HIGH YIELD STRENGTH IN HIGH-ENTROPY ALLOYS DESIGNED WITH MACHINE-LEARNING METHODS

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

In Partial Fulfillment of the Requirements for the Degree of

Master of Science

by

Changhwi Han

May 2023

ABSTRACT

High Entropy Alloys (HEAs) have emerged as a novel paradigm in alloy design due to their superior material properties compared to the normal alloys. However, designing the HEAs with multiple metallic elements and their molar fractions is a challenging task. To resolve this issue, this project proposes a modern approach for designing new HEAs with high yield strength by utilizing machine learning techniques. The study outlines a comprehensive process for collecting and pre-processing dataset from online resources, and explores the application of five different machine learning models, two hyperparameter tuning methods, and two optimization algorithms to design HEAs. The effectiveness of each approach is evaluated based on their ability to predict the yield strength of HEAs and determine the required molar fractions of metallic elements to obtain high yield strength. The findings of this study suggest that the proposed method can serve as a valuable reference in the experimental design process of HEAs.

BIOGRAPHICAL SKETCH

Changhwi Han was born in 1990 in Tongyeong-si, South Korea. After graduating from Kyeongil High School, he served in the R.O.K. Navy before pursuing his academic and professional goals.

To earn money for college tuition, he moved to Australia and worked at a meat factory. Eventually, he moved to the United States and earned a Bachelor's degree in Mechanical Engineering from the University of California, Los Angeles in 2017.

Upon graduation, he returned to South Korea to work as a Mechanical Engineer at Samsung Electronics in the consumer electronics industry. After four years, he was admitted to Cornell University to pursue a Master of Science degree in Mechanical Engineering.

Under the guidance of Professor Jingjie Yeo, Changhwi focused on designing High Entropy Alloys using machine learning techniques to achieve the highest yield strength. He is currently applying his knowledge and skills to develop innovative solutions in the field of mechanical engineering.

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my thesis advisor, Professor Jingjie Yeo, for his exceptional support, guidance, and patience throughout the entire thesis project process. Whenever I encountered challenged or obstacles, Professor Yeo was always available in his office or via online meetings for insightful discussions and guidance. His constructive feedback and suggestions for my work have been invaluable in shaping my research project and writing. Whether he intended or not, in particular, his guidance on allowing students to conduct research independently has enabled me to develop a range of useful data analytic skills.

I am also grateful to the staff of the Department of Mechanical Engineering for their assistance, resources, and encouragement throughout my graduate studies, especially to Lataya Fann, the Graduate Field Administrator, for her support and guidance throughout the program.

Lastly, I would like to express my deep appreciation to my parents for their unwavering love, support, and encouragement throughout my academic journey. Their support during my undergraduate studies in the United States and their constant encouragement to pursue graduate school have been valuable in my success. This achievement would not have been possible without their love and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS & ABBREVIATIONS

Symbol	Remarks
%	Percentage
ΔS_{mix}	Entropy of mixing
$\sum$	Sum of numbers
B	Bulk modulus
C	Molar fractions
K	Kelvin, SI unit of temperature
R	Universal gas constant
R^2	Coefficient of determination
S	Entropy
T	Temperature
T_m	Melting temperature
x_{norm}	Normalized values
μ	Mean
ρ	Density
σ	Standard deviation
σ_y	Yield Strength

Abbreviation	Remarks
HEA	High Entropy Alloy
LAMMPS	Large-scale Atomic Molecular Massively Parallel Simulator
MPa	Mega pascal
VEC	Valence electron concentration

CHAPTER 1

INTRODUCTION

High Entropy Alloys (HEAs) have been becoming a new paradigm in the design of alloys and drawing the attention of many scientists since they were introduced in 2004 [22]. High-entropy alloys (HEAs) are a complex mixture of five or more metallic components. Each element in the HEAs shares a significant or equal portion of the mixture. They are different from conventional alloys, in which one primary element takes the most portion in the mixture. Well-designed HEAs have considerably better material properties than conventional alloys: high yield strength, high fatigue resistance, high ultimate tensile strength, and good corrosion resistance [19]. However, finding those well-designed HEAs are challenging. With the number of metallic elements with their molar fractions, there can be more than a thousand ways to design their combinations to achieve desired material properties [6]. Experimentally fabricating these combinations, testing, and verifying their material properties can be very impractical [6].

In order to address this problem, various strategies have been proposed to efficiently produce high-entropy alloy (HEAs), including: utilizing combinatorial materials synthesis techniques, employing computational materials science approaches such as ab initio simulations, molecular dynamics (MD), finite element, New PHACOMP, and CALPHAD methods, and implementing the Taguchi method to optimize properties [21]. Instead of those methods, in this study, we propose to use machine learning techniques for the efficient design of HEAs; it can minimize the trial-and-error process and reduce time and cost [6]. Specifically, we use five machine learning regression models, namely Random Forest, AdaBoost, Support Vector Machine, GradientBoost, and Neural Network, to predict the yield strength of TiNbMoTaW, NbMoTaW, TiVNbMoTaW, and VnbMoTaW at various temperatures ranging from 25 °C to 1200 °C [13]. Based on the features

used in the models, we predict the optimal molar fractions of metallic elements for achieving the highest possible yield strength using two genetic algorithm optimization methods: Differential evolution and Maxlipo. To validate our predictions, we compare them with experimental data from previous studies; if such data is unavailable, we use the LAMMPS molecular dynamics simulator, an open-source software, to obtain the yield strength of the predicted molar fractions of metallic elements.

The following METHODOLOGY section details the process for HEAs, including data preprocessing and machine learning model development. In the subsequent RESULTS AND DISCUSSION section, we present model performance for various machine learning models and their respective prediction results. Furthermore, we optimize molar fractions of metallic elements to obtain maximized yield strength values, and validate these results using LAMMPS. Finally, the CONCLUSIONS section summarizes the key findings of this project, outlines any limitations, and suggests potential opportunities for future research.

CHAPTER 2

METHODOLOGY

We combined data from two online sources [6, 11] to create a raw dataset of material properties and yield strength for HEAs. After cleaning and pre-processing the dataset (handling missing values, outliers, normalization, and feature selection), we performed hyperparameter tuning to optimize machine learning models foryield strength prediction. Then, features with high importance (i.e. metallic elements contributing the most to yield strength) were extracted from the models. Then, we built an object function (i.e. machine learning model in this case) to run optimization algorithms — two Genetic algorithms (i.e. Differential evolution and Maxlipo) — to predict the best molar fraction combinations of the metallic elements for maximum yield strength. Verification was done by generating stress-strain curves and comparing yield strength values from uniaxial tension deformation of HEAs in LAMMPS. Details on pre-processing the dataset, feature selection, hyperparameter tuning, and LAMMPS implementation can be found in the section below.

2.1. Data selection

The dataset in this project was collected from other publications including molar fractions, testing temperature, and corresponding yield strength of HEAs [6, 11] shown in Table 1. To establish a concrete dataset, a few more features of HEAs were added such as bulk modulus (B), melting temperature (T_m), valence electron concentration (VEC), entropy (S), and density (ρ). The added features were calculated by using the below equations (1) – (5),

Table 1 Collected Raw Dataset of High Entropy Alloys with Mechanical Properties [6, 11]

	%Al	%Co	%Cr	%Fe	%Ni	%Cu	%Mn	%Hf	%Nb	%Ta	...	%W	%Re	%C	Bulk modulus	Melting temperature	VEC	Testing temperature	Entropy	Density	Yield strength
0	5.40	0.0	0.0	0.0	0.0	0.0	0	0.0	25.5	0.00	...	0.0	0	0	164.5140	2387.0950	4.9220	300.0	12.632656	7.22646	1250.0
1	5.10	0.0	0.0	0.0	0.0	0.0	0	0.0	25.0	22.50	...	0.0	0	0	154.7160	2451.5750	4.6480	300.0	12.601890	8.52459	1330.0
2	5.00	0.0	0.0	0.0	0.0	0.0	0	0.0	26.0	28.90	...	0.0	0	0	146.3000	2528.3310	4.4990	300.0	12.493242	9.36723	1745.0
3	5.20	0.0	0.0	0.0	0.0	0.0	0	0.0	24.8	25.80	...	0.0	0	0	142.5130	2476.8450	4.4540	1273.0	12.592128	8.95963	366.0
4	4.76	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	23.81	...	0.0	0	0	170.2876	2501.2218	4.9048	300.0	12.568827	9.05727	1021.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
352	0.00	0.0	0.0	0.0	0.0	0.0	0	0.0	20.0	0.00	...	0.0	0	0	138.2000	2239.0000	4.6000	1275.0	11.076269	6.36000	72.0
353	0.00	0.0	0.0	0.0	0.0	0.0	0	0.0	25.0	0.00	...	0.0	0	0	132.7500	2252.5000	4.5000	300.0	11.526206	6.42250	1105.0
354	0.00	0.0	0.0	0.0	0.0	0.0	0	0.0	25.0	0.00	...	0.0	0	0	132.7500	2252.5000	4.5000	875.0	11.526206	6.42250	834.0
355	0.00	0.0	0.0	0.0	0.0	0.0	0	0.0	25.0	0.00	...	0.0	0	0	132.7500	2252.5000	4.5000	1075.0	11.526206	6.42250	187.0
356	0.00	0.0	0.0	0.0	0.0	0.0	0	0.0	25.0	0.00	...	0.0	0	0	132.7500	2252.5000	4.5000	1275.0	11.526206	6.42250	58.0

$$B = \sum_i^n (C_i B_i) \quad (1)$$

$$T_m = \sum_i^n C_i (T_m)_i \quad (2)$$

$$VEC = \sum_i^n C_i (VEC)_i \quad (3)$$

$$\Delta S_{mix} = -R \sum_i^n C_i (lnC_i) \quad (4)$$

$$\rho = \sum_i^n C_i \rho_i \quad (5)$$

where C and B were molar fractions and bulk modulus: T_m and VEC were melting temperature and valence electron concentration: ΔS_{mix} and ρ were the entropy of mixing and the density: R was the universal gas constant. The currently collected dataset consisted of 24 dependent variables and one independent variable (i.e. yield strength), and each variable had 356 instances.

2.2. Data preprocessing

2.2.1. Missing Values

Missing values refer to the data values that are not to be stored in the dataset. Missing values could lead to wrong results of prediction since the remaining data might weaken the representative of the dataset and cause bias. In the project, the missing values were treated in two ways; the features that had more than three missing values were removed from the data set (i.e. Mn, Re, and C); for

the features with less than three missing values, those missing values were replaced with the mean value (i.e. Fe, Ni, and Cu).

2.2.2. Feature Normalization

The dataset was normalized by using z-score normalization to understand the range of raw data points better, representing them in a range of 0 and 1, shown in equation (6) where μ was the mean, and σ was the standard deviation [12]. The normalization was not only useful for understanding the complexity of the data but also for reducing computational costs when training the model because all features were represented in a range of 0 and 1 [12].

$$x_{norm} = \frac{x - \mu}{\sigma} \tag{6}$$

2.2.3. Skewed to Normal distribution

The normal or gaussian distribution is popular in practice to fit the dataset with the statistical models to predict continuous outcomes more accurately. Oftentimes, however, many experimental datasets are in a skewed distribution. For the variables in skewed distribution, thus, they were transformed to a normal distribution by using log transformation [3].

2.2.4. Outliers

Outliers refer to the data points far away from the majority of the data points. It may occur when there are experimental or human errors during collecting data. Outliers are called noise in the data set and badly affect the average value of the dataset and prediction accuracy. To detect the outliers, the Z-score method was used, which also can measure the skewness of the data. To decrease the

affection of outliers, we can use the technique called the Quantile-based flooring and capping method; the outliers whose z-score was above positive three or below negative three were replaced with upper and lower limits that a z-score was three sigmas [12].

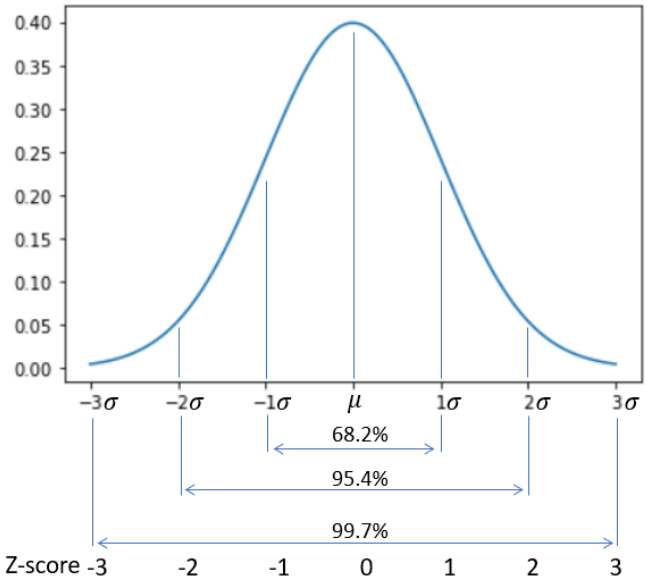

Figure 1 Z-Score Distribution

2.2.5. Feature selection

The feature selection was performed based on the Pearson correlation scores [1]. The features that showed high scores to each other are to be removed to prevent the model from overfitting; the higher score was, the more linearity between the two features was [1]. Having more features that were in high linearity helped to improve the model performance with the validation dataset. When it came to the unseen dataset, however, having those features degraded the model performance [1]. Therefore, removing those features was necessary to prevent the overfitting issue and reduce the feature complexity and computational cost [1]. The feature selection in the second topic was performed by using the “argsort” method, a built-in Python package. The result gave the list of

features in order of importance. In other words, we detected what features contributed to the yield strength the most. From the feature list, we could ideally select the top four or five features (i.e. metallic elements) to train the model and run the optimization method to find the best molar fractions of the metallic elements to obtain the maximum yield strength. In this project, however, we selected three metallic elements due to some restrictions in the LAMMPS simulation, those restrictions were discussed in the Results and Discussion section.

2.3. Machine Learning Models

2.3.1 Model Selection

The machine learning models used for both topics are Random Forest, AdaBoost, Support Vector Machine, GradientBoost, and Neural Network to test as many machine learning models as possible. Firstly, Random Forest Regression was a type of supervised learning that utilized an ensemble learning approach for regression [4]. This model combined the predictions of multiple machine learning models to achieve a more precise prediction than that of a single model [4].

Figure 2 depicted the layout of different machine learning algorithms. First of all, the Random Forest was the algorithms that the trees run independently without any interaction between them. During the training process, a Random Forest generated numerous decision trees and calculated the average class value as the prediction for all the trees [4].

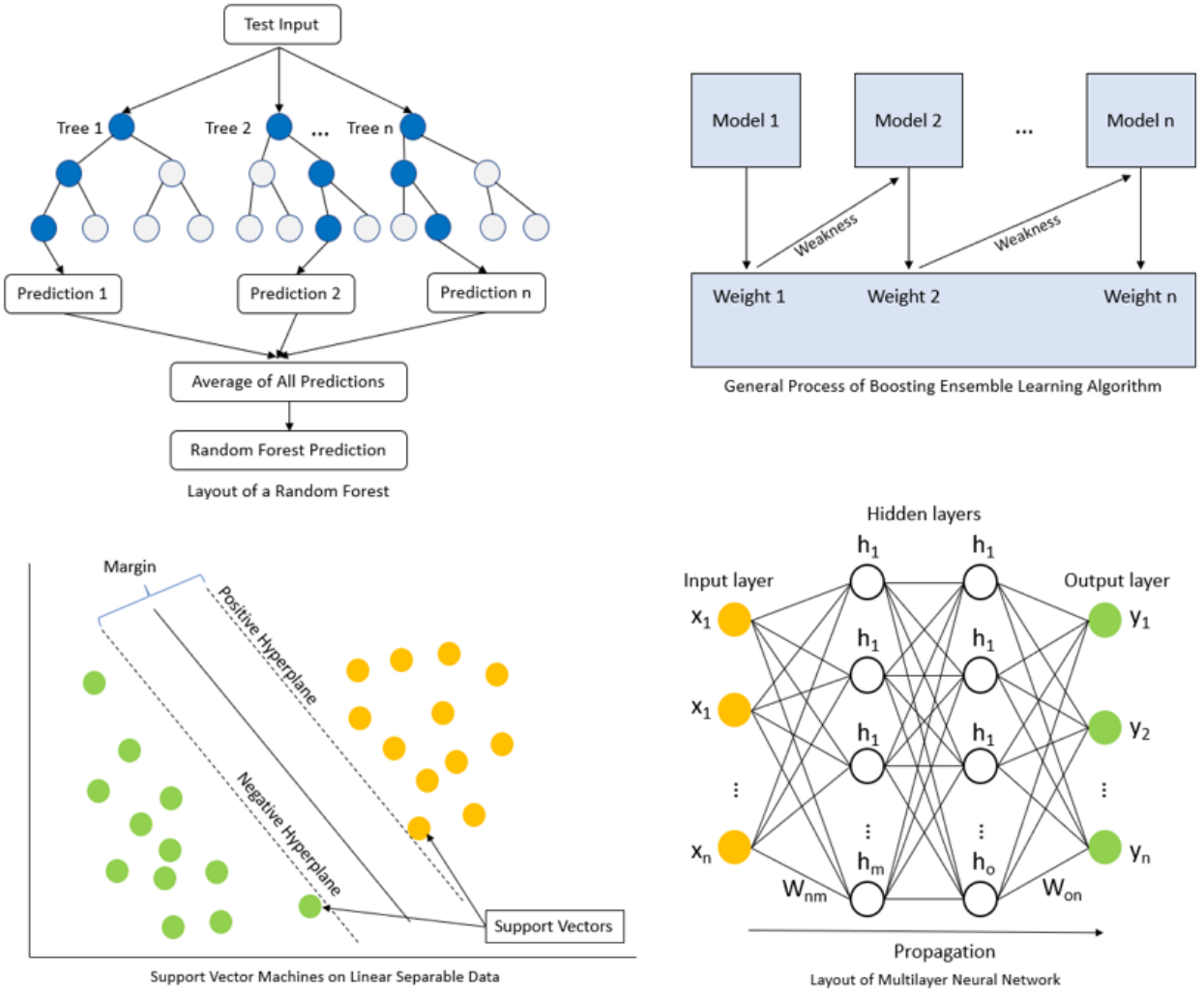

Figure 2 Layout of Different Machine Learning Algorithms

Secondly, AdaBoost was a boosting algorithm that shared similarities with Random Forest but also possessed some notable distinctions [17]. Instead of building a collection of decision trees, Ada Boost constructed a collection of decision stumps. Distinct weights were assigned to each decision stump for the ultimate decision-making process [17]. Thirdly, the Support Vector Machine aimed to identify a hyperplane in an n-dimensional space (where n represented the number of features) that can segregate data points into distinct classes [17]. This was accomplished by maximizing the margin or distance between the classes [17]. Support Vectors were data points

that were closest to the hyperplane aiding in the maximization of the margin between classes [17]. The dimensions of the hyperplane are determined by the number of input features[17]. Forth, Gradient Boosting built numerous decision trees with each tree learning from the error of previous trees [17]. It utilized the residual error to enhance prediction performance to minimize the residual error [17]. While similar to AdaBoost, Gradient Boosting differed in that it constructed decision trees with multiple leaves instead of decision stumps [17]. The process began by creating a base decision tree with initial predictions, usually the average [17]. Another decision tree was built with initial features and residual error as dependent variables. The prediction for the new tree was made by adding the initial prediction of the model with the residual error for the sample multiplied by the learning rate [17]. This process was repeated until minimal error was achieved [17]. Lastly, the Neural Network model was a collection of algorithms that aimed to recognize patterns and correlations within data, resembling the functioning of the human brain [14]. It fell under the category of machine learning and features a network of neurons capable of processing complex data sets that may be computationally demanding [14].

2.3.2 Model Hyperparameter Tuning & Optimization Algorithm

In the model training process, the dataset was split into the training set and test validation set. The training dataset contained 85% of the entire dataset to train the model, and the testing dataset contained 15% of the entire dataset to test the model performance [23]; this dataset split ratio was chosen since this ratio resulted in the best model accuracy (7) from the range of the data split ratio: 70% vs 30%, 80% vs 20%, 85% vs 15%, and 90% vs 10%. The training and test dataset was randomly shuffled. The model hyperparameters for each model were selected based on the model performance from using two hyperparameters tuning Python packages: RandomSearch method

and Bayesian optimization method. The RandomSearch method searched for the best hyperparameter sets within the user-defined number of hyperparameter sets at random [16]. Unlike the RandomSearch method, which searched for the hyperparameter sets independently, Bayesian Optimization method learned from the previous iterations in a search of hyperparameter sets [16]. The hyperparameter sets used in the hyperparameter tuning process was listed in Table 2. Due to the small amount of training data, the method of k-fold cross-validation was selected to create a model that would have the best overall performance among all of the validation folds [1]. The following hyperparameters were discovered during the tuning process for each model:

- Random Forest: n_estimators = 550, min_samples_split = 2, min_samples_leaf = 1, max_features = "auto", max_depth : 90, bootstrap : True
- AdaBoost: n_estimators = 500, loss = "linear", learnin_rate = 0.1
- SVR: kernel = "linear", C = 10, gamma = "scale", epsilon = 0.1
- Gradient Boosting: n_estimators = 500, min_samples_split = 2, min_samples_leaf = 5, max_features = "auto", max_depth = 5, loss = "huber"
- Neural Network: activation = "relu", alpha = 0.0001, batch_size = 16, hidden_layer_sizes = (10, 50, 30), learning_rate = "constant", max_iter = 500, solver : "adam"

The overall model performance of each hyperparameter tuning method was taken from the average of randomly shuffled training and test datasets by assigning 'Random_state' to ten different numbers. The measure of the model performance used the accuracy equation shown in Equation (7) where it took the mean of the absolute value of prediction of x_test data minus y_test data divided by y_test data (i = 1,2,…,n). In the second topic, two Genetic algorithms (i.e. Differential evolution and Maxlipo) were used. The Differential evolution generated multiple potential best

solutions and then gradually enhanced them by repositioning them within the search area to process the best solution [18]. The Maxlipo was a comparable approach to Differential evolution, but with an extra supposition that the function was continuous [18]. This was an advanced method than the Differential evolution since this method could find optimal values with less usage of the function, which could save a lot of time [18]. They were found in the Python package that allowed us to solve an optimization problem by iteratively examining a range of possible solutions for an objective function to yield maximum or minimum value [2]. In this project, they optimized the required molar fractions of metallic elements to obtain the maximum yield strength value.

$$Accuracy = \frac{100}{\mathrm{n}}\left(\sum_{i}^{n}\frac{\left|predict_{x\_test_i} - y\_test_i\right|}{y\_test_i}\right) \tag{7}$$

Table 2 Lists of the Hyperparameters used in the Hyperparameter Tuning Process for Machine Learning Models

Random Forest
n_estimators = [100, 200, 250, 300, 350, 400, 450, 500, 550, 600 , 700, 800]
max_features = ['auto', 'sqrt']
max_depth = [None, 10, 20, 30, 40, 50, 60, 70, 80, 90]
min_samples_split = [2, 4, 6, 8]
min_samples_leaf = [1, 2, 3, 4, 5]
bootstrap = [True, False]
AdaBoost
n_estimators = [10, 20, 50, 100, 200, 500]
loss = ['linear', 'square', 'exponential']
learning_rate = [0.01, 0.1, 1.0]

SVR
kernel': ['linear', 'rbf']
C':[1, 3, 5, 10]
gamma': ['scale', 'auto']
epsilon':[0.1, 0.3, 0.5]
Gradient Boosting
loss = ['squared_error', 'absolute_error', 'huber', 'quantile']
n_estimators = [100, 200, 300, 500]
max_depth = [3, 5, 10]
min_samples_leaf = [1, 3, 5]
min_samples_split = [2, 4, 6]
max_features = ['auto', 'sqrt', 'log2']
Neural Network
hidden_layer_sizes = [(50,50,50), (50,100,50), (100,50,30)]
max_iter = [500]
activation = ['tanh', 'relu']
solver = ['adam']
alpha = [0.0001, 0.05]
learning_rate = ['constant','adaptive']
batch_size = [16, 32]

2.4. LAMMPS

To validate the result from the second topic — the maximum yield strength from the optimized molar fractions of the metallic elements —, LAMMPS was used, an open-source molecular dynamics simulation software. In the process of writing the input script to run the simulation, obtaining the EAM potential files was necessary: AlNbTi [7], FeCrW [9], FeCuNi [10], and FeNiCr [20] were used. In the input source file, five thousand atoms were created randomly and they were evenly distributed by the metallic elements within the HEA. In the case of FeNiCr, there were 1666 atoms created for Fe, Ni, and Cr respectively. The simulation was conducted under the room temperature of 300K and pressure of one atm with rectangular-shaped nanowires shown in Figure 25. To obtain the yield strength, from the simulation, the strain-stress curve of the HEA was obtained first. The uniaxial tension deformation was performed under the pressure of one atm and a temperature of 300 K. The deformation proceeded up to 12 percent of the original length of the alloy. Then, yield strength was obtained from the curve. The yield strength was the stress value that change in length of the element under the stress was 0.2 percent of the stretched length. Then, the yield strength values from the LAMMPS simulation were compared to the yield strength values from the optimization.

CHAPTER 3

RESULTS AND DISCUSSION

Based on the Pearson Correlation Maxtrix in Figure 3, Bulk modulus, and VEC columns were removed as their linearity was close to one, which could potentially cause an overfitting problem. The overfitted model was not appropriate for predicting unseen datasets [1]. After the data cleaning process, the dataset that was ready for training the models, which was shown in Table 3.

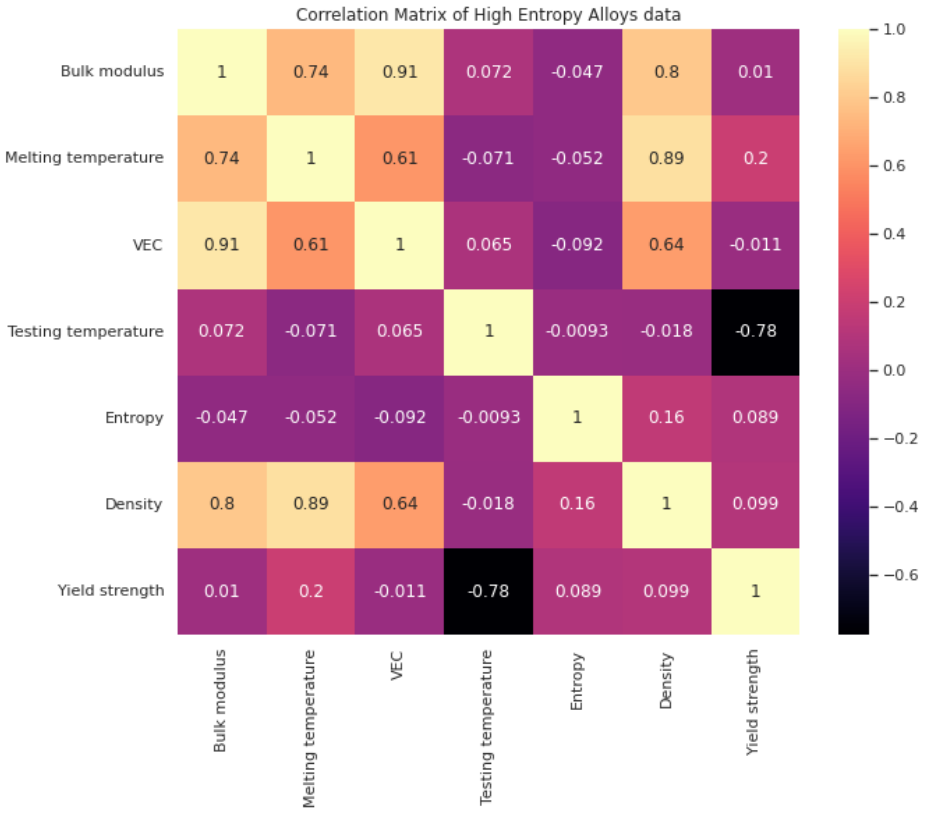

Figure 3 Pearson Correlation Matrix of High Entropy Alloys Data

Table 3 Pre-Processed Dataset of High Entropy Alloys with Mechanical Properties

	%Al	%Co	%Cr	%Fe	%Ni	%Cu	%Hf	%Nb	%Ta	%Ti	%Zr	%V	%Mo	%W	Melting temperature	Entropy	Density	Yield strength
0	-0.14	-0.2	-0.52	-0.16	-0.16	-0.16	-0.61	0.77	-0.91	0.34	-1.22	1.26	1.09	-0.32	0.03	-0.39	-0.54	1250.0
1	-0.17	-0.2	-0.52	-0.16	-0.16	-0.16	-0.61	0.70	1.39	0.69	-1.22	1.29	-0.83	-0.32	0.25	-0.42	0.15	1330.0
2	-0.18	-0.2	-0.52	-0.16	-0.16	-0.16	-0.61	0.85	2.04	0.24	0.65	-0.80	-0.83	-0.32	0.52	-0.52	0.61	1745.0
4	-0.21	-0.2	-0.52	-0.16	-0.16	-0.16	-0.61	-2.95	1.52	0.55	-1.22	1.42	1.00	-0.32	0.43	-0.45	0.44	1021.0
5	-0.13	-0.2	-0.52	-0.16	-0.16	-0.16	1.68	-0.19	1.02	-0.02	0.64	-0.80	-0.83	-0.32	0.21	1.23	0.68	1188.0
6	-0.17	-0.2	-0.52	-0.16	-0.16	-0.16	-0.61	0.29	0.38	1.01	1.52	-0.57	-0.83	-0.32	-0.41	-0.15	-0.48	1965.0

3.1. Hyperparameter Model Performance

Figure 4 showed the model performance as the hyperparameter tuning was processed for the five models: Random Forest, AdaBoost, Support Vector Machine, GradientBoost, and Neural Network. Among the five models, the Random Forest showed the best accuracy, 92%, followed by Neural Network, 87%, and GradientBoost, 74%. This indicated that the Random Forest model was suitable for the dataset with many features and small instances. When it came to selecting the hyperparameter tuning method, unlike the three least performed models, the RandomSearch method outperformed the Bayesian Optimization method by about 15% in the top two performed models: Random Forest, and Neural Network. In general, the Bayesian Optimization method was known as a better method than the RandomSearch method when it came to hyperparameter tuning although it was computationally expensive [16]. But, the performance also varied as the value of k-fold cross-validation changed. In the RandomSearch case, 2-fold cross-validation was used while 5-fold cross-validation was used in the Bayesian Optimization tuning case. This result gave intuition that small k-fold cross-validation during the hyperparameter tuning process gave a better model performance for this specific dataset. Figure 4 showed that the RandomSearch method performed better than another one.

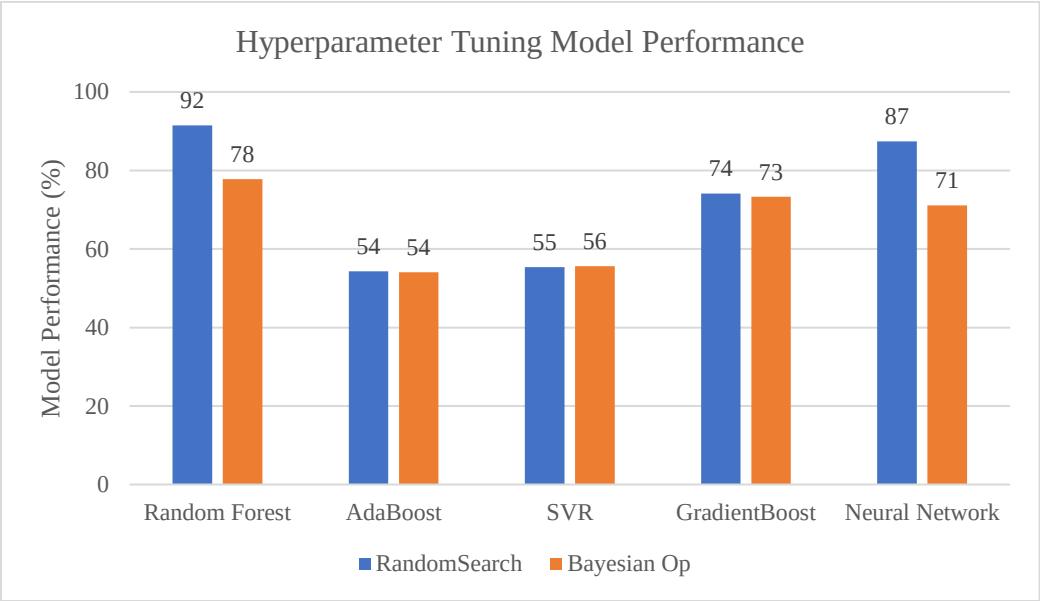

Figure 4 Hyperparameter Tuning Model Performance between RandomSearch and Bayesian Optimization for the Five Machine Learning Models

In the process of hyperparameter tuning for the Neural Network model, in Figure 5, the cost rapidly dropped and converges to zero as running iterations. In Figure 6, with tuned hyperparameters, the predicted and actual values resulted in a linear relationship, which visually showed the accuracy of the Neural Network model.

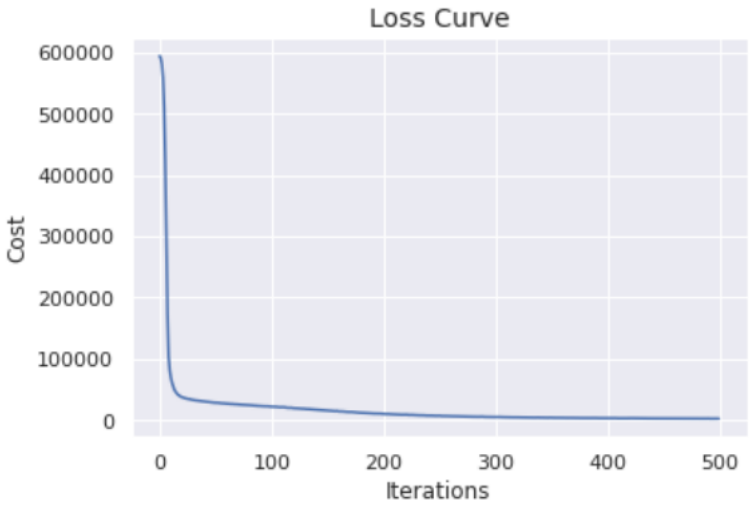

Figure 5 Error Curve per Iterations in Tuning Neural Network Model

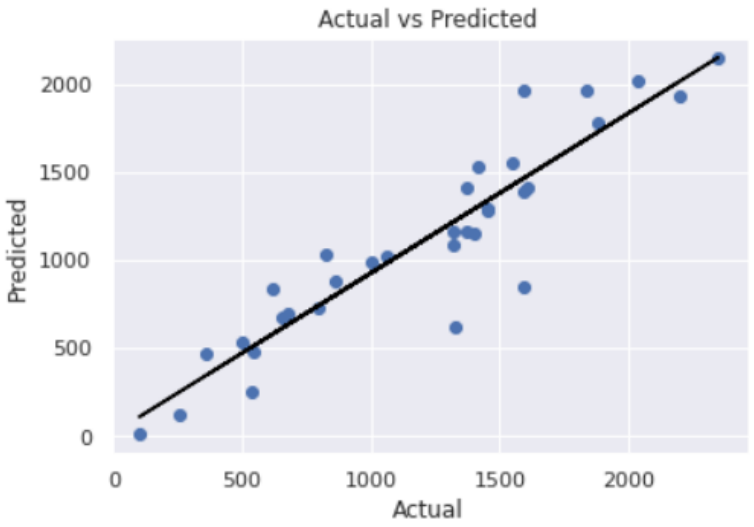

Figure 6 Prediction Results Compared to Actual with Tuned Hyperparameters in Neural Network Model

3.2. Prediction Error of HEAs in the Five Machine Learning Models

Figure 7 – 11 showed error between the predicted and the experimental yield strength values of TiNbMoTaW, NbMoTaW, TiVNbMoTaW, and VNbMoTaW at 25 °C, 600 °C, 800 °C, 1000 °C, and 1200 °C [13]. The experimental yield strength values were shown in Table 4. The Random Forest Regressor model gave error as low as 0.5% and as high as 39.5% followed by 1.7% & 56.3% in the AdaBoost Regressor model, 4.3% & 75.4% in the Support Vector Regressor model, 0.9% & 50.1% in Gradient Boosting Regressor model, and 1.9% & 63.7% in Neural Network Regressor model respectively. The best-performing model in predicting the unseen test dataset was the Random Forest Regressor model, and the worst was the Support Vector model: there were eight cases in total the error was less than 10% in the Random Forest model, four in AdaBoost Regressor, two in Support Vector Regressor, four in Gradient Boosting Regressor, and three in Neural Network model.

Table 4 Yield strength, peak strength, and plastic strain at different temperatures of the TiNbMoTaW, NbMoTaW, TiVNbMoTaW, and VnbMoTaW HEAs [13]

Alloy/properties		TiNbMoTaW	NbMoTaW	TiVNbMoTaW	VNbMoTaW
25 °C	σ_y (Mpa)	1343	1058	1515	1246
600 °C	σ_y (Mpa)	689	561	973	862
800 °C	σ_y (Mpa)	674	552	791.3	846
1000 °C	σ_y (Mpa)	620	548	752.8	842
1200 °C	σ_y (Mpa)	586	506	659	735

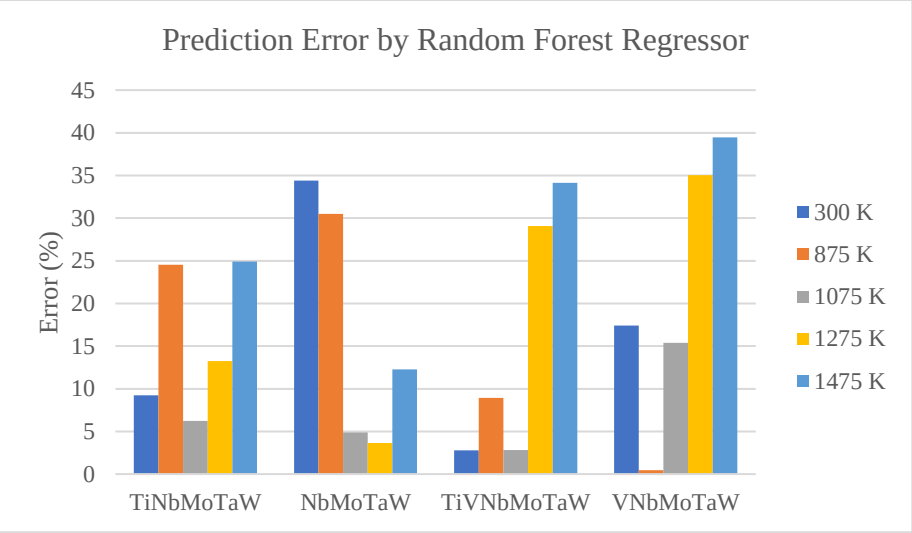

Figure 7 Prediction Error using Random Forest Regressor for Four HEA Alloys in Different Temperatures

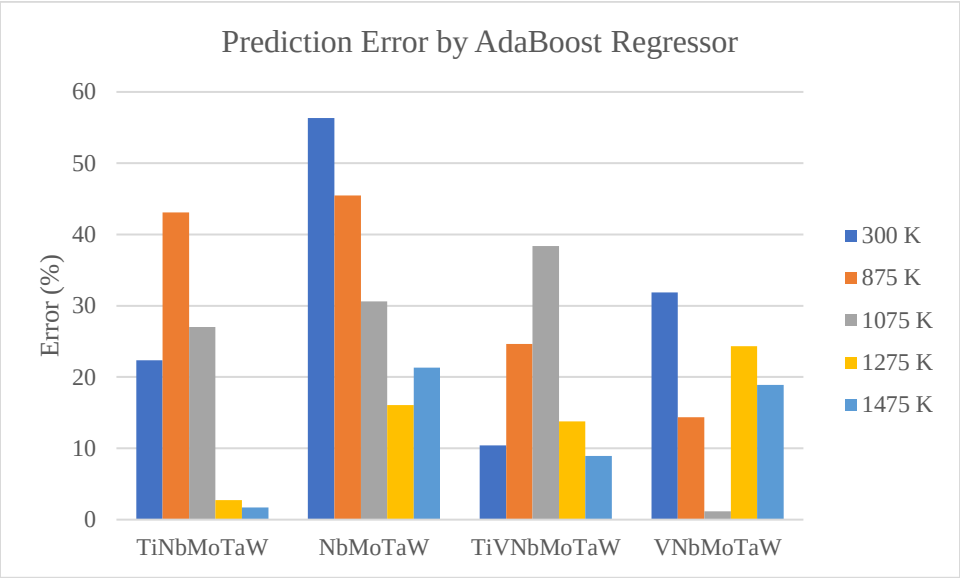

Figure 8 Prediction Error using AdaBoost Regressor for Four HEA Alloys in Different Temperatures

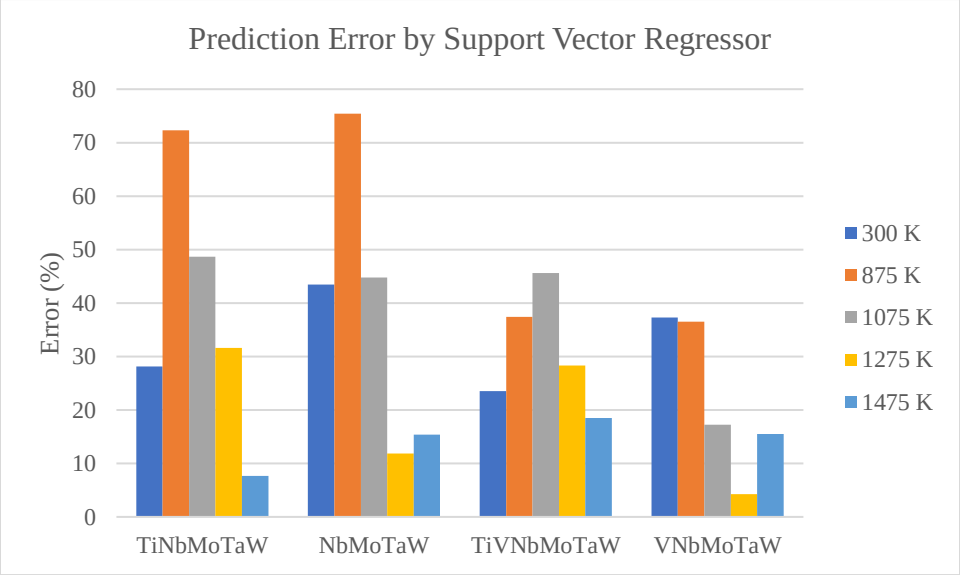

Figure 9 Prediction Error using Support Vector Regressor for Four HEA Alloys in Different Temperatures

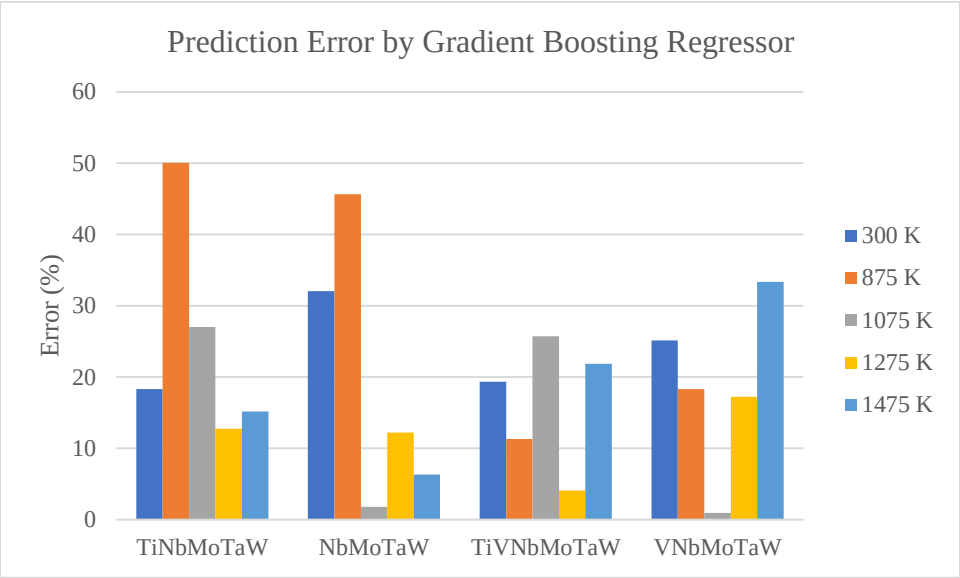

Figure 10 Prediction Error using Gradient Boosting Regressor for Four HEA Alloys in Different Temperatures

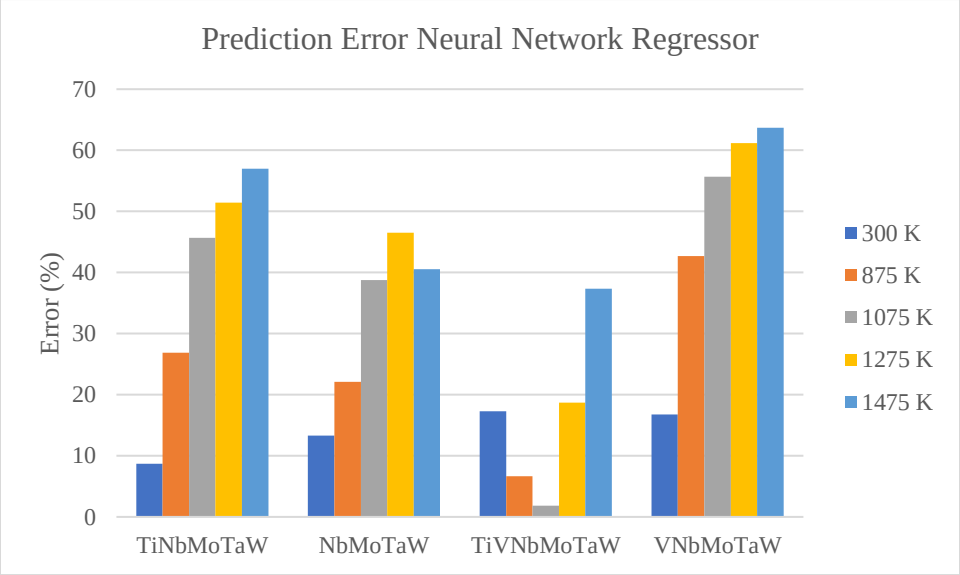

Figure 11 Prediction Error using Neural Network Regressor for Four HEA Alloys in Different Temperatures

To look closer into detail on the prediction results, the dataset was sorted into a new dataset the testing temperature was 300K but the rest were removed; Figure 12 showed the data points distribution by testing temperature, and that the majority of the data points were in 300K. With the

newly sorted dataset, yield strength values were predicted again for TiNbMoTaW, NbMoTaW, TiVNbMoTaW, and VnbMoTaW with the five different models, showing their results in Figure 13 – 17. The previous model-building process, Figure 7 - 11, used one Random_state value that resulted in the best accuracy among ten Random_state values when the model was built; Random_state was the parameter in Python programming that allowed the dataset to get randomly shuffled in a certain order. Unlike the previous case, Figure 7 – 11 where the prediction was only performed on one Random_state value, Figure 13 – 17 showed error from prediction results in the five different Random_state values. Except for the Support Vector Regressor model Figure 15 which the error was bad for all alloys, all the other models showed that the error was substantially large in NbMoTaW alloy. This probably occurred because the number of data points of NbMoTaW was the smallest among all the alloys. When it came to training and testing the machine learning models in this case, the larger dataset gave a better model performance and prediction results. The number of the data points for the alloy was strictly proportional to the number of the metallic elements within the alloy, thus, the more metallic elements in the alloy in this dataset gave better accuracy in predictions: the data points in TiVNbMoTaW were significantly outnumbered NbMoTaW alloy, and the prediction error of TiVNbMoTaW, 5%, was far less than NbMoTaW, 35%, in the Random Forest model Figure 13; other models showed the same trend in the three other models Figure 14, 16, and 17.

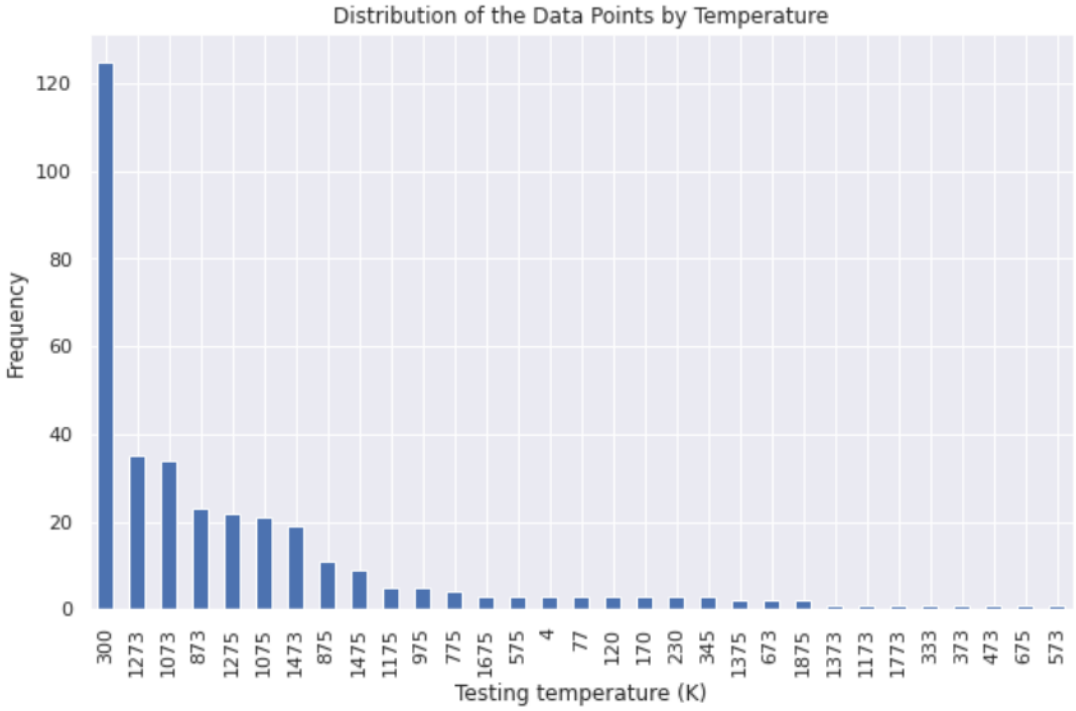

Figure 3 Data Distribution of the Dataset by Testing Temperature

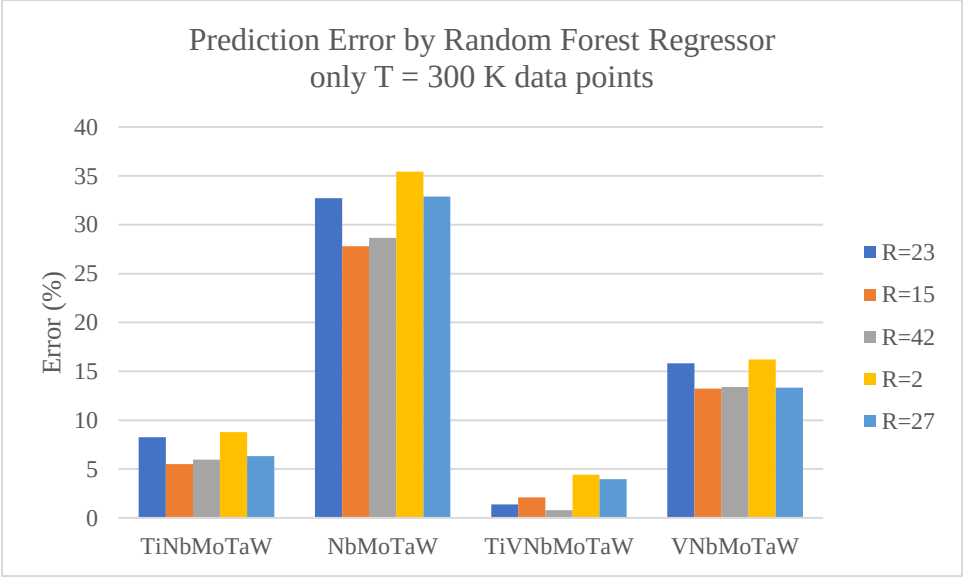

Figure 4 Prediction Error using Random Forest Regressor for Four HEA Alloys in various R values

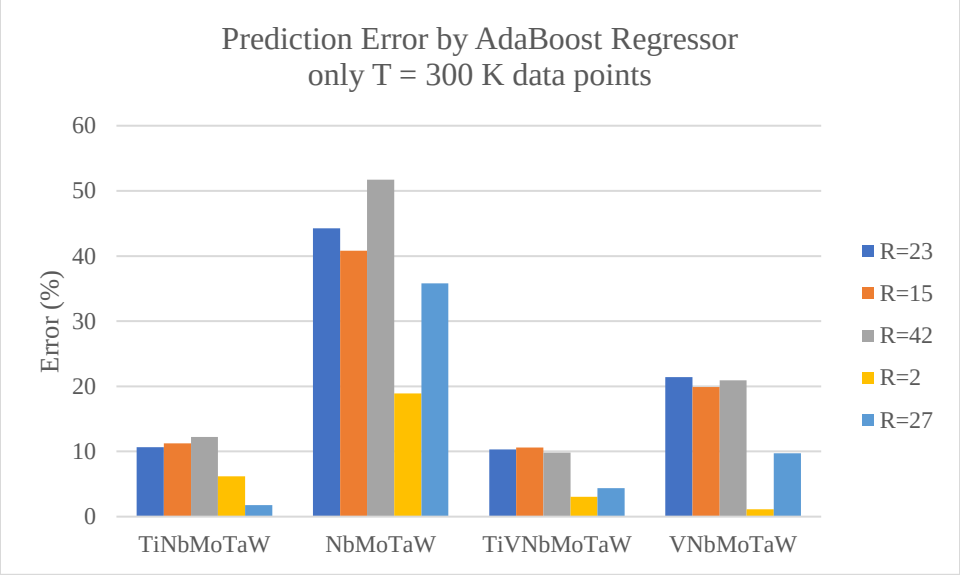

Figure 5 Prediction Error using AdaBoost Regressor for Four HEA Alloys in various R values

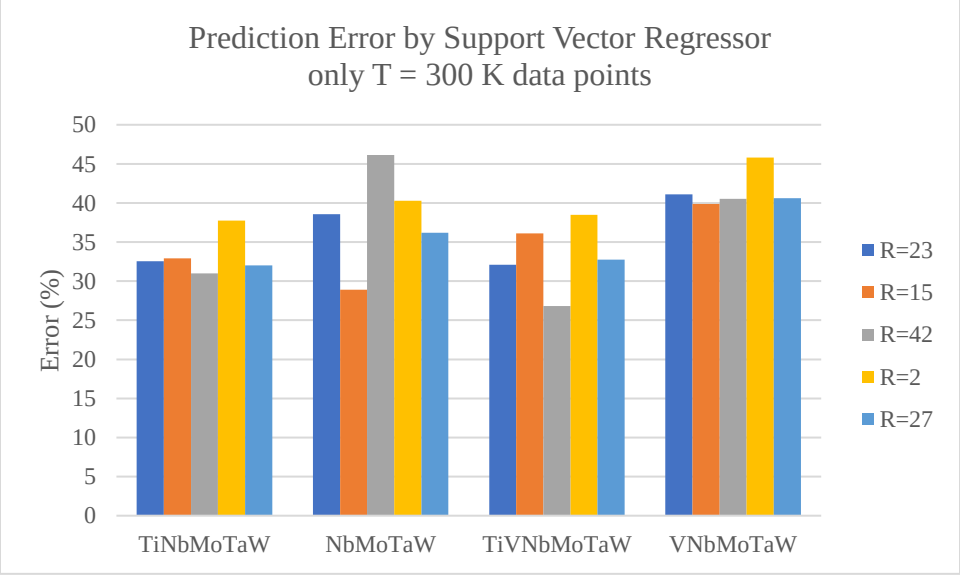

Figure 6 Prediction Error using Support Vector Regressor for Four HEA Alloys in various R values

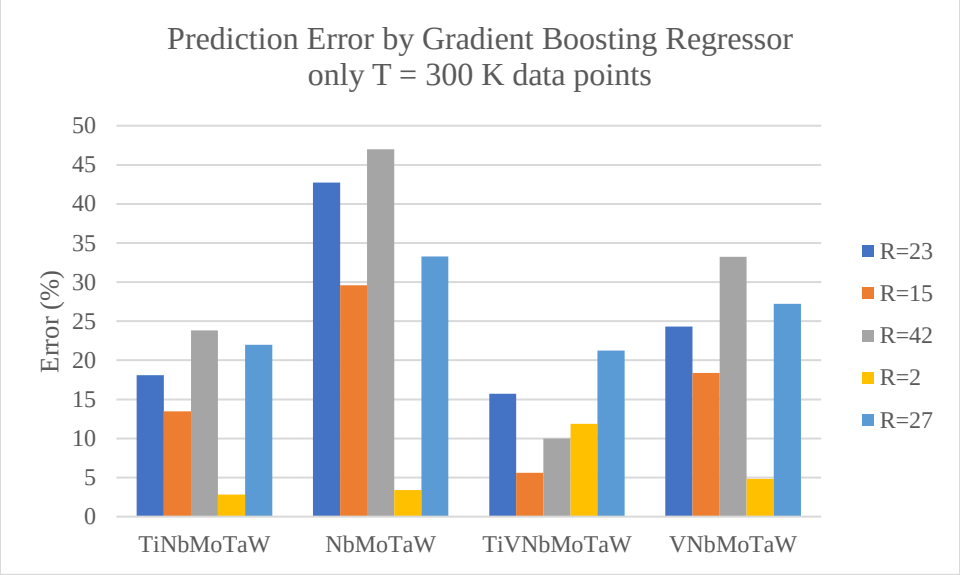

Figure 7 Prediction Error using Gradient Boosting Regressor for Four HEA Alloys in various R values

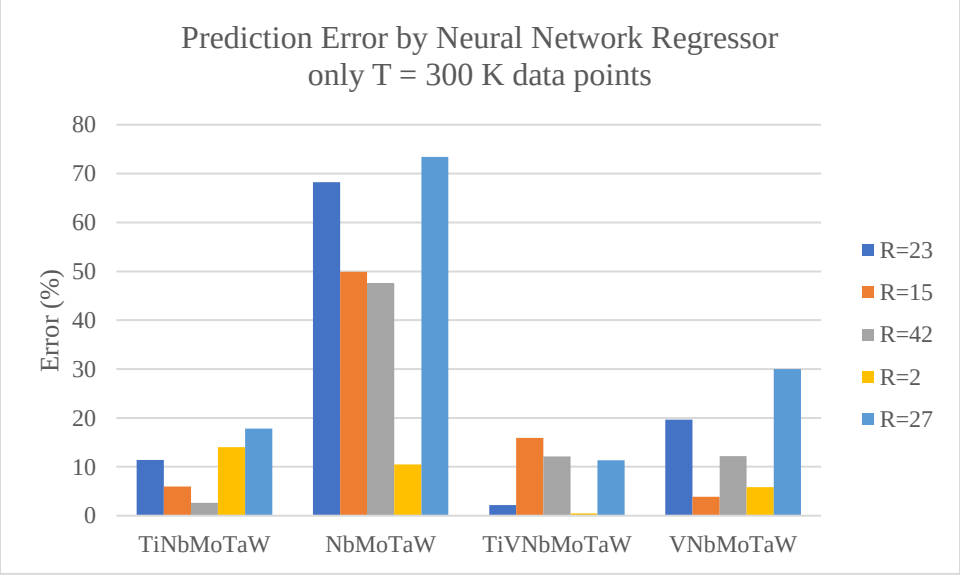

Figure 17 Prediction Error using Neural Network Regressor for Four HEA Alloys in various R values

3.3. Prediction Error of AlNbTiV & NbMoTaW with Different Feature Weights

To investigate how the dataset was constructed, the distribution of the data points at 300 K was shown in Figure 18. As shown in Figure 18, frequencies of Nb and Ti were significantly higher than other metallic elements while Co, Fe, Ni, Cu, and W elements had relatively small frequencies in the dataset. As discussed above, having fewer data in specific metallic elements affected the overall model accuracy performance and prediction results. Instead of collecting more data, to cope with this skewed dataset, the method of adjusting the weights of features was used. Before training the machine learning model with the dataset, weights were applied to the features that did not have enough data points (i.e. multiplied weights to the features). This method was applied to the two HEAs — AlNbTiV and NbMoTaW — to see whether there was an improvement or not. Figures 19 – 20 showed the prediction error per weight for the five different machine learning models. In Figure 19, the overall trend of the error for all five models was the same, and the error got decreased as adding more weights to the features, then after a certain point, the error did not get improved anymore but converged. In Figure 20, the attenuation trend was the same for the three models: Random Forest, Support Vector, and Gradient Boosting. Within these three models, the error for the Support Vector model did not converge overweight but increased after a certain weight. This showed that the model did not always get improved but blew up with excessive weight to the features. On the other hand, the other two models — AdaBoost and Neural Network — showed a different trend, which the error did not decrease over the weight but kept the same value.

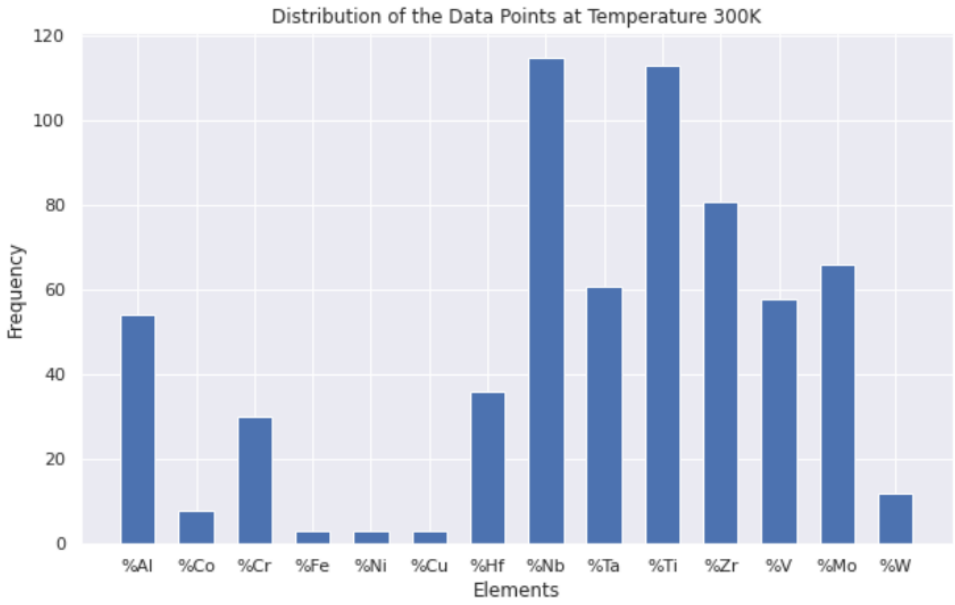

Figure 18 Data Distribution of the dataset by the Metallic Elements

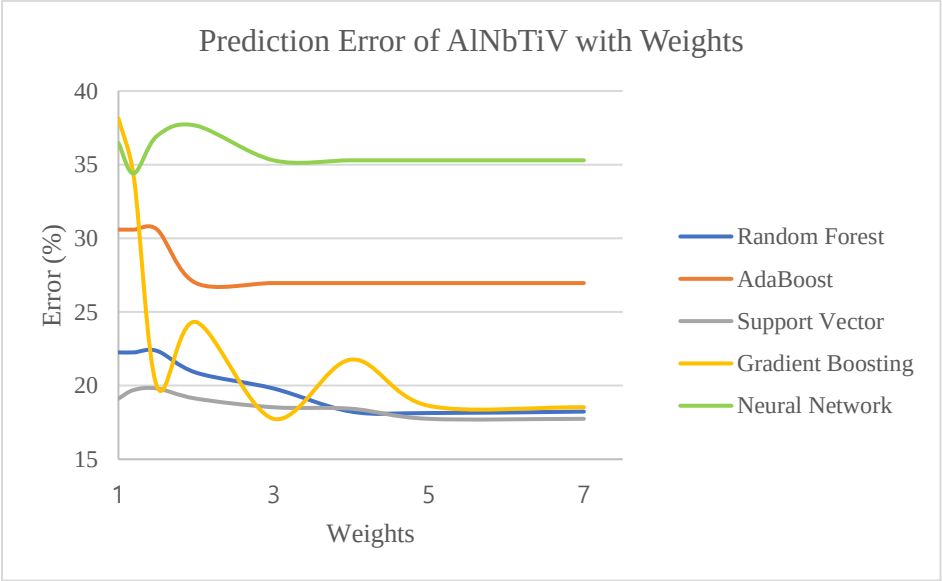

Figure 19 Prediction Error of AlNbTiV per Weights for Five Machine Learning Models

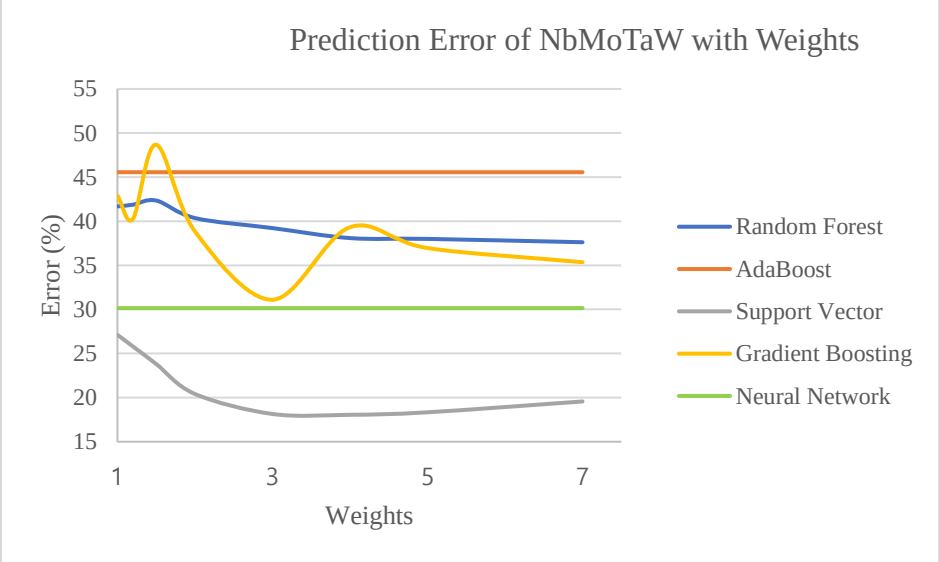

Figure 20 Prediction Error of NbMoTaW per Weights for Five Machine Learning Models

As a beginning part of the second part of the project, the feature importance of all the features was presented in Figure 21. Feature importance in the x-axis represented how much each feature (i.e. y-axis) contributed to the model performance. As clearly shown, the melting temperature and entropy were the features that had the most contribution. Since we were interested in finding combinations of metallic elements to obtain the max yield strength of the HEAs, only the metallic elements in the y-axis were considered such as Al, Nb, Ta, etc. With these four to five features, metallic elements, that had the most contribution to the model, we could run optimization algorithms in Python to find out what their combinations were to reach the maximum yield strength. In an ideal case, choosing Al, Nb, Ta, and Zr was appropriate to run the optimization algorithm. But, there were limitations that we also had to consider how and with what method we would compare the results. In this project, the method chosen to use was LAMMPS. To run a uniaxial tension simulation of alloy, in LAMMPS, using the right EAM potential files was important, which contained types of metallic elements of the alloy. For example, to find out the yield strength of

AlTiV alloy, using the potential file for AlTiV was mandatory. Therefore, if we decided to use Al, Nb, Ta, and Zr as the ideal case to run the LAMMPS simulation, we must have the potential file for these metallic elements. The limitation was, unfortunately, that there was not any potential file for AlNbTaZr, and were not many kinds of potential files available for other HEAs online. The potential files found were AlNbTi, FeCrW, FeCuNi, and FeNiCr; these four potential files were only possible to use among all the potential files available online to compare the simulation results with the optimization results from the dataset. Therefore, Although the metallic elements from those potential files did not have high feature importance, these elements were listed low in Figure 21, we proceeded with the LAMMPS simulation with these files as there was no other choice.

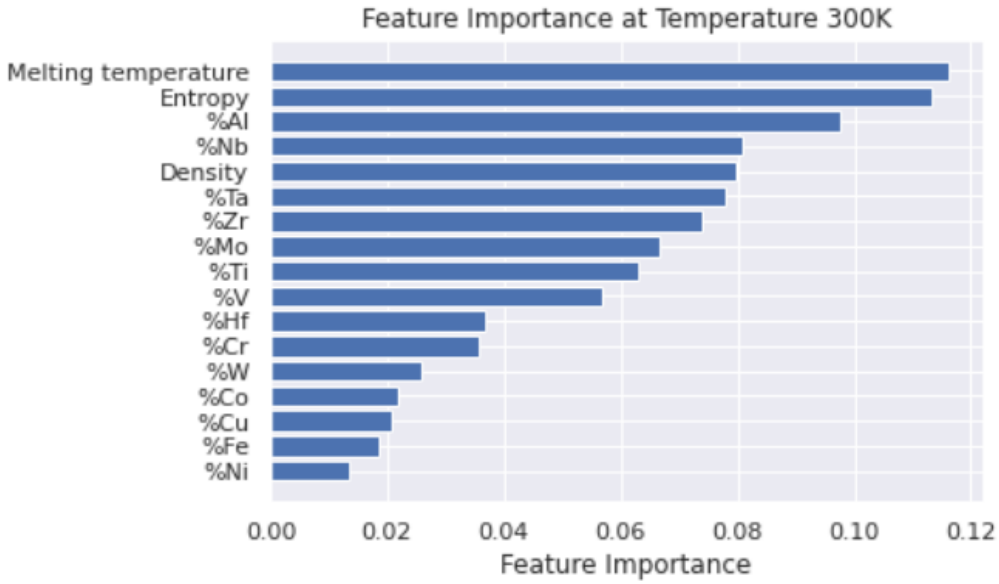

Figure 8 Importance of the Features in the Dataset at 300K Temperature

3.4. Yield Strength of the Four HEAs in LAMMPS

Figure 22 showed the rectangular-shaped wire of FeNiCr. The three different colored atoms in the wire represented Fe, Ni, and Cr. Under the uniaxial tension deformation, the stress-strain curve was obtained shown in blue points, Figure 23: this was the plot for FeNiCr. To obtain the yield strength value from the plot, finding young's modulus slope was necessary. To find the best slope line, Figure 24, we added the straight line equation and R^2 to the first several data points and iterated this process from 5 to 20 data points: the R^2 represented how well the equation fitted the data points, $R^2 = 1$ meaning the equation perfectly fitted to the points. Then, the first twelve points showed the highest R^2 value. From the twelve data points and R^2, the slope of the data points was obtained (i.e. 120928), which represented Young's modulus of the stress-strain curve. The cross point where the slope and the stress-strain curve met was the yield strength value, and it was 2300 MPa for FeNiCr followed by 1830 MPa for AlNbTi, 1611 MPa for FeCrW, and 1800 MPa for FeCuNi.

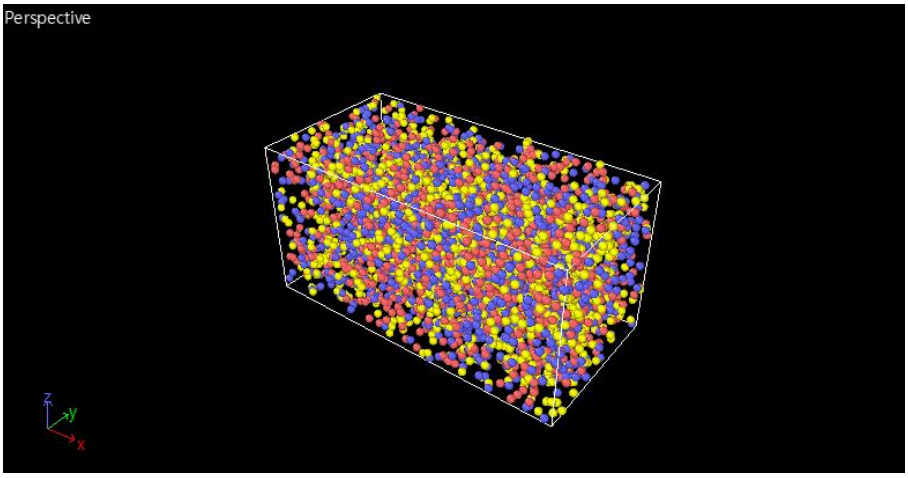

Figure 9 Rectangular FeNiCr Alloy Box for LAMMPS simulation

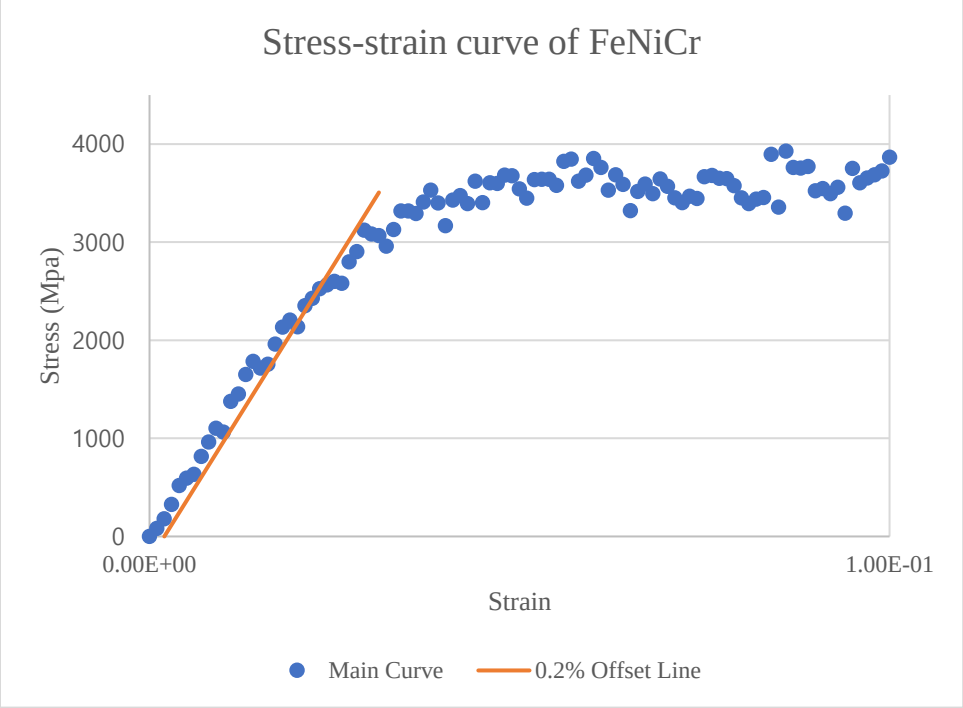

Figure 10 Stress-strain Curve of FeNiCr from the LAMMPS simulation

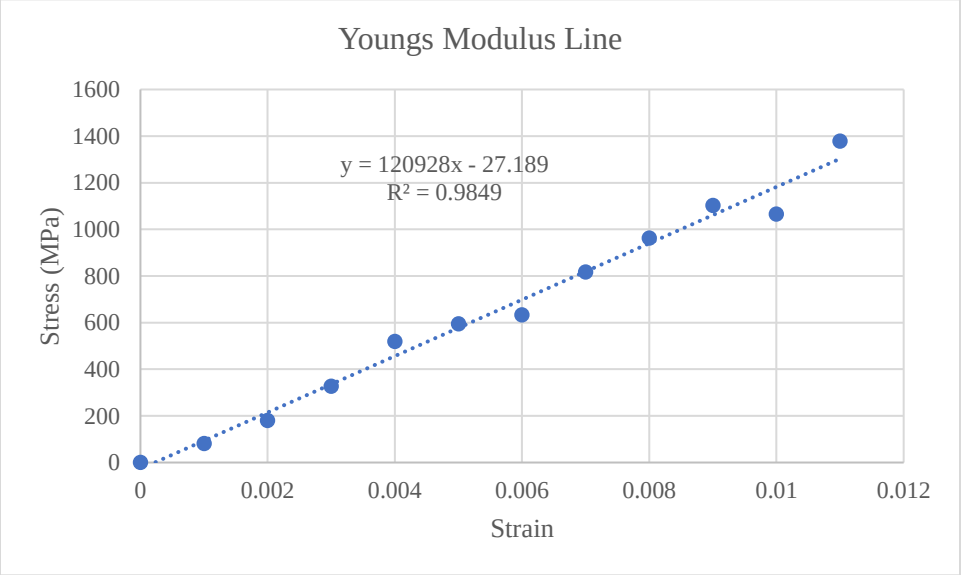

Figure 11 Youngs Modulus slope of FeNiCr alloy from LAMMPS simulation

3.5. Comparison between Machine Learning Optimized Results with LAMMPS Results

Tables 5 and 6 showed the yield strength values with weights of elements for AlNbTi, FeCrW, FeCuNi, and FeNiCr respectively as a result of the two different optimization methods used to produce the maximum yield strength values. As shown in the tables, the Test Temperature was also one of the features to run optimization with along with the metallic elements, and most of the test temperatures resulted in the range of 300 – 310 K; the lowest test temperature in the dataset was 300 K. This result was intuitively correct as the lower test temperature got, the higher yield strength value became. Another important observation was that the weights of all the metallic elements within the alloy did not add up to 100% but a little less than 100%. For instance, the total sum of Al, Nb, and Ti for AlNbTi in the Random Forest model with the Differential evolution optimization method was 60.34% as Al, Nb, and Ti were 9.57%, 19.96%, and 30.81% respectively. This trend was observed in the other HEAs while there were several cases that the sum of weights of the alloy elements was almost 100%. This situation occurred due to the nature of the raw dataset. The optimization method took the weights from the features whose values were similar to those in the raw dataset. Although the total sum of the metallic elements within the alloy from the optimization method was not 100%, which might not directly intuitive, we could at least consider this ratio of the elements as a good reference to build high-yield strength HEAs. In the case of AlNbTi in the Random Forest model with the Differential evolution optimization method, for example, the sum of the elements was 60.34%. If we scaled it to 100%, therefore, the weight of Al became from 9.57% to 15.86%, the weight of Nb became from 19.96% to 33.09%, and the weight of Ti became from 30.81% to 51.06%, which the sum of the newly scaled weights became 100%; this ratio then could be a reference to build the high yield strength HEA at the beginning of the

design process.

Table 5 Yield Strength and Optimal Molar Fractions of HEAs using Differential Evolution Method

Random Forest	Differential Evolution							
	AlNbTi		FeCrW		FeCuNi		FeNiCr	
Element	Weight (%)	Element	Weight (%)	Element	Weight (%)	Element	Weight (%)	
Al	9.57	Fe	0.73	Fe	8.14	Fe	4.31	
Nb	19.96	Cr	24.73	Cu	8.05	Ni	7.6	
Ti	30.81	W	22.8	Ni	8.12	Cr	24.75	
Test Temperature (K)	306	Test Temperature (K)	308	Test Temperature (K)	309	Test Temperature (K)	306	
Yield Strength (Mpa)	2035	Yield Strength (Mpa)	2058	Yield Strength (Mpa)	1544	Yield Strength (Mpa)	1955	

AdaBoost	Differential Evolution							
	AlNbTi		FeCrW		FeCuNi		FeNiCr	
Element	Weight (%)	Element	Weight (%)	Element	Weight (%)	Element	Weight (%)	
Al	17.6	Fe	3.9	Fe	7.64	Fe	7.46	
Nb	24.86	Cr	31.83	Cu	13.83	Ni	4.1	
Ti	31.75	W	26.92	Ni	8.19	Cr	28.99	
Test Temperature (K)	318	Test Temperature (K)	318	Test Temperature (K)	305	Test Temperature (K)	311	
Yield Strength (Mpa)	1562	Yield Strength (Mpa)	1593	Yield Strength (Mpa)	1393	Yield Strength (Mpa)	1548	

SVR	Differential Evolution							
	AlNbTi		FeCrW		FeCuNi		FeNiCr	
Element	Weight (%)	Element	Weight (%)	Element	Weight (%)	Element	Weight (%)	
Al	0	Fe	0	Fe	0	Fe	0	
Nb	32.2	Cr	53.7	Cu	0	Ni	0	
Ti	0	W	29.23	Ni	0	Cr	53.7	
Test Temperature (K)	300	Test Temperature (K)	300	Test Temperature (K)	300	Test Temperature (K)	300	
Yield Strength (Mpa)	1415	Yield Strength (Mpa)	1559	Yield Strength (Mpa)	1320	Yield Strength (Mpa)	1333	

Gradient Boosting	Differential Evolution							
	AlNbTi		FeCrW		FeCuNi		FeNiCr	
Element	Weight (%)	Element	Weight (%)	Element	Weight (%)	Element	Weight (%)	
Al	8.93	Fe	0.81	Fe	0.39	Fe	0.98	
Nb	20.03	Cr	29.48	Cu	6.1	Ni	5.65	
Ti	28.35	W	23.12	Ni	6.52	Cr	26.73	
Test Temperature (K)	304	Test Temperature (K)	314	Test Temperature (K)	305	Test Temperature (K)	306	
Yield Strength (Mpa)	2289	Yield Strength (Mpa)	2030	Yield Strength (Mpa)	1359	Yield Strength (Mpa)	1614	

Neural Network	Differential Evolution							
	AlNbTi		FeCrW		FeCuNi		FeNiCr	
Element	Weight (%)	Element	Weight (%)	Element	Weight (%)	Element	Weight (%)	
Al	39	Fe	18.18	Fe	0	Fe	18.18	
Nb	32.2	Cr	53.7	Cu	0	Ni	18.18	
Ti	40.6	W	29.23	Ni	18.18	Cr	53.7	
Test Temperature (K)	448	Test Temperature (K)	300	Test Temperature (K)	1875	Test Temperature (K)	300	
Yield Strength (Mpa)	2121	Yield Strength (Mpa)	2666	Yield Strength (Mpa)	1176	Yield Strength (Mpa)	3851	

Table 6 Yield Strength and Optimal Molar Fractions of HEAs using Maxlipo Optimizer Method

Random Forest	Maxlipo Optimizer						
AlNbTi		FeCrW		FeCuNi		FeNiCr	
Element	Weight (%)	Element	Weight (%)	Element	Weight (%)	Element	Weight (%)
Al	8.17	Fe	3.24	Fe	7.22	Fe	7.83
Nb	26.24	Cr	31.04	Cu	6.19	Ni	9.29
Ti	21.46	W	22.57	Ni	12.09	Cr	25.15
Test Temperature (K)	308	Test Temperature (K)	308	Test Temperature (K)	315	Test Temperature (K)	300
Yield Strength (Mpa)	1934	Yield Strength (Mpa)	1930	Yield Strength (Mpa)	1469	Yield Strength (Mpa)	1876

AdaBoost	Maxlipo Optimizer						
AlNbTi		FeCrW		FeCuNi		FeNiCr	
Element	Weight (%)	Element	Weight (%)	Element	Weight (%)	Element	Weight (%)
Al	17.83	Fe	5.25	Fe	7.82	Fe	7.96
Nb	24.58	Cr	25.8	Cu	17.08	Ni	0
Ti	30.47	W	29.22	Ni	6.54	Cr	53.7
Test Temperature (K)	300	Test Temperature (K)	300	Test Temperature (K)	305	Test Temperature (K)	303
Yield Strength (Mpa)	1562	Yield Strength (Mpa)	1593	Yield Strength (Mpa)	1393	Yield Strength (Mpa)	1548

SVR	Maxlipo Optimizer						
AlNbTi		FeCrW		FeCuNi		FeNiCr	
Element	Weight (%)	Element	Weight (%)	Element	Weight (%)	Element	Weight (%)
Al	0	Fe	0	Fe	0	Fe	0
Nb	32.2	Cr	53.7	Cu	0	Ni	0
Ti	0	W	29.23	Ni	0	Cr	53.7
Test Temperature (K)	300	Test Temperature (K)	300	Test Temperature (K)	300	Test Temperature (K)	300
Yield Strength (Mpa)	1415	Yield Strength (Mpa)	1559	Yield Strength (Mpa)	1320	Yield Strength (Mpa)	1333

Gradient Boosting	Maxlipo Optimizer						
AlNbTi		FeCrW		FeCuNi		FeNiCr	
Element	Weight (%)	Element	Weight (%)	Element	Weight (%)	Element	Weight (%)
Al	0	Fe	6.76	Fe	6.89	Fe	0.01
Nb	11.95	Cr	28.77	Cu	5.72	Ni	3.17
Ti	3.27	W	13.89	Ni	0	Cr	2.4
Test Temperature (K)	301	Test Temperature (K)	301	Test Temperature (K)	300	Test Temperature (K)	307
Yield Strength (Mpa)	2072	Yield Strength (Mpa)	1873	Yield Strength (Mpa)	1359	Yield Strength (Mpa)	1594

Neural Network	Maxlipo Optimizer						
AlNbTi		FeCrW		FeCuNi		FeNiCr	
Element	Weight (%)	Element	Weight (%)	Element	Weight (%)	Element	Weight (%)
Al	39	Fe	18.18	Fe	0	Fe	18.18
Nb	32.2	Cr	53.7	Cu	0	Ni	18.18
Ti	40.6	W	29.23	Ni	18.18	Cr	53.7
Test Temperature (K)	448	Test Temperature (K)	300	Test Temperature (K)	1875	Test Temperature (K)	300
Yield Strength (Mpa)	2121	Yield Strength (Mpa)	2666	Yield Strength (Mpa)	1176	Yield Strength (Mpa)	3851

Figures 25 – 26 showed the bar plots to visualize and compare the yield strength results from the two optimization methods with the results from the LAMMPS simulation for the five machine learning models. When the Differential evolution method was used, Figure 25, the error for

AlNbTi was relatively consistent in the five models: the range of the error was from 10% to 25%. A similar trend was observed in FeCuNi alloy but with a larger error that was from 15% to 30%. The best accuracy results were found in FeCrW alloy with AdaBoost and Support Vector models, 1% and 3% respectively, while the error in other models was substantially large – 27% in Random Forest, 26% in Gradient Boosting, and 65 in Neural Network models. As a last, the worst accuracy occurred in FeNiCr alloy. In terms of the result comparison between the two optimization methods, the error trend turned out to be about the same as both algorithms originated from the genetic algorithm [2]. The Maxlipo optimizer, when used with the Random Forest model, exhibited better performance compared to the Differential Evolution method, as evidenced by its lower error in AlNbTi (6%) and FeCrW (7%), but higher error in FeCuNi (4%) and FeNiCr (3%). As in Figure 13, a similar trend was observed in Figures 25 – 26, in which more data points in the dataset resulted in lower error: the error was lowest for AlNbTi compared to the other three alloys.

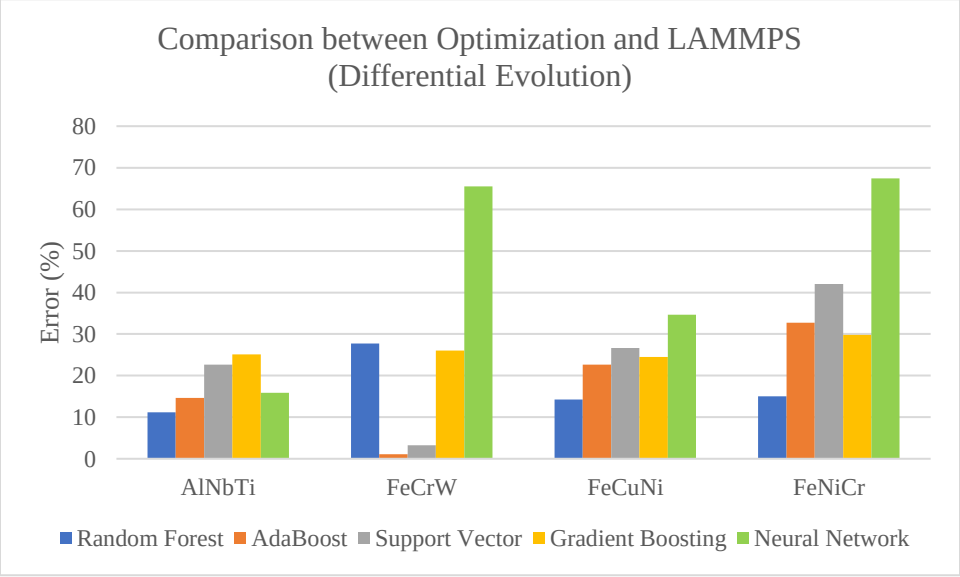

Figure 12 Error Comparison between Differential Evolution Optimization and LAMMPS

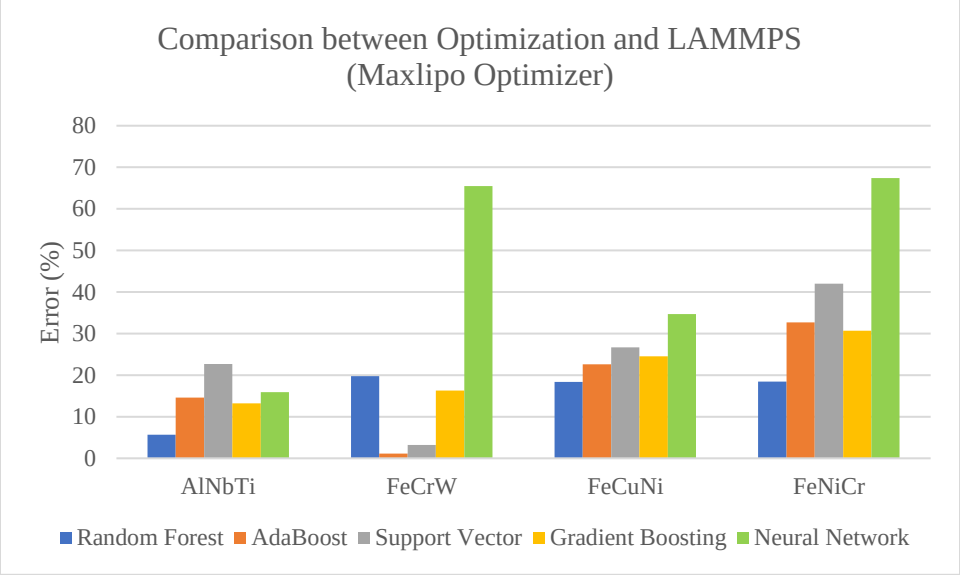

Figure 13 Error Comparison between Maxlipo Optimizer and LAMMPS

CHAPTER 4

CONCLUSIONS

This thesis project examined two topics related to the design of HEAs, which were sequentially related. The first topic investigated the use of machine learning models to predict the yield strength of HEAs at different temperatures by cleaning the material property dataset. Five machine learning models were applied, and the Random Forest regressor model with RandomSearch hyperparameter tuning showed the most accurate results at a testing temperature of 300 K.

The second topic examined how to identify the optimal combination of metallic elements to achieve the highest yield strength within HEAs. In an ideal situation, the optimization algorithm was supposed to use the features that had high feature importance from the first topic. However, due to the limitations of the EAM potential files used in LAMMPS from online resources, the optimization algorithms actually used features with low feature importance, which resulted in low accuracy of yield strength values from the optimization process. After evaluating five different machine learning models, two hyperparameter tuning methods, and two optimization algorithms, the Random Forest regressor model with RandomSearch hyperparameter tuning and Maxlipo Optimizer demonstrated the highest level of accuracy.

The second topic had several implications for future research and practice. First of all, future research could be involving more data collection to aim for better machine learning model performance. Secondly, the weight ratios of the metallic elements from the optimization process did not add up to 100 percent, but an additional process of scaling them up to 100 percent was needed in this project. Future research could investigate this issue such that the optimization algorithm could automatically scale up the weights of the metallic elements so that the manual

additional process did not need. Finally, most importantly, future research could involve finding more EAM potential files that contained metallic elements that had high feature importance so that the final findings could be more useful in the experimental HEA designing process.

Overall, this thesis project provided insights into the development of designing new HEAs using machine learning and optimization algorithms techniques, even though they might have potential inaccuracies.

Transparency document. Supporting information

Transparency data associated with this paper can be found in the online version at https://doi.org/10.1016/j.dib.2018.10.071 and https://doi.org/10.1016/j.dib.2018.11.111

REFERENCES

[1] Bhandari, U.; Zhang, C. Zeng, C.; Guo, S.; Adhikari, A. Yang, S. Deep Learning-Based Hardness Prediction of Novel Refractory High-Entropy Alloys with Experimental Validation. *Crystals* 2021, *11*, 46. https://doi.org/10.3390/cryst11010046

[2] Cannarile, F. (2022, September 1). *Metaheuristic optimization with the differential evolution algorithm*. Medium. Retrieved October 9, 2022, from https://medium.com/eni-digitalks/metaheuristic-optimization-with-the-differential-evolution-algorithm-5301480eca58

[3] Chambers, J. M., W. S. Cleveland, B. Kleiner and P. A. Tukey, Graphical Methods for Data Analysis, 395 pp., Wadsworth & Brooks/Cole Publishing Co., 1983.

[4] Chaya. (2022, April 14). *Random Forest regression*. Medium. Retrieved March 13, 2023, from https://levelup.gitconnected.com/random-forest-regression-209c0f354c84

[5] Choi, S., Yi, S., Kim, J., Shin, B., & Hyun, S. (2021, September 29). *High-entropy alloys properties prediction model by using artificial neural network algorithm*. MDPI. Retrieved August 22, 2022, from https://doi.org/10.3390/met11101559

[6] Couzinié, J.-P., Senkov, O. N., Miracle, D. B., & Dirras, G. (2018). Comprehensive data compilation on the mechanical properties of refractory high-entropy alloys. *Data in Brief*, *21*, 1622–1641. https://doi.org/10.1016/j.dib.2018.10.071

[7] D. Farkas, and C. Jones (1996), "Interatomic potentials for ternary Nb - Ti – Al alloys", *Modelling and Simulation in Materials Science and Engineering*, **4(1)**, 23-32. DOI: 10.1088/0965-0393/4/1/004

[8] Fernández-Cabán, P. L., Masters, F. J., & Phillips, B. M. (2018). Predicting roof pressures on a low-rise structure from freestream turbulence using artificial neural networks. *Frontiers in Built Environment*, *4*. https://doi.org/10.3389/fbuil.2018.00068

[9] G. Bonny, N. Castin, J. Bullens, A. Bakaev, T.C.P. Klaver, and D. Terentyev (2013), "On the mobility of vacancy clusters in reduced activation steels: an atomistic study in the Fe-Cr-W model alloy", *Journal of Physics: Condensed Matter*, **25(31)**, 315401. DOI: 10.1088/0953-8984/25/31/315401

[10] G. Bonny, R.C. Pasianot, N. Castin, and L. Malerba (2009), "Ternary Fe-Cu-Ni many-body potential to model reactor pressure vessel steels: First validation by simulated thermal annealing", *Philosophical Magazine*, **89(34-36)**, 3531-3546. DOI: 10.1080/14786430903299824

[11] Gorsse, S., Nguyen, M. H., Senkov, O. N., & Miracle, D. B. (2018). Database on the mechanical properties of high entropy alloys and complex concentrated alloys. *Data in Brief*, *21*, 2664–2678. https://doi.org/10.1016/j.dib.2018.11.111

[12] Goyal, C. (2023, February 15). *Outlier detection and removal: How to detect and remove outliers*. Analytics Vidhya. Retrieved February 20, 2023, from https://www.analyticsvidhya.com/blog/2021/05/feature-engineering-how-to-detect-and-remove-outliers-with-python-code/

[13] Han, Z. D., Chen, N., Zhao, S. F., Fan, L. W., Yang, G. N., Shao, Y., & Yao, K. F. (2017). Effect of TI additions on mechanical properties of NbMoTaW and VNbMoTaW refractory high entropy alloys. *Intermetallics*, *84*, 153–157. https://doi.org/10.1016/j.intermet.2017.01.007

[14] Kumar, G. S. (n.d.). *Understanding and building neural network (NN) models*. Built In. Retrieved March 13, 2023, from https://builtin.com/machine-learning/nn-models

[15] Machaka, Ronald (2020), "Dataset for High-Entropy Alloys Phases", Mendeley Data, V1, doi: 10.17632/7fhwrgfh2s.1

[16] Nair, A. (2022, May 2). *Grid search vs Random Search vs bayesian optimization*. Medium. Retrieved February 23, 2023, from https://towardsdatascience.com/grid-search-vs-random-search-vs-bayesian-optimization-2e68f57c3c46

[17] Parashar, A. (2022, August 25). *Explaining each machine learning model in brief*. Medium. Retrieved March 13, 2023, from https://levelup.gitconnected.com/explaining-each-machine-learning-model-in-brief-92f82b41ba71

[18] Pierre, S. (n.d.). *A guide to metaheuristic optimization for machine learning models in Python*. Built In. Retrieved March 7, 2023, from https://builtin.com/data-science/metaheuristic-optimization-python

[19] Tsai, M.-H., & Yeh, J.-W. (2014). High-entropy alloys: A critical review. *Materials Research Letters*, *2*(3), 107–123. https://doi.org/10.1080/21663831.2014.912690

[20] X.W. Zhou, M.E. Foster, and R.B. Sills (2018), "An Fe-Ni-Cr embedded atom method potential for austenitic and ferritic systems", *Journal of Computational Chemistry*, **39(29)**, 2420-2431. DOI: 10.1002/jcc.25573

[21] Yeh, J.-W. (2013). Alloy design strategies and future trends in high-entropy alloys. *JOM*, *65*(12), 1759–1771. https://doi.org/10.1007/s11837-013-0761-6

[22] Yeh, J.-W., Chen, S.-K., Lin, S.-J., Gan, J.-Y., Chin, T.-S., Shun, T.-T., Tsau, C.-H., & Chang, S.-Y. (2004). Nanostructured High-entropy alloys with multiple principal elements: Novel Alloy Design Concepts and outcomes. *Advanced Engineering Materials*, *6*(5), 299–303. https://doi.org/10.1002/adem.200300567

[23] Zeyad Yousif Abdoon Al-Shibaany *et al* 2020 *IOP Conf. Ser.: Mater. Sci. Eng.* 987 012025