful. In this case, grid search technique is used to tune the models. In the next three subsections, we discuss the parameters tuned for various algorithms in order to improve their performance.

4.3.1 Bayesian Ridge

In this model, only iterations are considered and the hyperparameters are listed in Table 4.1.

Table 4.1: Tuned Bayesian ridge model parameters from training set for formation energy and final energy.

Target Property (eV)	Bayesian Ridge Hyperparameter	R ²
Formation energy		
Without CV	300	0.98
With CV	50	0.99
Final energy		
Without CV	300	0.98
With CV	50	0.98

The hyperparameters without cross validation (CV) with iterations of 300, resulted with model regression score of 0.98 for both energies. According to this matrix evaluation methods model Bayesian ridge seemed to be the best model with high capability for the prediction of both formation energy and final energy. The iterations were changed to 50, under 5-fold cross validation and regression scores of 0.99 and 0.98 are obtained for formation energy and final energy, respectively as shown in table 4.1. From the Scikit learn the maximum number of iterations for a Bayesian ridge model should be greater or equals to 1. By default, Bayesian ridge also have alpha 1, alpha 2, lambda 1 and 2 which equals to 1×10^{-6} hyperparameters. After tuning the model by increasing the iterations, the regression score reduced to 0.98, implying that the model does not improve as it has a regression score of 0.98 for formation energy. Tuning the iteration for final energy gave the same regression score of 0.98.

The algorithm handles the process of changing features obtained from elements into a forecast utilizing purely statistical techniques. The expectation is that the key characteristics must correspond to the formation and the final energy.

4.3.2 Light Gradient Boosting Machine

The light gradient boosting machine parameters that are considered for model tuning are:

- i. Maximum depth (max depth) - the maximum depth of a tree
- ii. L2 regularization - the model needs L2 regularization to make predictions. The model's ability to predict is determined by L2 values. These values are estimated from data, or they can be learned from data. In many cases, they are not set manually. They may be stored in learned models.
- iii. Bagging temperature - Bayesian bootstrapping is controlled by this parameter, ranging from zero to infinity.

Table 4.2: Tuned light gradient boosting machine model parameters from training set for Fermi and energy above hull.

Target Property (eV)	Number of trees	Maximum depth	L2 regularization	Bagging Temperature	R ²
Fermi energy					
Without CV	1000	3	1	1	0.80
With CV	350	10	10	20	0.82
Energy above hull					
Without CV	1000	3	1	1	0.58
With CV	350	10	10	20	0.67

For light gradient boosting machine model (table 4.2), hyperparameters without cross validation are number of trees of 1000, maximum depth of 3, L2 regularization of 1, bagging temperature of 1 which resulted in regression score of 0.80, and 0.58 for Fermi energy and energy above hull, respectively. With cross validation, after tuning the model by 5-fold, number of trees are 350, maximum depth of 10, L2 regularization of 10 and bagging temperature of 20 the regression score improved. Regression score of Fermi energy optimized from 0.80 to 0.82 and falls between the accepted score of 0.6 to 1, suggesting that the model does not suffer under or overfitting. The model performance is fairly good. For energy above hull regression score improved from 0.58 to 0.67. The regression score for energy above hull now falls above 0.6, which is within an acceptable score range. This lower regression score of 0.6 may have been caused

by model underfitting on the training dataset. The overall performance for energy above hull is poor.

4.3.3 Extra Trees Regressor

The hyperparameters considered in tuning extra trees include:

- i. Maximum depth - the maximum depth of a tree, the length of the longest path from a root to a leaf to which each tree will be build.
- ii. Maximum leaf node - the maximum node which does not have any child node, it helps in reducing overfitting and reduce model biasness.

For extra tree regressor model (table 4.3), hyperparameters without cross validation are maximum depth of 4, maximum leaf node of 1 which resulted in model regression score of 0.94 and 0.76 for density and band gap, respectively. With cross validation the maximum depth and maximum leaf node changed to 10 and 13, respectively. This improved the regression score by 1 percent and regression score optimized from 0.94 to 0.95 for density. The regression score for band gap improved from 0.76 to 0.78.

Table 4.3: Tuned extra trees regressor model parameters from training set for density and band gap.

Target property	Maximum depth	Maximum leaf node	R ²
Density (g/cm³)			
Without CV	4	1	0.94
With CV	10	13	0.95
Band gap (eV)			
Without CV	4	1	0.76
With CV	10	13	0.78

4.4 Model Performance

Model performance is an assessment of the model's ability to perform a task accurately. It is measured based on the comparison of the models' predictions with the (known) values of the dependent variable in a dataset. The performance plays a dominant role in the predictive modelling technique since it determines whether a model is performing efficiently or not. The DFT calculated properties from the Materials Project Database are compared with the corresponding ML predicted values for model

performance, using parity plots. For the best advantage most points should pass through the diagonal regressor line.

Figure 4.16 shows a graphical representation of model performance of the training set (left) and the testing set (right), containing data points reflecting the predicted formation energy in (eV) as a function of DFT calculated formation energy in (eV). Bayesian ridge regression with elemental descriptors predicted formation energy scores of 0.99, 0.98 and 0.01 eV, 0.03 eV for coefficient of determination R^2 and MSE for the training and the testing sets, respectively. The results indicated an excellent agreement between the calculated DFT and the predicted formation for the train and the test sets. This further demonstrate the robustness nature of our machine-learning model.

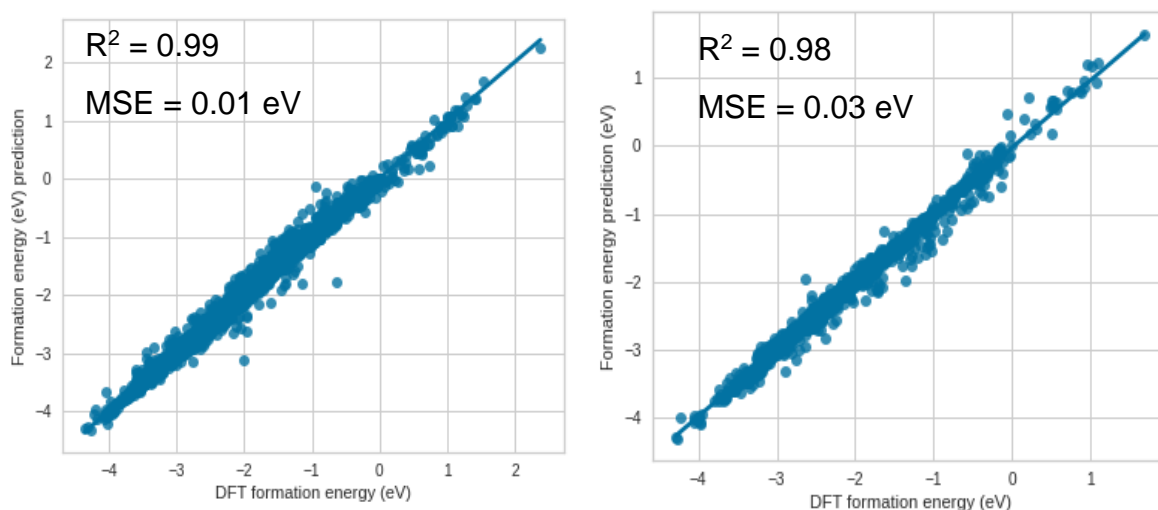


Figure 4.16: Parity plot of Bayesian ridge model predicted formation energy versus DFT formation energy showing model performance for training set (left) and test set (right).

Figure 4.17 shows the performance of the model on the training set (left) and the test set (right), with data points representing predicted final energies in (eV) versus DFT calculated final energies in (eV). Bayesian ridge regression was found to be the best performing model, predicting the final energy with scores of 0.98, 0.97 and 0.03 eV, 0.04 eV for coefficient of determination and MSE for the training and the testing sets, respectively. The regression score and the mean-square error of the calculated DFT and predicted final energy of the train and the test sets showed a good consistency. The results showed that the calculated final energies of SIB dataset from the MP

database can be used as a test set for further justification on the selection of ML training using an experimental sample set.

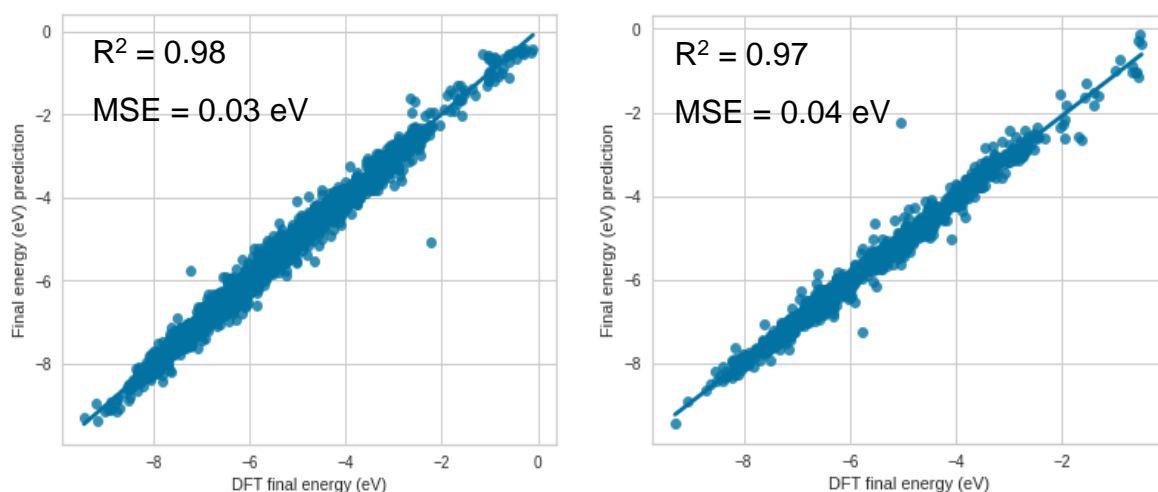


Figure 4.17: The parity plot compares the predicted final energy with DFT final energy for the training set (left) and test set (right) model performance by the Bayesian ridge.

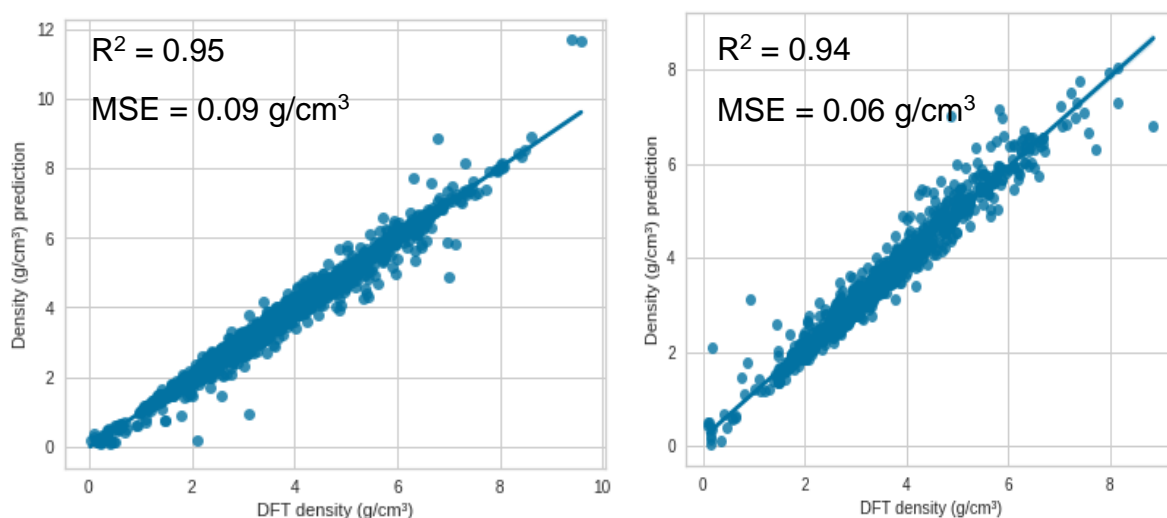


Figure 4.18: Parity plot showing the performance of the extra trees regressor model predicted density versus the DFT density in the training set (left) and test set (right).

A graphical representation of model performance is presented in training (left) and test (right) sets in figure 4.18, with data points that show predicted density in (g/cm^3) versus DFT calculated density in (g/cm^3). ETR regressor model with elemental descriptors, predicted the density with scores of 0.95, 0.94 and 0.09 g/cm^3 , 0.06 g/cm^3 for coefficient of determination R^2 and MSE for the training and testing sets, respectively.

In comparison to the test set, the training set contains data points that are very close to the regression line, and are also less dispersed. In addition, the regression scores for both the training and the testing sets are close to 1, confirming that the extra trees regressor model is the best effective model for predicting density.

The parity plot in Figure 4.19 represents the predicted Fermi energy in (eV) as a function of the DFT calculated Fermi energy, also in (eV). LGBM predicted the Fermi energy with coefficient of determination R^2 of 0.82 and MSE of 0.57 eV and R^2 of 0.80 and MSE of 0.57 eV for the training and the testing sets, respectively. The parity plot in Figure 4.19 showed a satisfactory correlation between the calculated DFT and predicted Fermi energy, and this is justified by the closeness of the data points to the diagonal line, suggesting that the model accurately predicted the outcome of the DFT calculations with a minimal error.

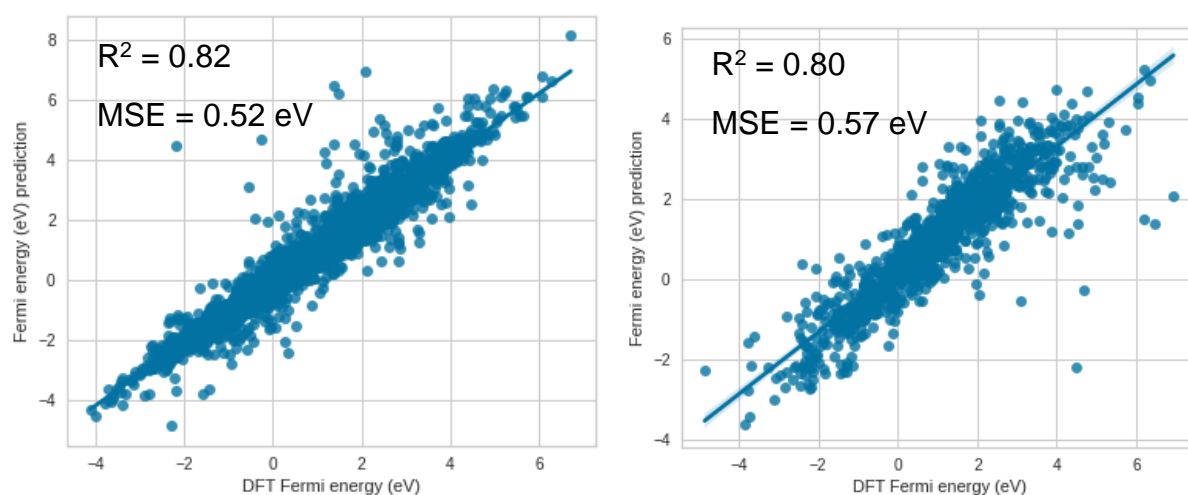


Figure 4.19: The parity plot of LGBM predicted Fermi energy versus DFT Fermi energy model performance for training set (left) and test set (right).

Figure 4.20 depicts parity plots comparing band gap values computed using DFT against predictions made using light gradient boosting machine model, trained using compositional feature vectors. LGBM predicted band gap with scores of 0.78, 0.69 and 0.66 eV, 0.76 eV for coefficient of determination R^2 and MSE for the training and the testing sets, respectively. The training set data points are very close to the regression line, and are also less dispersed, while some of the points for the testing set are scattered outside the regression line. We also observed that there are many outliers points for the testing set as compared to the training set, which explains why

coefficient of determination is smaller for the testing as compared to the one for the training set. However, the regression score is also closer to 1, confirming that LGBM is the best model for band gap prediction. Despite the large MSE, the linear relationship between the predicted and DFT calculated band gap is still preserved by our ML models. However, the poor performance of the model for this test set is reflected in the relatively small R^2 value as well as the ‘best fit line’.

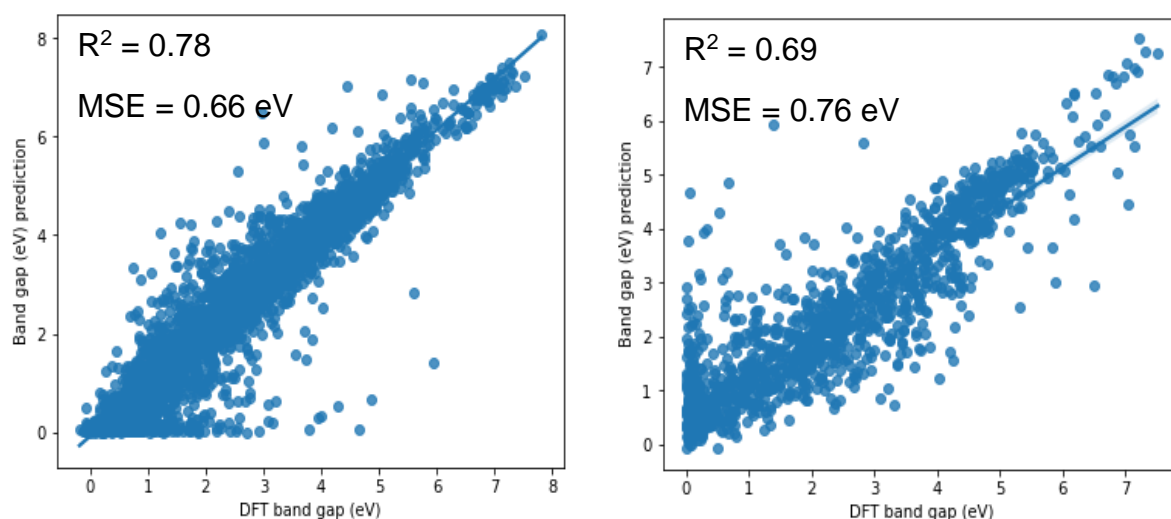


Figure 4.20: The parity plot of LGBM predicted band gap versus DFT band gap model performance for training set (left) and test set (right).

LGBM model performance in the train (left) and the test (right) sets is shown in Figure 4.21, which depicts DFT calculated energy above hull and the predicted energy above hull in (eV). Based on elements descriptors, light gradient boosting machine predicted energy above hull with scores of 0.67, 0.58 and 0.01 eV, 0.05 eV for coefficient of determination and MSE for the training and testing sets, respectively. LGBM model predicted the regression score 0.67, which is considered to be low according to the regression accuracy measures. This may have been caused by the model’s failure to select optimal features for this target property. We also observed that there are many outliers for the testing set as compared to the training set, meaning the coefficient of determination is smaller compared to the one for training set, hence the score of 0.58 was obtained. This regression score is too far from 1, implying that light gradient boosting machine is not the best model for energy above hull prediction. The findings correlated with those reported in model selection section. The model was trained under several different conditions, however the accuracy of the model did not improve.

The few data points obtained could be contributing factor to the poor model performance.

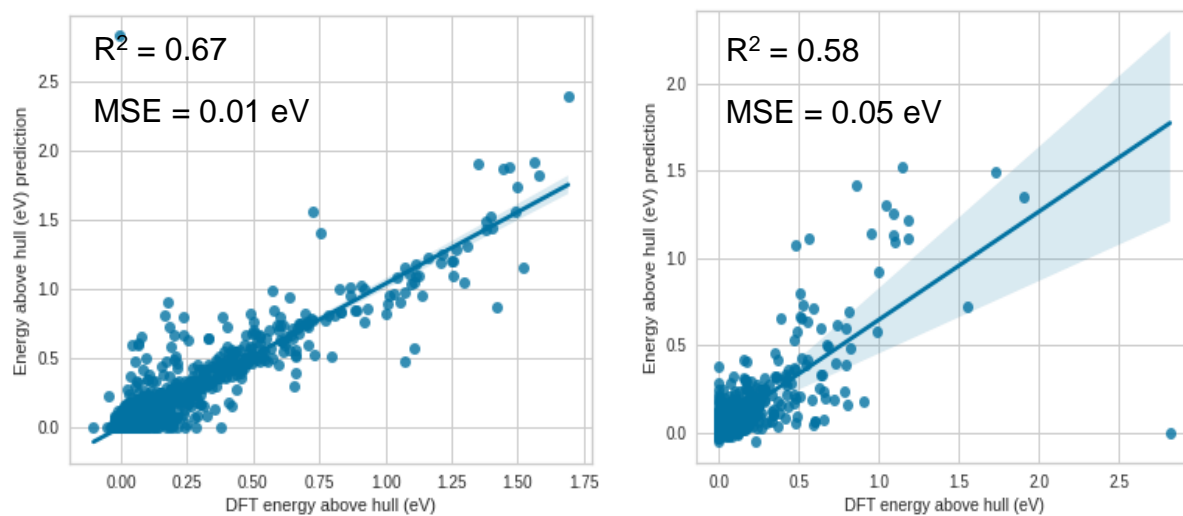


Figure 4.21: The parity plot of LGBM predicted energy above hull versus DFT energy above hull model performance for training set (left) and test set (right).

CHAPTER 5

CONCLUSION

Machine learning models were successfully developed to predict properties of sodium-ion battery materials, namely, formation energy, final energy, Fermi energy, energy above hull, density and band gap. Firstly, feature engineering process was carried out to determine the most important features for different properties, as there is no descriptive information available from the literature. The importance of feature vectors derived from the properties of the materials' chemical compounds and the fundamental properties of their constituents' elements was evaluated for all the aforementioned properties of interest. The average covalent radius and average single bond covalent radius were found to be the most important descriptors for predicting the formation and final energy of the SIB materials as per Bayesian ridge model. The average estimated FCC lattice parameter based on the DFT volume, average Ghosh's scale of electronegativity and average density were the important features for predicting the Fermi energy of SIB materials as per light gradient boosting machine algorithm. The most targeted variables according to light gradient boosting machine resulting in accuracy and performance for predicting the energy above hull were found to be the average atomic number in Mendeleev's periodic table and atomic weight. The maximum mass specific heat capacity and variance of DFT energy per atom descriptors are the most important features in predicting the density of the material. The average Van der Waals radius, valence electron in d shell and average electronegativity were the most important features for predicting the energy band gap using LGBM. Essentially, we established important features for the different properties associated with SIB materials.

Secondly, after the feature engineering process was completed, different ML models were evaluated, and the best model was selected based on its accuracy in predicting the afore-mentioned properties. Amongst various algorithms that were evaluated, the Bayesian ridge model was found to be the best model in predicting the formation energy with an accuracy of 0.99 and 0.01 eV coefficient of determination and mean square error, respectively, as well as predicting the final energy with 0.98 and 0.03 eV accuracy for the coefficient of determination and mean square error, respectively. Light

gradient boosting machine model was found to be the best model in predicting the Fermi energy with an accuracy of 0.82 and 0.52 eV coefficient of determination and mean square error, respectively, and energy above hull with 0.67 and 0.01 eV, for the coefficient of determination and mean square error, respectively. Notably, although LGBM was found to be the best model, the MSE for predicting the Fermi energy was relatively higher in terms of accuracy of measure, on the other hand the R^2 for the energy above hull was lower. ETR is found to be the best model in predicting the density with an accuracy of 0.95 and 0.09 g/cm³ for the coefficient of determination and mean square error, respectively. Lastly, ETR predicted energy band gap with accuracy of 0.78 and 0.66 eV, for the coefficient of determination and mean square error, respectively. The LGBM poor performing model for energy above hull and Fermi energy prediction can be attributed to the model's failure to select optimal features for the target properties, and by model under-fitting on the training dataset.

The machine learning models were further validated by comparing the DFT calculated properties with their corresponding predicted machine learning values. The results suggests that the developed ML models can predict formation energy, final energy, Fermi energy, energy above hull, density, and band gap with near-DFT accuracy. Thus, there is a good agreement between the model performance based on the train and the test set. Machine learning models can yield accurate material properties faster, making them useful in materials-properties prediction.

5.1 Recommendations and Future Work

This ground work could be a stepping stone for exploring other descriptors for electrode materials. Future research requires the development of tools that will enable scientists to predict and classify sodium-ion battery materials properties without going through the different databases and all the machine learning steps. In that case, the following are recommended for future works:

- Development of organized sodium-ion battery materials database that can be easily stored and accessed.
- Development of ML workflows, indicating which steps are implemented and how are the implemented.
- Building a tool that can be used to analyze SIB materials.
- Expanding the models to other types of materials and solving the model's poor performance which was attributed to failure to select optimal features for the target properties, and model underfitting on the training data.

REFERENCES

- [1] K. Steven, S. Kauwe, T. Rhone, and T. Sparks, "Data-driven studies of Li-ion battery materials," *Crystals (Basel)*, vol. 9, pp. 1–9, 2019.
- [2] L. Peng, Y. Zhu, D. Chen, R. Ruoff, and G. Yu, "Two-dimensional materials for beyond lithium-ion batteries," *Adv. Energy Mater.*, vol. 6, p. 1600025, 2016.
- [3] S. Kim, D. Seo, X. Ma, G. Ceder, and K. Kang, "Electrode materials for rechargeable sodium-ion batteries: potential alternatives to current lithium-ion batteries," *Adv. Energy Mater.*, vol. 2, pp. 710–721, 2012.
- [4] Z. Xu, X. Liu, Y. Luo, L. Zhou, and J. Kim, "Nanosilicon anodes for high performance rechargeable batteries," *Prog. Mater. Sci.*, vol. 90, pp. 1–44, 2017.
- [5] J. Hwang, S. Myung, and Y. Sun, "Sodium-ion batteries: present and future," *Chem. Soc. Rev.*, vol. 46, pp. 3529–3614, 2017.
- [6] K. Abraham, "How comparable are sodium-ion batteries to lithium-ion counterparts?," *ACS Energy Lett.*, vol. 5, pp. 3544–3547, 2020.
- [7] "<https://depts.washington.edu>."
- [8] "<https://www.hollingsworth-voise.com>."
- [9] X. He, J. Wang, B. Qiu, E. Paillard, C. Ma, and X. Cao, "Durable high-rate capability $\text{Na}_{0.44}\text{MnO}_2$ cathode material for sodium-ion batteries," *Nano Energy*, vol. 27, pp. 602–610, 2016.
- [10] Y. Lu, L. Li, Q. Zhang, Z. Niu, and J. Chen, "Electrolyte and interface engineering for solid-state sodium batteries," *J. (Basel)*, vol. 2, pp. 1747–1770, 2018.
- [11] Y. Sun, L. Zhao, H. Pan, X. Lu, L. Gu, and Y. Hu, "Direct atomic-scale confirmation of three-phase storage mechanism in $\text{Li}_4\text{Ti}_5\text{O}_{12}$ anodes for room-temperature sodium-ion batteries," *Nat. Commun.*, vol. 4, pp. 1–10, 2013.
- [12] Z. Xu, J. Park, G. Yoon, H. Kim, and K. Kang, "Graphitic carbon materials for advanced sodium-ion batteries," *Small Methods*, vol. 3, pp. 1–42, 2019.
- [13] Z. Xu, S. Yao, J. Cui, L. Zhou, and J. Kim, "Atomic scale, amorphous FeO_x /carbon nanofiber anodes for Li-ion and Na-ion batteries," *Energy. Storage Mater.*, vol. 8, pp. 10–19, 2017.
- [14] Z. Hu, Q. Liu, S. Chou, and S. Dou, "Advances and challenges in metal sulfides/selenides for next-generation rechargeable sodium-ion batteries," *Adv. Mater.*, vol. 29, p. 1700606, 2017.
- [15] M. Lao, Y. Zhang, W. Luo, Q. Yan, W. Sun, and S. Dou, "Alloy-based anode materials toward advanced sodium-ion batteries," *Adv. Mater.*, vol. 29, p. 1700622, 2017.

- [16] Y. Lu, Y. Lu, Z. Niu, and J. Chen, "Graphene-based nanomaterials for sodium-ion batteries," *Adv. Energy Mater.*, vol. 8, p. 1702469, 2018.
- [17] G. Hautier, C. Fischer, A. Jain, T. Mueller, and G. Ceder, "Finding nature's missing ternary oxide compounds using machine learning and density functional theory," *Chem. Mater.*, vol. 22, pp. 3762–3767, 2010.
- [18] R. Armiento, B. Kozinsky, G. Hautier, M. Fornari, and G. Ceder, "High-throughput screening of perovskite alloys for piezoelectric performance and thermodynamic stability," *Phys. Rev. B*, vol. 89, p. 134103, 2014.
- [19] "<https://afloplib.org/>."
- [20] A. Jain, S. Ong, G. Hautier, W. Chen, W. Richards, and S. Dacek, "Commentary: the materials project: a materials genome approach to accelerating materials innovation," *APL Mater.*, vol. 1, p. 011002, 2013.
- [21] "<https://nomad-lab.eu/>."
- [22] "<https://oqmd.org/>."
- [23] S. Curtarolo, W. Setyawan, G. Hart, M. Jahnatek, R. Chepulskii, and R. Taylor, "AFLOW: an automatic framework for high-throughput materials discovery," 2013.
- [24] W. Ye, C. Chen, S. Dwaraknath, A. Jain, S. Ong, and K. Persson, "Harnessing the materials project for machine-learning and accelerated discovery," *MRS Bull.*, vol. 43, pp. 664–669, 2018.
- [25] J. Saal, S. Kirklin, M. Aykol, B. Meredig, and C. Wolverton, "Materials design and discovery with high-throughput density functional theory: the open quantum materials database (OQMD)," *JOM*, vol. 65, pp. 1501–1509, 2013.
- [26] C. Oses, C. Toher, and S. Curtarolo, "Data-driven design of inorganic materials with the automatic flow framework for materials discovery," *MRS Bull.*, vol. 43, pp. 670–675, 2018.
- [27] A. Seko, K. Toyoura, S. Muto, T. Mizoguchi, and S. Broderick, "Progress in nanoinformatics and informational materials science," *MRS Bull.*, vol. 43, pp. 690–695, 2018.
- [28] K. Hatakeyama-Sato, T. Tezuka, Y. Nishikitani, H. Nishide, and K. Oyaizu, "Synthesis of lithium-ion conducting polymers designed by machine learning-based prediction and screening," *Chem. Lett.*, vol. 48, pp. 130–132, 2019.
- [29] O. Allam, B. Cho, K. Kim, and S. Jang, "Application of DFT-based machine learning for developing molecular electrode materials in Li-ion batteries," *RSC Adv.*, vol. 8, pp. 39414–39420, 2018.
- [30] T. Parthiban, R. Ravi, and N. Kalaiselvi, "Exploration of artificial neural network [ANN] to predict the electrochemical characteristics of lithium-ion cells," *Electrochim. Acta*, vol. 53, pp. 1877–1882, 2007.

- [31] Y. Liu, B. Guo, X. Zou, Y. Li, and S. Shi, "Machine learning assisted materials design and discovery for rechargeable batteries," *Energy Storage Mater.*, vol. 31, pp. 434–450, 2020.
- [32] R. Joshi, J. Eickholt, L. Li, M. Fornari, V. Barone, and J. Peralta, "Machine learning the voltage of electrode materials in metal-ion batteries," *ACS Appl. Mater. Interfaces*, vol. 11, pp. 18494–18503, 2019.
- [33] P. Attia, A. Grover, N. Jin, K. Severson, T. Markov, and Y. Liao, "Closed-loop optimization of fast-charging protocols for batteries with machine learning," *Nature*, vol. 578, pp. 397–402, 2020.
- [34] X. Chen, X. Liu, X. Shen, and Q. Zhang, "Applying machine learning to rechargeable batteries: from the microscale to the macroscale," *Angew. Chem. Int. Ed.*, vol. 60, pp. 24354–24366, 2021.
- [35] Y. Okamoto and Y. Kubo, "Ab initio calculations of the redox potentials of additives for lithium-ion batteries and their prediction through machine learning," *ACS Omega*, vol. 3, pp. 7868–7874, 2018.
- [36] L. Kang, X. Zhao, and J. Ma, "A new neural network model for the state-of-charge estimation in the battery degradation process," *Appl. Energy*, vol. 121, pp. 20–27, 2014.
- [37] I. Haq, R. Saputra, F. Edison, D. Kurniadi, E. Leksono, and B. Yulianto, "State of charge (SoC) estimation of LiFePO₄ battery module using support vector regression," in *ICEVT & IMECE*, 2015, pp. 16–21.
- [38] D. Darbar and I. Bhattacharya, "Application of machine learning in battery: state of charge estimation using feed forward neural network for sodium-ion battery," *Electrochem.*, vol. 3, pp. 42–57, 2022.
- [39] C. Ruhatiya, S. Singh, A. Goyal, X. Niu, T. Hanh Nguyen, and V. Nguyen, "Electrochemical performance enhancement of sodium-ion batteries fabricated with NaNi_{1/3}Mn_{1/3}Co_{1/3}O₂ cathodes using support vector regression-simplex algorithm approach," *J. Electrochem. Energy Convers. Storage*, vol. 17, 2020.
- [40] J. Jo, E. Choi, M. Kim, and K. Min, "Machine learning-aided materials design platform for predicting the mechanical properties of Na-ion solid-state electrolytes," *ACS Appl. Energy Mater.*, vol. 4, pp. 7862–7869, 2021.
- [41] S. Shi, J. Gao, Y. Liu, Y. Zhao, Q. Wu, and W. Ju, "Multi-scale computation methods: their applications in lithium-ion battery research and development," *Chin. Phys. B*, vol. 25, pp. 1–25, 2016.
- [42] S. Shai and B. Shai, *Understanding machine learning: from theory to algorithms*. New York: Cambridge University Press, 2014.
- [43] A. Mansouri Tehrani, A. Oliynyk, M. Parry, Z. Rizvi, S. Couper, and F. Lin, "Machine learning directed search for ultraincompressible, superhard materials," *J. Am. Chem. Soc.*, vol. 140, pp. 9844–9853, 2018.

- [44] S. Torp, "Prediction of battery materials properties with machine learning: developing algorithms to discover electrodes for Li-ion and Mg-ion batteries," Master thesis, University of Oslo, 2020.
- [45] M. Winter, B. Barnett, and K. Xu, "Before Li-Ion batteries," *Chem. Rev.*, vol. 118, pp. 11433–11456, 2018.
- [46] N. Nitta, F. Wu, J. Lee, and G. Yushin, "Li-ion battery materials: present and future," *Mater. Today*, vol. 18, pp. 252–264, 2015.
- [47] C. Eames and M. Islam, "Ion intercalation into two-dimensional transition-metal carbides: global screening for new high-capacity battery materials," *J. Am. Chem. Soc.*, vol. 136, pp. 16270–16276, 2014.
- [48] R. Joshi, B. Ozdemir, V. Barone, and J. Peralta, "Hexagonal BC₃: a robust electrode material for Li, Na, and K- ion batteries," *J. Phys. Chem. Lett.*, vol. 6, pp. 2728–2732, 2015.
- [49] P. Bhauriyal, A. Mahata, and B. Pathak, "Hexagonal BC₃ electrode for a high-voltage Al-ion battery," *J. Phys. Chem. C*, vol. 121, pp. 9748–9756, 2017.
- [50] J. Posada, A. Rennie, S. Villar, V. Martins, J. Marinaccio, and A. Barnes, "Aqueous batteries as grid scale energy storage solutions," *Renew. Sust. Energ. Rev.*, vol. 68, pp. 1174–1182, 2017.
- [51] B. Dunn, H. Kamath, and J. Tarascon, "Electrical energy storage for the grid: a battery of choices," *Sci.*, vol. 334, pp. 928–935, 2011.
- [52] R. Guduru and J. Icaza, "A brief review on multivalent intercalation batteries with aqueous electrolytes," *Nanomater.*, vol. 6, p. 41, 2016.
- [53] V. Kulish, D. Koch, and S. Manzhos, "Ab initio study of Li, Mg and Al insertion into rutile VO₂: fast diffusion and enhanced voltages for multivalent batteries," *Phys. chem. Chem. Phys.*, vol. 19, pp. 22538–22545, 2017.
- [54] M. Thackeray, C. Wolverton, and E. Isaacs, "Electrical energy storage for transportation approaching the limits of, and going beyond, lithium-ion batteries," *Energy Environ. Sci.*, vol. 5, p. 7854, 2012.
- [55] N. Rajput, T. Seguin, B. Wood, X. Qu, and K. Persson, *Modeling electrochemical energy storage at the atomic scale*, vol. 11. Cham: Springer International Publishing, 2018.
- [56] J. Wei, X. Chu, X. Sun, K. Xu, H. Deng, and J. Chen, "Machine learning in materials science," *InfoMat.*, vol. 1, pp. 338–358, 2019.
- [57] A. Sendek, E. Cubuk, E. Antoniuk, G. Cheon, Y. Cui, and E. Reed, "Machine learning-assisted discovery of solid Li-ion conducting materials," *Chem. Mater.*, vol. 31, pp. 342–352, 2019.
- [58] R. Ramprasad, R. Batra, G. Pilania, A. Mannodi-Kanakkithodi, and C. Kim, "Machine learning in materials informatics: recent applications and prospects," *npj. Comput. Mater.*, vol. 3, p. 54, 2017.

- [59] K. Butler, D. Davies, H. Cartwright, O. Isayev, and A. Walsh, "Machine learning for molecular and materials science," *Nat.*, vol. 559, pp. 547–555, 2018.
- [60] J. Schmidt, M. Marques, S. Botti, and M. Marques, "Recent advances and applications of machine learning in solid-state materials science," *npj Comput. Mater.*, vol. 5, p. 83, 2019.
- [61] J. Schmidt, J. Shi, P. Borlido, L. Chen, S. Botti, and M. Marques, "Predicting the thermodynamic stability of solids combining density functional theory and machine learning," *Chem. Mater.*, vol. 29, pp. 5090–5103, 2017.
- [62] "<https://xenonpy.readthedocs.io>."
- [63] S. Ong, W. Richards, A. Jain, G. Hautier, M. Kocher, and S. Cholia, "Python materials genomics (pymatgen): a robust, open-source python library for materials analysis," *Comput. Mater. Sci.*, vol. 68, pp. 314–319, 2013.
- [64] M. Kusaba, C. Liu, Y. Koyama, K. Terakura, and R. Yoshida, "Recreation of the periodic table with an unsupervised machine learning algorithm," *arXiv. Org*, vol. 1912, p. 10708, 2019.
- [65] H. Liu, E. Dougherty, J. Dy, K. Torkkola, E. Tuv, and H. Peng, "Evolving feature selection," *IEEE Intell. Syst.*, vol. 20, pp. 64–76, 2005.
- [66] "<https://scikit-learn.org>."
- [67] Y. Liu, T. Zhao, W. Ju, and S. Shi, "Materials discovery and design using machine learning," *Journal of Materiomics*, vol. 3, pp. 159–177, 2017.
- [68] Y. Liu, O. Esan, Z. Pan, and L. An, "Machine learning for advanced energy materials," *Energy and AI*, vol. 3, p. 100049, 2021.
- [69] "<https://www.projectpro.io>".
- [70] "<https://www.analyticsvidhya.com/blog/author/sauravkaushik8/>".
- [71] A. Ferreira and M. Figueiredo, "Efficient feature selection filters for high-dimensional data," *Pattern Recognit. Lett.*, vol. 33, pp. 1794–1804, 2012.
- [72] S. Kauwe, J. Graser, A. Vazquez, and T. Sparks, "Machine learning prediction of heat capacity for solid inorganics," *Integr. Mater. Manuf. Innov.*, vol. 7, pp. 43–51, 2018.
- [73] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspectives," *IEEE Transact. Pattern Anal. Mach. Intell.*, vol. 35, pp. 1798–1828, 2013.
- [74] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated machine learning*. Cham: Springer International Publishing, 2019.
- [75] F. Hutter, H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," vol. 6683, Berlin: Springer, 2011, pp. 507–523.

- [76] "<https://cambridgecoding.wordpress.com>."
- [77] "<https://www.python.org>."
- [78] H. Adun, O. Bamisile, M. Mukhtar, M. Dagbasi, D. Kavaz, and A. Oluwasanmi, "Novel python-based 'all-regressor model' application for photovoltaic plant-specific yield estimation and systematic analysis," *Energy Sources A: Recovery Util. Environ. Eff.*, pp. 1–19, 2021.
- [79] M. Ghanbari and M. Goldani, "Support vector regression parameters optimization using golden sine algorithm and its application in stock market," 2021.
- [80] B. Wang and N. Gong, "Stealing hyperparameters in machine learning," *IEEE Secur. Priv.*, pp. 36–52, 2018.
- [81] P. Wang, J. Fan, Y. Ou, Z. Li, Y. Wang, and B. Deng, "A comparative study of machine learning based modeling methods for Lithium-ion battery," *IOP Conf. Ser.: Earth. Environ. Sci.*, vol. 546, p. 052045, 2020.
- [82] R. Peck, C. Olsen, and J. Devore, *Introduction to statistics and data analysis*, 5th ed. New York: Cengage Learning, 2015.
- [83] T. Li, C. Zhang, and X. Li, "Machine learning for flow batteries: opportunities and challenges," *Chem. Sci.*, vol. 13, pp. 4740–4752, 2022.
- [84] M. Bojarski, D. del Testa, D. Dworakowski, B. Firner, P. Goyal, and L. Jackel, "End to end learning for self-driving cars," 2016.
- [85] G. Fan, S. Ong, and H. Koh, "Determinants of house price: a decision tree approach," *Urban Stud.*, vol. 43, pp. 2301–2315, 2006.
- [86] D. Czerwinski, J. Gęca, and K. Kolano, "Machine learning for sensorless temperature estimation of a BLDC motor," *J. Sens.*, vol. 21, p. 4655, 2021.
- [87] "<https://pycaret.org>."
- [88] T. Hastie, R. Tibshirani, J. Friedman, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, vol. 2. New York: Springer International Publishing, 2009.
- [89] X. He, J. Luo, P. Li, G. Zuo, and J. Xie, "A hybrid model based on variational mode decomposition and gradient boosting regression tree for monthly runoff forecasting," *Water Resour. Manag.*, vol. 34, pp. 865–884, 2020.
- [90] S. Liao, Z. Liu, B. Liu, C. Cheng, X. Jin, and Z. Zhao, "Multistep-ahead daily inflow forecasting using the ERA-Interim reanalysis data set based on gradient-boosting regression trees," *Hydrology and Earth Syst. Sci.*, vol. 24, pp. 2343–2363, 2020.
- [91] J. Fan, X. Ma, L. Wu, F. Zhang, X. Yu, and W. Zeng, "Light gradient boosting machine: an efficient soft computing model for estimating daily reference

- evapotranspiration with local and external meteorological data,” *Agric. Water. Manag.*, vol. 225, p. 105758, 2019.
- [92] R. Maphanga, T. Mokoena, and M. Ratsoma, “Estimating DFT calculated voltage using machine learning regression models,” *Mater. Today: Proc*, vol. 38, pp. 773–778, 2021.
- [93] T. Oshiro, P. Perez, and J. Baranauskas, “How Many Trees in a Random Forest?,” in *Machine Learning and data mining in pattern recognition: lecture notes in computer science*, vol. 7376, Berlin: Springer, pp. 154–168, 2012.
- [94] M. Ahmad, M. Mourshed, and Y. Rezgui, “Tree-based ensemble methods for predicting PV power generation and their comparison with support vector regression,” *Energy*, vol. 164, pp. 465–474, 2018.
- [95] D. MacKay, “Bayesian interpolation,” *Comput. Neural Syst.*, vol. 4, pp. 415–447, 1992.
- [96] M. Tipping, “Sparse Bayesian learning and the relevance vector machine,” *J. Mach. Learn. Res.*, vol. 1, pp. 211–244, 2001.
- [97] “<https://scikit-learn.org>.”
- [98] R. Ron, Z. Michael, and E. Michael, “Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit,” 2008.
- [99] “<https://medium.com>.”
- [100] V. Joseph, “Optimal ratio for data splitting,” *Stat. Anal. Data Min.*, vol. 15, pp. 531–538, 2022.
- [101] C. Qi, A. Fourie, Q. Chen, and Q. Zhang, “A strength prediction model using artificial intelligence for recycling waste tailings as cemented paste backfill,” *J. Clean. Prod.*, vol. 183, pp. 566–578, 2018.
- [102] “<http://www.faqs.org/faqs/ai-faq/neural-nets/part3/section-12.html>”.
- [103] J. Rodriguez, A. Perez, and J. Lozano, “Sensitivity analysis of K-Fold cross validation in prediction error estimation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, pp. 569–575, 2010.
- [104] P. Burman, “A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning-testing methods,” *Biometrika*, vol. 76, pp. 503–514, 1989.
- [105] W. Fu, R. Carroll, and S. Wang, “Estimating misclassification error with small samples via bootstrap cross-validation,” *J. Bioinform.*, vol. 21, pp. 1979–1986, 2005.
- [106] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, and O. Grisel, “Scikit-learn: machine learning in python,” *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2012.

- [107] M. Rupp, "Machine learning for quantum mechanics in a nutshell," *Int. J. Quantum Chem.*, vol. 115, pp. 1058–1073, 2015.
- [108] Z. Yang and W. Gao, "Applications of machine learning in alloy catalysts: rational selection and future development of descriptors," *Adv. Sci.*, vol. 9, p. 2106043, 2022.
- [109] J. Ding, V. Tarokh, and Y. Yang, "Model selection techniques: an overview," *IEEE Signal Process Mag.*, vol. 35, pp. 16–34, 2018.
- [110] J. Mphaka, "Development of a mathematical model to enable optimal efficiency of the indabuko lithium-ion battery," Master thesis, University of KwaZulu-Natal, 2020.
- [111] M. Afzal, A. Sonpal, M. Haghightlari, A. Schultz, and J. Hachmann, "A deep neural network model for packing density predictions and its application in the study of 1.5 million organic molecules," *Chem. Sci.*, vol. 10, pp. 8374–8383, 2019.
- [112] C. Li, H. Hao, B. Xu, Z. Shen, and E. Zhou, "Improved physics-based structural descriptors of perovskite materials enable higher accuracy of machine learning," *Comput. Mater. Sci.*, vol. 198, p. 110714, 2021.

APPENDIX

A.1 Papers Presented at Conferences

1. K. M Monareng, P. S Ntoahae and R. R Maphanga, Machine learning models for predicting formation energy of lithium-ion battery materials, 65th Annual conference of the SA Institute of Physics, University of North-West, July 2021 **(Poster Presentation)**.
2. K. M Monareng, P. S Ntoahae and R. R Maphanga, Development of machine learning models for predicting energies of sodium-ion battery materials, 66th Annual conference of the SA Institute of Physics, University of Nelson Mandela, July 2022 **(Oral Presentation)**.
3. K. M Monareng, P. S Ntoahae and R. R Maphanga, Machine learning models for predicting the density of sodium-ion battery materials, 66th Annual conference of the SA Institute of Physics, University of Nelson Mandela, July 2022 **(Poster Presentation)**.
4. K. M Monareng, P. S Ntoahae and R. R Maphanga, Development of machine learning models for predicting density of sodium-ion battery materials, 7th CSIR Emerging Researchers Symposium, July 2022 **(Oral Presentation)**.
5. K. M Monareng, P. S Ntoahae and R. R Maphanga, Machine learning models for predicting Fermi energy of sodium-ion battery materials, 12th Postgraduate Research Day, September 2022 **(Oral Presentation)**.
6. K. M Monareng, P. S Ntoahae and R. R Maphanga, Fermi energy prediction of sodium-ion battery cathode materials: a machine learning regression approach, 67th Annual conference of the SA Institute of Physics, University of Zululand, July 2023 **(Oral Presentation)**.

A.2 Code Details

The code employed to build and authenticate the ML models is outlined in the appendix. Examples are showcased only for demonstration purposes and will not be discussed as the essential elements have already been talked about in the dissertation.

```
[ ] import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sbn
from scipy.stats import norm
import torch
%matplotlib inline

import json
import requests

data = {
    'criteria': {
        'elements': { '$all': ['Na']},
    },
    'properties': ['pretty_formula',
        'formation_energy_per_atom', 'efermi', 'final_energy_per_atom', 'density', 'e_above_hull',
        'band_gap']
}
r = requests.post('https://materialsproject.org/rest/v2/query',
    headers={'X-API-KEY': 'txxxxxxxxJ'},
    data={k: json.dumps(v) for k,v in data.items()})
response_content = r.json() # a dict

train=pd.DataFrame(response_content['response'])
```

Figure A.1: Dataset extraction.

```

import pymatgen.core as mg
from xenonpy.descriptor import Compositions
from xenonpy.datatools import preset

preset.sync('elements_completed')

fetching dataset `elements_completed` from https://github.com/yoshida-lab/dataset/releases/download/v0.1.3/elements\_completed.pd.xz.

cal = Compositions()
cal

preset.sync('elements')

fetching dataset `elements` from https://github.com/yoshida-lab/dataset/releases/download/v0.1.3/elements.pd.xz.

from xenonpy.datatools import preset
elements = preset.elements
elements.info()

df = pd.DataFrame( columns=['A', 'composition'] ) #calculating features
for i in range(train.shape[0]):
    tmp = pd.Series( [ i, (mg.Composition(train["pretty_formula"][i])) ], index=df.columns )
    df = df.append( tmp, ignore_index=True )

cal = Compositions()
comp = df['composition']
descriptors = cal.transform(comp)
descriptors.insert(0, 'pretty_formula', train['pretty_formula'])
descriptors.insert(1, 'formation_energy_per_atom', train['formation_energy_per_atom'])
descriptors.insert(2, 'density', train['density'])
descriptors.insert(3, 'e_above_hull', train['e_above_hull'])
descriptors.insert(4, 'band_gap', train['band_gap'])
descriptors.insert(5, 'efermi', train['efermi'])
descriptors.insert(6, 'final_energy_per_atom', train['final_energy_per_atom'])
descriptors.to_csv('train_descriptors.csv') #saving preprocessed data

train = pd.read_csv('train_descriptors.csv') # loading preposed data
train=train.drop_duplicates(subset='pretty_formula', keep= "first") # removing the duplication
train=train.drop_duplicates(subset='band_gap', keep= "first")
train.head() # show what the feature vectors look like. Each row is a new formula.

```

Figure A.2: Descriptor calculations.

```

    Unnamed:
    0 pretty_formula formation_energy_per_atom density e_above_hull band_gap efermi final_energy_per_atom ave:atomic_number ave:atomic_radius
0      0          Na          0.002497 1.050961 0.002497 0.0000 0.561706 -1.309726 11.000000 190.000000
17     17        NaP15         -0.072940 2.215538 0.004880 1.5192 3.726658 -5.229910 14.750000 131.875000
21     21        Na3As         -0.420880 2.339055 0.001100 0.0922 1.172908 -2.569674 16.500000 177.250000
27     27         NaN3         -0.391422 1.798138 0.000000 4.0886 -0.158743 -6.683832 8.000000 116.500000
32     32        Na3P11        -0.241579 2.046217 0.000000 1.9682 2.942440 -4.776066 14.142857 141.285714

[ ] X = train.drop(['Unnamed: 0', 'pretty_formula', 'band_gap'], axis=1)

[ ] X.head(1)

    formation_energy_per_atom density e_above_hull efermi final_energy_per_atom ave:atomic_number ave:atomic_radius ave:atomic_radius_rahm ave:atomic_volume
0          0.002497 1.050961 0.002497 0.561706 -1.309726 11.0 190.0 225.0 23.7

```

Figure A.3: Pre-processed data.

```

from pycaret.regression import * # loading all machine learning models from pycaret

train_test_split = setup(data = X, target = 'formation_energy_per_atom', session_id=123, train_size=0.70)

compare_models(round=2, fold=10)

```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
br	Bayesian Ridge	0.07	0.01	0.11	0.99	0.05	0.12	0.15
et	Extra Trees Regressor	0.09	0.02	0.14	0.98	0.06	0.22	7.54
lightgbm	Light Gradient Boosting Machine	0.08	0.02	0.13	0.98	0.05	0.19	2.06
omp	Orthogonal Matching Pursuit	0.10	0.02	0.15	0.97	0.06	0.17	0.05
rf	Random Forest Regressor	0.11	0.03	0.17	0.97	0.07	0.23	17.33

Figure A.4: Performance of the models.

```
br = create_model('br', fold = 5) # best model in terms of R2 and MSE
```

	MAE	MSE	RMSE	R2	RMSLE	MAPE
Fold						
0	0.0780	0.0172	0.1312	0.9808	0.0601	0.1502
1	0.0687	0.0108	0.1039	0.9875	0.0431	0.0825
2	0.0696	0.0113	0.1064	0.9861	0.0473	0.1140
3	0.0691	0.0107	0.1035	0.9881	0.0477	0.0934
4	0.0734	0.0125	0.1116	0.9850	0.0474	0.1794
Mean	0.0718	0.0125	0.1113	0.9855	0.0491	0.1239
Std	0.0035	0.0024	0.0104	0.0026	0.0058	0.0361

Figure A.5: Model building.

```
features = [i for i in train.columns]
```

```
pip install seaborn
```

```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(16, 8))
heatmap = sns.heatmap(train[features[1:20]].corr(), vmin=-1, vmax=1, annot=True)
heatmap.set_title('Correlation Heatmap for Sodium-ion battery materials', fontdict={'fontsize':12}, pad=12);
```

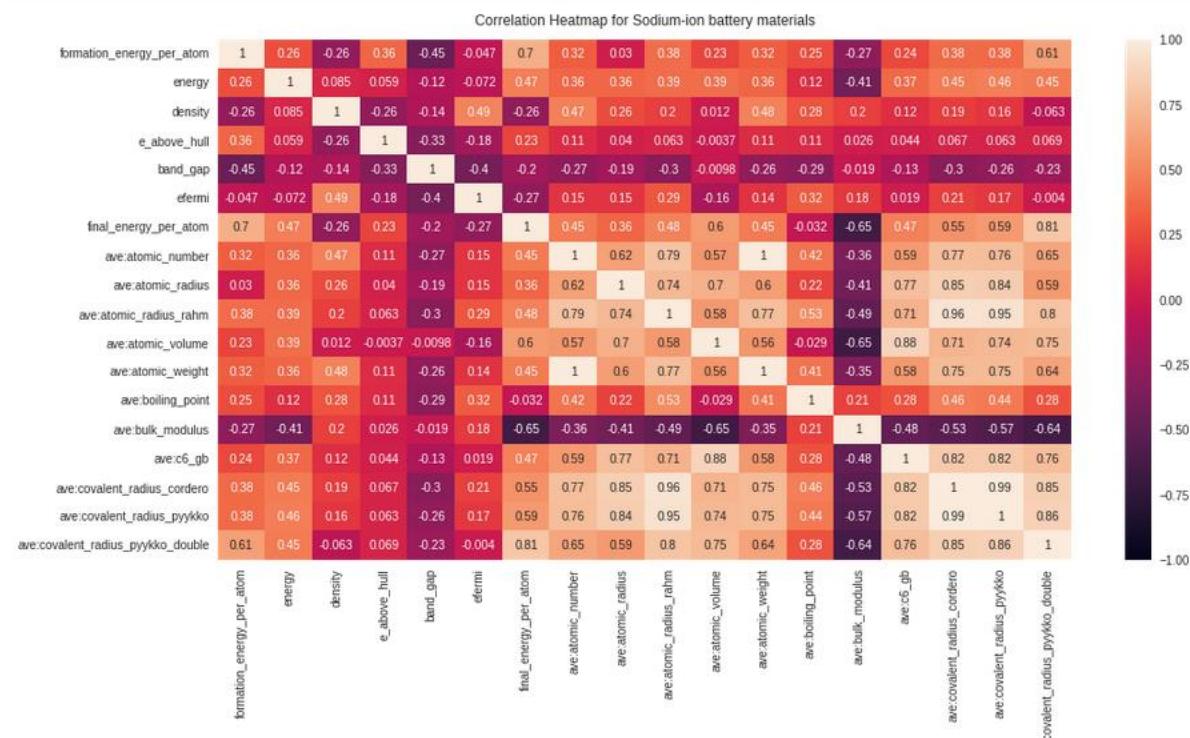


Figure A.6: Correlation heatmap for important features.

```

sample_test=pd.DataFrame()
sample_test['DFT formation energy (eV)']=predict_model(br)['formation_energy_per_atom']
sample_test['Formation energy (eV) prediction']= predict_model(br)['Label']

```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Bayesian Ridge	0.0693	0.012	0.1094	0.9869	0.0485	0.0948

Figure A.7: Bayesian ridge performance for the testing data.

```
sample_test.head(2)
```

	DFT formation energy (eV)	Formation energy prediction (eV)
0	-3.327381	-3.105658
1	-2.435405	-2.639598

Figure A.8: DFT and Machine learning formation energy comparison.

```

import seaborn as sns
plt.figure(figsize=(6,5))
ax = sns.regplot(x="DFT formation energy (eV)", y='Formation energy (eV) prediction', data=sample_test)

```

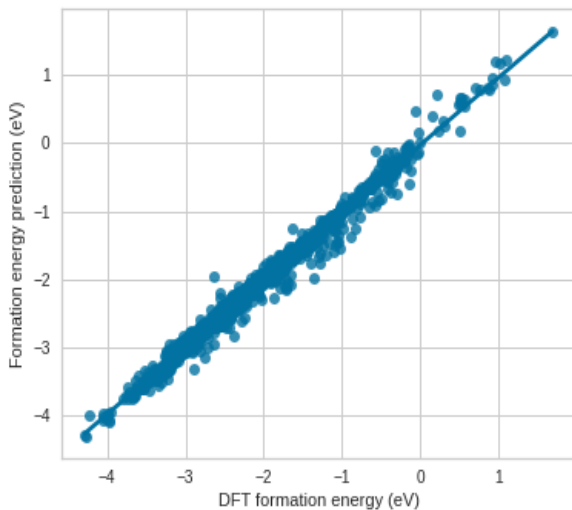


Figure A.9: Scatter plot for testing data.

```
[ ] X_train_predictions = predict_model(br, data=X.drop('formation_energy_per_atom', axis=1))
X_train_predictions['Formation energy (eV) prediction'] = X['formation_energy_per_atom'].values
X_train_predictions["DFT formation energy (eV)"] = X_train_predictions['Label']
```

```
X_train_predictions.head()
```

	density	e_above_hull	efermi	final_energy_per_atom	ave:atomic_number	ave:atomic_radius
0	1.050961	0.002497	0.561706	-1.309726	11.000000	190.000000
17	2.215538	0.004880	3.726658	-5.229910	14.750000	131.875000
21	2.339055	0.001100	1.172908	-2.569674	16.500000	177.250000
27	1.798138	0.000000	-0.158743	-6.683832	8.000000	116.500000
32	2.046217	0.000000	2.942440	-4.776066	14.142857	141.285714

Figure A.10: Model predictions on the training data.

```
import seaborn as sns
plt.figure(figsize=(6,5))
ax = sns.regplot(x="DFT formation energy (eV)", y='Formation energy (eV) prediction', data=sample_test)
```

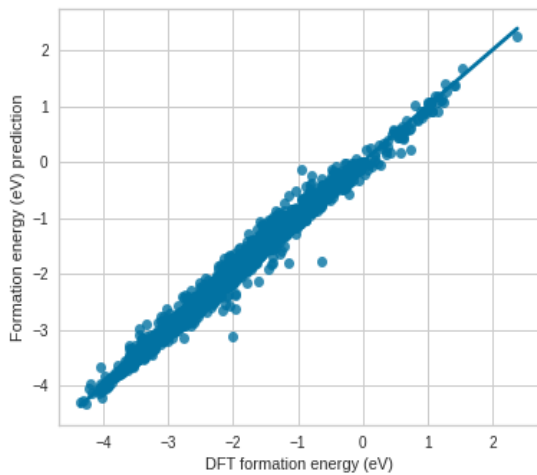


Figure A.11: Scatter plot for the training data.

```
tuned_br = tune_model(br, n_iter = 50)

X_test_predictions = predict_model(tuned_br) #final prediction
```

Figure A.12: Model tuning.

```
data = {'Models': ['BR', 'ETR', 'LGBM', 'OMP', 'RFR', 'GBR'],
        'Mean square error (MSE)': [0.01, 0.02, 0.02, 0.02, 0.03, 0.02],
        'Coefficient of determination (R2)': [0.99, 0.98, 0.98, 0.97, 0.97, 0.97]}
```

```
df = pd.DataFrame(data, columns = ['Models', 'Mean square error (MSE)', 'Coefficient of determination (R2)'])
```

Figure A.13: The dataframe for model results.

```
import seaborn as sns
import seaborn as sns; sns.set()
from matplotlib import pyplot as plt

plt.figure(figsize=(6,5))
ax = sns.lineplot(
    x='Models', y='Coefficient of determination (R2)', data=df,
    markers=True, dashes=False
)
```

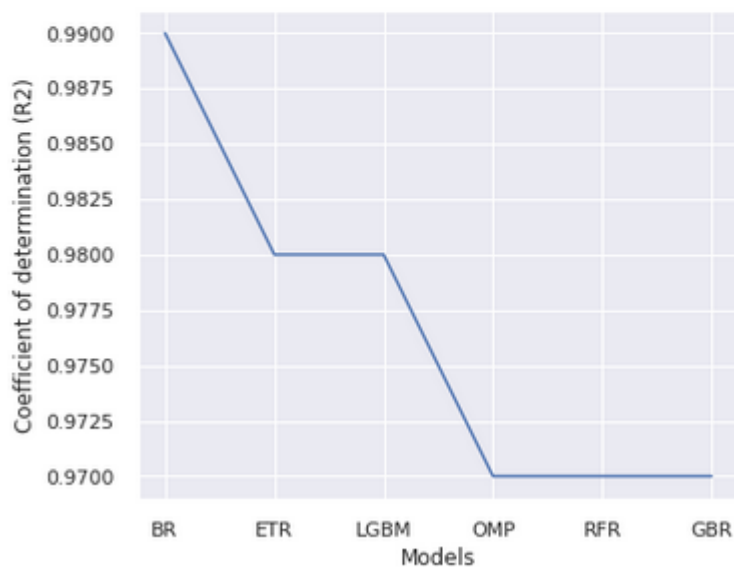


Figure A.14: Performance of the model based on regression score.


```
import seaborn as sns
import seaborn as sns; sns.set()
from matplotlib import pyplot as plt

plt.figure(figsize=(6,5))
ax = sns.lineplot(
    x='Models', y='Mean square error (MSE)', data=df,
    markers=True, dashes=False
)
```

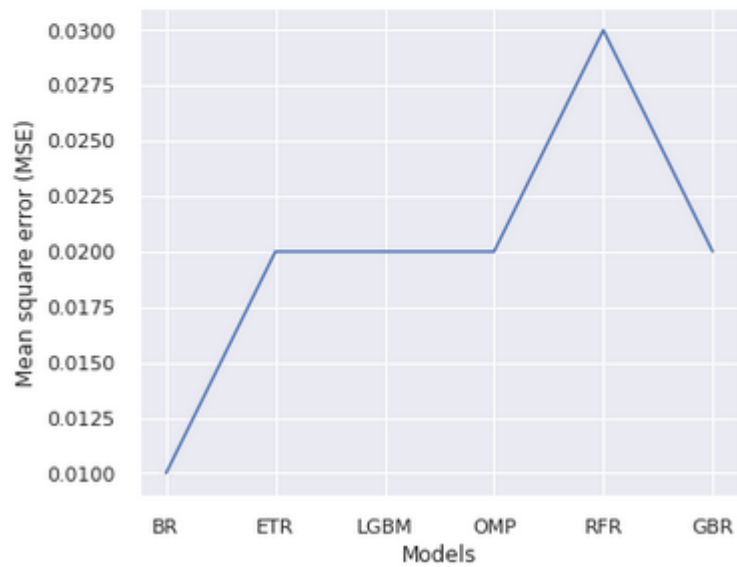


Figure A.15: Performance of the model based on mean square error.

```

import matplotlib.pyplot as plt
import seaborn as sns
#feature importance plot
dfeature_importanc=pd.DataFrame(sorted(zip(et.feature_importances_,X.columns)), columns=['Values', 'Features'])
plt.figure(figsize=(10, 5))
sns = sns.barplot(x="Values", y="Features", data=dfeature_importanc.iloc[209:227,:].sort_values(by="Values", ascending=False)).set(ylabel=None)
plt.tight_layout()
plt.ylabel("Features")
plt.xlabel("Values")
# plt.savefig('C:/Users/monareng/Desktop/feature_importan.png')
plt.show()

```

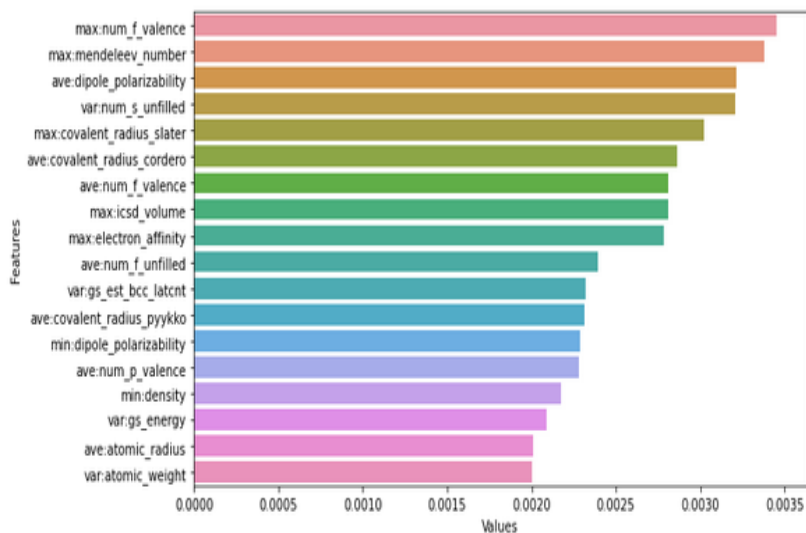


Figure A.16: Feature importance for light gradient boosting machine and extra trees regressor.

The other properties were predicted following the same procedure, with feature importance plot as illustrated by figure A.16 instead of heatmaps.