Case Study 1 – Super Conductors

# Balaji Avvaru, Ravi Sivaraman, Apurv Mittal

**Abstract**

The objective of this study is to predict a composition to create a new superconductor and the critical temperature the superconductor will operate. The superconductors are materials that give little or no resistance to electrical currents at a specific (critical) temperature. This study analyzes the recorded data to predict the most important parameters to create a new superconductor.

# 1. Introduction

Superconductors are materials that conduct electricity with no resistance. This means a superconductor can carry a current indefinitely without losing any energy. However, the superconductors exhibit this property only at a critical temperature. For example: Mercury if cooled below 4.1 Kelvin becomes a super conductor and offers no resistance to electrical current.

The dataset used for this study provides the list of different superconductors identified with their chemical compositions and the critical temperature. The objective of this study is to predict the critical temperature based on the features extracted for a superconductor.

The dataset provided has two files

1. *train.csv* which contains 81 features extracted from 21,263 superconductors along with the critical temperature.
2. *unique\_m.csv* contains the chemical formula broken up for all the 21,263 superconductors from the train.csv file.

The main variables captured in the file include:

* **atomic\_mass** - total proton and neutron rest masses, in Atomic Mass Units (AMU).
* **fie** - First Ionization Energy, energy required to remove a valence electron, in kilojoules per mole (kJ/mol).
* **atomic\_radius** - calculated atomic radius, in picometer (pm).
* **density** - density at standard temperature and pressure, in kilograms per meters cubed (kg/m3).
* **electron\_affinity** - energy required to add an electron to a neutral atom, in kilojoules per mole (kJ/mol).
* **fusion\_heat** - energy to change from solid to liquid without temperature change, in kilojoules per mole (kJ/mol).
* **thermal\_conductivity** - thermal conductivity coefficient k, in watts per meter-kelvin.
* **valence** - typical number of chemical bonds formed by the element, no units.
* **critical\_temp** - superconductor critical temperature, in Kelvin.

Reference: <https://www.neuraldesigner.com/learning/examples/superconductivity>

Statistics of each variable are included such as: mean, weighted mean, geometric mean, weighted geometric mean, entropy, weighted entropy, standard, weighted standard, range and weighted range.

# 2. Methods

# The data from the files were extracted and joined via index of each row since each row in one file corresponds to the same row in the other file. First step of the data analysis is to look for any missing data. The data was identified to be complete and no missing values, thus no data imputation was needed.

Two methods were primarily used for regression models:

1. Method 1: All features including highly correlated variables
2. Method 2: Reduced feature set with highly correlated features removed

# The results from both Method 1 and Method 2 are shared in “Results” section for more analysis.

## Method 2: Reduced Dataset

# Following methodology was used to remove the variables from the original dataset:

# The dataset was plotted to check the data distribution. Most of the variables shows normal distribution or close to normal distribution. Some variables were skewed as shown below. The variables which were variant of the original data like: weighted mean, geometric mean or weighted geometric mean etc. were checked for distribution and only the variable with closer to normal distribution was retained in the dataset for further analysis. Since there are several variables capturing similar information of the same variable for example below variable are variants of the same data:

1. mean\_atomic\_radius
2. wtd\_mean\_atomic\_radius
3. gmean\_atomic\_radius
4. wtd\_gmean\_atomic\_radius
5. wtd\_mean\_atomic\_radius
6. range\_atomic\_radius

In above example, ‘wtd\_mean\_atomic\_radius’, ‘range\_atomic\_radius’ were removed from the dataset as part of the Method 2 reduced dataset.

A picture containing window

Description automatically generated

As explained above, the following variables were removed from the dataset for capturing similar information:

1. range\_atomic\_mass
2. wtd\_range\_atomic\_mass
3. wtd\_mean\_fie
4. range\_fie
5. wtd\_std\_fie
6. wtd\_mean\_atomic\_radius
7. range\_atomic\_radius
8. wtd\_mean\_Density
9. gmean\_Density
10. range\_Density
11. wtd\_std\_Density
12. range\_ElectronAffinity
13. range\_FusionHeat
14. wtd\_gmean\_ThermalConductivity
15. range\_ThermalConductivity

# Several variables in the *train.csv* file have high collinearity and to build a good independent model it's important to reduce the impact of multi-collinearity. The variables with correlation above 95% were removed from the dataset. Total of 12 variables were identified to have greater than 95% correlation with other variables as listed below:

1. 'wtd\_gmean\_atomic\_mass'
2. 'gmean\_fie'
3. 'entropy\_fie'
4. 'entropy\_atomic\_radius'
5. 'wtd\_entropy\_atomic\_radius'
6. 'wtd\_gmean\_FusionHeat'
7. 'wtd\_std\_ThermalConductivity'
8. 'gmean\_Valence'
9. 'wtd\_gmean\_Valence'
10. 'entropy\_Valence'
11. 'wtd\_entropy\_Valence'
12. 'std\_Valence'

In “*Unique\_m.csv*” data set removed the variables without variance. There were 9 variables with only “0” values, those variables were dropped. Here is the list of variables dropped:

1. 'He'
2. 'Ne'
3. 'Ar'
4. 'Kr'
5. 'Xe'
6. 'Pm'
7. 'Po'
8. 'At'
9. 'Rn'

Also, the variable "material" which comprises of all elements used as materials is dropped from the data frame. This information is captured in the other variables which covers the combination of variable used in each allow in *“Unique\_m.csv”.*

## Regression Models:

After the identification of the variables following regression models were executed on the final dataset using cross validation (at k folds = 10):

1. Linear Regression
2. Lasso (at various alpha levels)
3. Ridge Regression (at various alpha levels)
4. ElasticNet

Root Mean Square Error (RMSE) was used to identify the most successful models.

# 3. Results

Method 1: Models with original data set

|  |  |  |
| --- | --- | --- |
| Model | RMSE | Standard Deviation of RMSE at CV=10 |
| Multi-Linear Regression | -22.57 | 12.42 |
| Lasso (at alpha = 0.3) | -17.96 | 0.28 |
| Ridge (at alpha =1000) | -17.89 | 1.04 |
| ElasticNet | -19.46 | 0.27 |

Method 2: Models with reduced data set as explained in the “Methods” section.

|  |  |  |
| --- | --- | --- |
| Model | RMSE | Standard Deviation of RMSE at CV=10 |
| Multi-Linear Regression | -22.68 | 10.94 |
| Lasso (at alpha = 0.3) | -18.35 | 0.33 |
| Ridge (at alpha =1000) | -18.21 | 1.09 |
| ElasticNet | -19.67 | 0.29 |

Ridge Model showed the lowest RMSE for Method 1, however it has higher standard deviation compared to Lasso at alpha of 0.3. Upon comparing the score for Ridge, its apparent that the scores are not consistent.

**Ridge Scores** = -17.20146094, -17.76609165, -17.70643718, -17.55575516, -17.52395924, -18.28908214, -17.33259556, -17.65596932, -17.0831061 , -20.85931023

Lasso at a = 0.3 shows consistent scores and very low standard deviation. Even with slightly higher RMSE score for Lasso at -17.96 vs -17.89 for Ridge model. The best model with consistent scores is Lasso (a=0.3) for Method 1 (with all data variables considered in the model).

**Lasso Scores** = -17.49501274, -18.18279108, -18.04697072, -17.9540785,

-17.99042669, -18.24794041, -17.76672689, -18.01578225,

-17.48547188, -18.39358418

The recommended model is **Lasso (a=0.3)** with all data variables in its original form and will be used for further analysis and prediction.

Lasso – Important Features (Top 10 Features)

Based on the Lasso model, the top 10 features to be considered for the regression model are listed below:

Chart, bar chart

Description automatically generated

Most important features for the regression model:

1. Bi
2. wtd\_mean\_ThermalConductivity
3. gmean\_Density
4. wtd\_gmean\_ThermalConductivity
5. wtd\_entropy\_ThermalConductivity
6. wtd\_std\_Valence
7. range\_atomic\_mass
8. Ba
9. wtd\_std\_atomic\_mass
10. wtd\_entropy\_ElectronAffinity
11. wtd\_entropy\_atomic\_mass
12. Si
13. wtd\_gmean\_ElectronAffinity
14. Ca
15. wtd\_std\_ThermalConductivity
16. mean\_fie
17. range\_atomic\_radius
18. wtd\_std\_FusionHeat
19. As

Lasso Residual Plot

The residual plot of Lasso appears to be well within bounds and together with very few outliers. However, there is slight pattern in the distribution of the residuals. Since majority of the residual values are co-located this provides confidence that the model is successful in predicting the critical temperature with the selected parameters.

Scatter chart

Description automatically generated

Similarly, Error distribution also shows centered at 0 with uniform distribution.

Histogram

Description automatically generated with low confidence

# 4. Conclusion

Based on the Lasso model, the multiple linear regression model can be interpreted as below:

wtd\_mean\_ThermalConductivity

….

wtd\_mean\_ThermalConductivity

…..

Interpretation:

1. All features being zero, the critical temperature of the allow will be 34.42 K.
2. Considering all features at constant, with per unit addition of the Bi (Bismuth) the critical temperature increases by 3.85 K.
3. Considering all features at constant, with per unit increase in wtd\_gmean\_ThermalConductivity the critical temperature decreases by 8.91 K.
4. Similarly, all identified features impact the critical temperature of the superconductor.

# Appendix – Code

Nbviewer link: [Jupyter Notebook Viewer (nbviewer.org)](https://nbviewer.org/github/balajiavvaru/MSDS7333-/blob/main/CaseStudy1_Balaji_Ravi_Apurv_Final%20CODE.ipynb)

# Case Study 1 : Super Conductor

Submitted by:

* Ravi Sivaraman
* Balaji Avvaru
* Apurv Mittal

Superconductors are materials that conduct electricity with no resistance. This means a superconductor can carry a current indefinitely without losing any energy. However, the superconductors exhibit this property only at a critical temperature. For example: Mercury if cooled below 4.1 Kelvin becomes a super conductor and offers no resistance to electrical current.

The dataset used for this study provides the list of different superconductors identified with their chemical compositions and the critical temperature. The objective of this study is to predict the critical temperature based on the features extracted for a superconductor.

The dataset provided has two files

train.csv which contains 81 features extracted from 21,263 superconductors along with the critical temperature.

unique\_m.csv contains the chemical formula broken up for all the 21,263 superconductors from the train.csv file.

The code for importing the data is combined with the inital loading of various analysis and visualization packages below.

In [1]:

**import** pandas **as** pd

**import** numpy **as** np

**import** seaborn **as** sns

**import** matplotlib.pyplot **as** plt

**from** scipy **import** stats

**from** statsmodels.stats.outliers\_influence **import** variance\_inflation\_factor

**%matplotlib** inline

**import** warnings

warnings**.**filterwarnings('ignore')

**from** sklearn.feature\_selection **import** VarianceThreshold

**from** sklearn **import** linear\_model

**from** sklearn.model\_selection **import** cross\_val\_score

**from** sklearn.model\_selection **import** cross\_val\_predict

**from** sklearn.model\_selection **import** KFold

**from** sklearn.model\_selection **import** GridSearchCV

**from** sklearn.linear\_model **import** Lasso

**from** sklearn.linear\_model **import** Ridge

**from** sklearn.linear\_model **import** ElasticNet

**from** sklearn.feature\_selection **import** RFE

**from** sklearn.preprocessing **import** StandardScaler

#### Read and alalyze train csv file

In [2]:

*# read train csv files*

train\_df **=** pd**.**read\_csv('train.csv')

*# critical\_temp is part of both csv files drop critical\_temp from train dataframe*

train\_df**.**drop(['critical\_temp'], inplace**=True**, axis**=**1)

In [3]:

train\_orin\_df **=** pd**.**read\_csv('train.csv')

train\_orin\_df**.**drop(['critical\_temp'], inplace**=True**, axis**=**1)

In [4]:

train\_df**.**describe()

Out[4]:

|  | **number\_of\_elements** | **mean\_atomic\_mass** | **wtd\_mean\_atomic\_mass** | **gmean\_atomic\_mass** | **wtd\_gmean\_atomic\_mass** | **entropy\_atomic\_mass** | **wtd\_entropy\_atomic\_mass** | **range\_atomic\_mass** | **wtd\_range\_atomic\_mass** | **std\_atomic\_mass** | **...** | **mean\_Valence** | **wtd\_mean\_Valence** | **gmean\_Valence** | **wtd\_gmean\_Valence** | **entropy\_Valence** | **wtd\_entropy\_Valence** | **range\_Valence** | **wtd\_range\_Valence** | **std\_Valence** | **wtd\_std\_Valence** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 21263.000000 | 21263.000000 | 21263.000000 | 21263.000000 | 21263.000000 | 21263.000000 | 21263.000000 | 21263.000000 | 21263.000000 | 21263.000000 | ... | 21263.000000 | 21263.000000 | 21263.000000 | 21263.000000 | 21263.000000 | 21263.000000 | 21263.000000 | 21263.000000 | 21263.000000 | 21263.000000 |
| **mean** | 4.115224 | 87.557631 | 72.988310 | 71.290627 | 58.539916 | 1.165608 | 1.063884 | 115.601251 | 33.225218 | 44.391893 | ... | 3.198228 | 3.153127 | 3.056536 | 3.055885 | 1.295682 | 1.052841 | 2.041010 | 1.483007 | 0.839342 | 0.673987 |
| **std** | 1.439295 | 29.676497 | 33.490406 | 31.030272 | 36.651067 | 0.364930 | 0.401423 | 54.626887 | 26.967752 | 20.035430 | ... | 1.044611 | 1.191249 | 1.046257 | 1.174815 | 0.393155 | 0.380291 | 1.242345 | 0.978176 | 0.484676 | 0.455580 |
| **min** | 1.000000 | 6.941000 | 6.423452 | 5.320573 | 1.960849 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **25%** | 3.000000 | 72.458076 | 52.143839 | 58.041225 | 35.248990 | 0.966676 | 0.775363 | 78.512902 | 16.824174 | 32.890369 | ... | 2.333333 | 2.116732 | 2.279705 | 2.091251 | 1.060857 | 0.775678 | 1.000000 | 0.921454 | 0.451754 | 0.306892 |
| **50%** | 4.000000 | 84.922750 | 60.696571 | 66.361592 | 39.918385 | 1.199541 | 1.146783 | 122.906070 | 26.636008 | 45.123500 | ... | 2.833333 | 2.618182 | 2.615321 | 2.434057 | 1.368922 | 1.166532 | 2.000000 | 1.063077 | 0.800000 | 0.500000 |
| **75%** | 5.000000 | 100.404410 | 86.103540 | 78.116681 | 73.113234 | 1.444537 | 1.359418 | 154.119320 | 38.356908 | 59.322812 | ... | 4.000000 | 4.026201 | 3.727919 | 3.914868 | 1.589027 | 1.330801 | 3.000000 | 1.918400 | 1.200000 | 1.020436 |
| **max** | 9.000000 | 208.980400 | 208.980400 | 208.980400 | 208.980400 | 1.983797 | 1.958203 | 207.972460 | 205.589910 | 101.019700 | ... | 7.000000 | 7.000000 | 7.000000 | 7.000000 | 2.141963 | 1.949739 | 6.000000 | 6.992200 | 3.000000 | 3.000000 |

8 rows × 81 columns

In [5]:

*# Validate null values in train csv file*

train\_df**.**isnull()**.**sum()**.**sum()

Out[5]:

0

In [6]:

*#Visualizing the hist of data to check normality of independent variable*

train\_df**.**hist(bins**=**50,figsize**=**(25,30))

plt**.**show()

A picture containing window

Description automatically generated

In [7]:

*# variables which are not normally distributed and captured the duplicate information*

non\_normal\_var **=** ['range\_atomic\_mass', 'wtd\_range\_atomic\_mass', 'wtd\_mean\_fie', 'range\_fie', 'wtd\_std\_fie', 'wtd\_mean\_atomic\_radius',

'range\_atomic\_radius', 'wtd\_mean\_Density', 'gmean\_Density', 'range\_Density', 'wtd\_std\_Density', 'range\_ElectronAffinity',

'range\_FusionHeat', 'wtd\_gmean\_ThermalConductivity', 'range\_ThermalConductivity']

*# Removing not normal variables from dataframe*

train\_df**.**drop(non\_normal\_var, inplace**=True**, axis**=**1)

In [8]:

*#https://www.projectpro.io/recipes/drop-out-highly-correlated-features-in-python*

*# to drop features with colliniarity more than 95%*

pd**.**set\_option('display.max\_rows', 100)

corr\_df **=** pd**.**DataFrame(train\_df**.**corr()**.**abs())

corr\_df**.**head(100)

Out[8]:

|  | **number\_of\_elements** | **mean\_atomic\_mass** | **wtd\_mean\_atomic\_mass** | **gmean\_atomic\_mass** | **wtd\_gmean\_atomic\_mass** | **entropy\_atomic\_mass** | **wtd\_entropy\_atomic\_mass** | **std\_atomic\_mass** | **wtd\_std\_atomic\_mass** | **mean\_fie** | **...** | **mean\_Valence** | **wtd\_mean\_Valence** | **gmean\_Valence** | **wtd\_gmean\_Valence** | **entropy\_Valence** | **wtd\_entropy\_Valence** | **range\_Valence** | **wtd\_range\_Valence** | **std\_Valence** | **wtd\_std\_Valence** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **number\_of\_elements** | 1.000000 | 0.141923 | 0.353064 | 0.292969 | 0.454525 | 0.939304 | 0.881845 | 0.513998 | 0.546391 | 0.167451 | ... | 0.609412 | 0.648551 | 0.618512 | 0.659268 | 0.967832 | 0.892559 | 0.231874 | 0.447770 | 0.105365 | 0.035216 |
| **mean\_atomic\_mass** | 0.141923 | 1.000000 | 0.815977 | 0.940298 | 0.745841 | 0.104000 | 0.097609 | 0.196460 | 0.130675 | 0.285782 | ... | 0.374099 | 0.304683 | 0.392153 | 0.321399 | 0.156786 | 0.145610 | 0.107450 | 0.168633 | 0.080279 | 0.081253 |
| **wtd\_mean\_atomic\_mass** | 0.353064 | 0.815977 | 1.000000 | 0.848242 | 0.964085 | 0.308046 | 0.412666 | 0.060739 | 0.089471 | 0.209296 | ... | 0.534450 | 0.545587 | 0.539780 | 0.548981 | 0.375718 | 0.331025 | 0.039155 | 0.330904 | 0.003681 | 0.077323 |
| **gmean\_atomic\_mass** | 0.292969 | 0.940298 | 0.848242 | 1.000000 | 0.856975 | 0.190214 | 0.232183 | 0.121708 | 0.166042 | 0.367690 | ... | 0.487021 | 0.427961 | 0.511508 | 0.450357 | 0.306246 | 0.307662 | 0.165010 | 0.272303 | 0.124627 | 0.117336 |
| **wtd\_gmean\_atomic\_mass** | 0.454525 | 0.745841 | 0.964085 | 0.856975 | 1.000000 | 0.370561 | 0.484664 | 0.274487 | 0.331657 | 0.276668 | ... | 0.599413 | 0.614100 | 0.608417 | 0.623261 | 0.477785 | 0.448072 | 0.078641 | 0.409674 | 0.033313 | 0.030361 |
| **entropy\_atomic\_mass** | 0.939304 | 0.104000 | 0.308046 | 0.190214 | 0.370561 | 1.000000 | 0.889709 | 0.357951 | 0.413647 | 0.058804 | ... | 0.551641 | 0.586855 | 0.554477 | 0.592817 | 0.963621 | 0.861479 | 0.203207 | 0.350047 | 0.087838 | 0.018758 |
| **wtd\_entropy\_atomic\_mass** | 0.881845 | 0.097609 | 0.412666 | 0.232183 | 0.484664 | 0.889709 | 1.000000 | 0.474542 | 0.495970 | 0.126955 | ... | 0.583784 | 0.644336 | 0.586226 | 0.650247 | 0.897636 | 0.918284 | 0.161350 | 0.546910 | 0.056309 | 0.006937 |
| **std\_atomic\_mass** | 0.513998 | 0.196460 | 0.060739 | 0.121708 | 0.274487 | 0.357951 | 0.474542 | 1.000000 | 0.919788 | 0.196162 | ... | 0.313584 | 0.337815 | 0.330630 | 0.352901 | 0.518439 | 0.540601 | 0.192347 | 0.273598 | 0.144106 | 0.122547 |
| **wtd\_std\_atomic\_mass** | 0.546391 | 0.130675 | 0.089471 | 0.166042 | 0.331657 | 0.413647 | 0.495970 | 0.919788 | 1.000000 | 0.253661 | ... | 0.372045 | 0.394987 | 0.386043 | 0.414098 | 0.559955 | 0.606574 | 0.177675 | 0.391024 | 0.123636 | 0.155702 |
| **mean\_fie** | 0.167451 | 0.285782 | 0.209296 | 0.367690 | 0.276668 | 0.058804 | 0.126955 | 0.196162 | 0.253661 | 1.000000 | ... | 0.109749 | 0.098992 | 0.142754 | 0.122387 | 0.156312 | 0.260380 | 0.205324 | 0.235069 | 0.161033 | 0.170006 |
| **gmean\_fie** | 0.024229 | 0.240565 | 0.109490 | 0.286844 | 0.154323 | 0.075007 | 0.018499 | 0.095263 | 0.138114 | 0.969325 | ... | 0.076551 | 0.075439 | 0.046003 | 0.054233 | 0.009336 | 0.125304 | 0.191984 | 0.131647 | 0.166488 | 0.194419 |
| **wtd\_gmean\_fie** | 0.424152 | 0.219381 | 0.508109 | 0.341585 | 0.588014 | 0.358642 | 0.475331 | 0.322310 | 0.406154 | 0.517754 | ... | 0.458337 | 0.479924 | 0.465677 | 0.480221 | 0.431439 | 0.382479 | 0.036008 | 0.340174 | 0.000583 | 0.105682 |
| **entropy\_fie** | 0.973195 | 0.166935 | 0.369773 | 0.316670 | 0.471280 | 0.964695 | 0.891982 | 0.522052 | 0.557283 | 0.163814 | ... | 0.596644 | 0.624606 | 0.608303 | 0.637874 | 0.992726 | 0.907923 | 0.260018 | 0.389817 | 0.139747 | 0.077940 |
| **wtd\_entropy\_fie** | 0.719209 | 0.163565 | 0.129779 | 0.287701 | 0.227652 | 0.691974 | 0.698922 | 0.439915 | 0.479914 | 0.316151 | ... | 0.340607 | 0.340946 | 0.369296 | 0.375613 | 0.730266 | 0.862061 | 0.361073 | 0.454246 | 0.249305 | 0.332966 |
| **wtd\_range\_fie** | 0.329624 | 0.080545 | 0.420457 | 0.155439 | 0.451326 | 0.358509 | 0.324119 | 0.215224 | 0.226462 | 0.008830 | ... | 0.442376 | 0.460419 | 0.429217 | 0.438015 | 0.379288 | 0.137482 | 0.104309 | 0.061797 | 0.107349 | 0.307760 |
| **std\_fie** | 0.674005 | 0.276561 | 0.459323 | 0.450045 | 0.578719 | 0.600950 | 0.661304 | 0.497456 | 0.576330 | 0.505927 | ... | 0.721737 | 0.696705 | 0.740587 | 0.713240 | 0.689995 | 0.689920 | 0.151535 | 0.500329 | 0.063530 | 0.000063 |
| **mean\_atomic\_radius** | 0.001389 | 0.497664 | 0.288451 | 0.510867 | 0.301508 | 0.095771 | 0.085179 | 0.000581 | 0.046852 | 0.690319 | ... | 0.052578 | 0.003385 | 0.086126 | 0.032473 | 0.009756 | 0.065894 | 0.238393 | 0.109244 | 0.220036 | 0.291992 |
| **gmean\_atomic\_radius** | 0.240444 | 0.561061 | 0.468457 | 0.647560 | 0.527074 | 0.122754 | 0.173624 | 0.193301 | 0.264117 | 0.770120 | ... | 0.342367 | 0.295607 | 0.372333 | 0.322096 | 0.243188 | 0.310648 | 0.213301 | 0.304293 | 0.170579 | 0.200041 |
| **wtd\_gmean\_atomic\_radius** | 0.518256 | 0.359894 | 0.667112 | 0.496461 | 0.749593 | 0.447304 | 0.564591 | 0.359721 | 0.453859 | 0.400405 | ... | 0.632103 | 0.651223 | 0.636740 | 0.654187 | 0.535898 | 0.485474 | 0.042693 | 0.456791 | 0.006524 | 0.096146 |
| **entropy\_atomic\_radius** | 0.972245 | 0.140034 | 0.345071 | 0.282048 | 0.441916 | 0.972329 | 0.890615 | 0.508515 | 0.541000 | 0.120416 | ... | 0.579872 | 0.610735 | 0.590868 | 0.623531 | 0.989546 | 0.898930 | 0.259130 | 0.375326 | 0.139544 | 0.077988 |
| **wtd\_entropy\_atomic\_radius** | 0.904121 | 0.147604 | 0.400483 | 0.311701 | 0.514618 | 0.880213 | 0.961464 | 0.551753 | 0.619910 | 0.187760 | ... | 0.623502 | 0.660216 | 0.635244 | 0.677313 | 0.919184 | 0.951463 | 0.226873 | 0.576320 | 0.114904 | 0.093575 |
| **wtd\_range\_atomic\_radius** | 0.371350 | 0.141100 | 0.363882 | 0.240296 | 0.432896 | 0.288179 | 0.572284 | 0.280473 | 0.392248 | 0.228869 | ... | 0.413485 | 0.439858 | 0.423345 | 0.455683 | 0.332892 | 0.512876 | 0.069213 | 0.722955 | 0.018608 | 0.061663 |
| **std\_atomic\_radius** | 0.624810 | 0.326403 | 0.551141 | 0.512841 | 0.665166 | 0.566698 | 0.660625 | 0.505261 | 0.566690 | 0.325563 | ... | 0.713658 | 0.688617 | 0.722552 | 0.696988 | 0.670556 | 0.660613 | 0.084398 | 0.469035 | 0.019095 | 0.056573 |
| **wtd\_std\_atomic\_radius** | 0.695089 | 0.280440 | 0.554820 | 0.462397 | 0.681130 | 0.647365 | 0.737633 | 0.510406 | 0.611431 | 0.312821 | ... | 0.713816 | 0.733805 | 0.711929 | 0.736105 | 0.738201 | 0.730072 | 0.039561 | 0.543134 | 0.031263 | 0.109204 |
| **mean\_Density** | 0.418675 | 0.756861 | 0.749261 | 0.779757 | 0.740131 | 0.371944 | 0.396968 | 0.054127 | 0.126586 | 0.194406 | ... | 0.709641 | 0.626874 | 0.721242 | 0.643125 | 0.425309 | 0.396193 | 0.083691 | 0.395385 | 0.024797 | 0.016377 |
| **wtd\_gmean\_Density** | 0.649882 | 0.525588 | 0.767011 | 0.663642 | 0.843708 | 0.574118 | 0.655734 | 0.384091 | 0.473078 | 0.309339 | ... | 0.787529 | 0.804085 | 0.795337 | 0.814657 | 0.657725 | 0.638629 | 0.087742 | 0.571594 | 0.014400 | 0.056154 |
| **entropy\_Density** | 0.871832 | 0.043416 | 0.246377 | 0.125672 | 0.300078 | 0.932668 | 0.813476 | 0.341714 | 0.363766 | 0.098050 | ... | 0.438001 | 0.479754 | 0.430860 | 0.479727 | 0.900579 | 0.779776 | 0.156710 | 0.261252 | 0.052318 | 0.001252 |
| **wtd\_entropy\_Density** | 0.767078 | 0.026325 | 0.195894 | 0.093881 | 0.273122 | 0.785862 | 0.850983 | 0.450138 | 0.466107 | 0.059349 | ... | 0.319051 | 0.389292 | 0.326816 | 0.402037 | 0.780306 | 0.837233 | 0.250488 | 0.425261 | 0.146199 | 0.152149 |
| **wtd\_range\_Density** | 0.355389 | 0.342391 | 0.585687 | 0.368143 | 0.576836 | 0.313413 | 0.494731 | 0.072304 | 0.161116 | 0.143612 | ... | 0.452547 | 0.503307 | 0.451274 | 0.507844 | 0.328885 | 0.413359 | 0.008209 | 0.634471 | 0.055460 | 0.048841 |
| **std\_Density** | 0.210724 | 0.245042 | 0.103157 | 0.037866 | 0.048110 | 0.132053 | 0.201740 | 0.654985 | 0.603394 | 0.233809 | ... | 0.047196 | 0.012949 | 0.016267 | 0.028931 | 0.223375 | 0.275794 | 0.259181 | 0.052541 | 0.257379 | 0.198795 |
| **mean\_ElectronAffinity** | 0.119303 | 0.088230 | 0.147303 | 0.079376 | 0.119314 | 0.113803 | 0.094036 | 0.043337 | 0.085604 | 0.466972 | ... | 0.321216 | 0.330667 | 0.289160 | 0.312905 | 0.110131 | 0.033251 | 0.193715 | 0.104693 | 0.200917 | 0.218167 |
| **wtd\_mean\_ElectronAffinity** | 0.195608 | 0.061103 | 0.096427 | 0.006353 | 0.158608 | 0.197941 | 0.239710 | 0.216503 | 0.286153 | 0.296057 | ... | 0.063204 | 0.029280 | 0.074342 | 0.031283 | 0.214876 | 0.176687 | 0.052746 | 0.029781 | 0.051092 | 0.010135 |
| **gmean\_ElectronAffinity** | 0.356067 | 0.189282 | 0.272261 | 0.219651 | 0.274209 | 0.331789 | 0.320524 | 0.110884 | 0.090454 | 0.225023 | ... | 0.500995 | 0.504130 | 0.480164 | 0.495592 | 0.346279 | 0.279780 | 0.082682 | 0.247611 | 0.116236 | 0.143877 |
| **wtd\_gmean\_ElectronAffinity** | 0.052884 | 0.134382 | 0.021877 | 0.111858 | 0.011941 | 0.025710 | 0.022881 | 0.053965 | 0.119017 | 0.153236 | ... | 0.111913 | 0.154391 | 0.110341 | 0.161498 | 0.022912 | 0.094571 | 0.050487 | 0.143140 | 0.027394 | 0.079913 |
| **entropy\_ElectronAffinity** | 0.877304 | 0.091539 | 0.290220 | 0.238002 | 0.395866 | 0.884913 | 0.796011 | 0.484241 | 0.533046 | 0.098082 | ... | 0.491108 | 0.523597 | 0.505040 | 0.537412 | 0.904659 | 0.823543 | 0.265024 | 0.324851 | 0.150735 | 0.094690 |
| **wtd\_entropy\_ElectronAffinity** | 0.625798 | 0.107651 | 0.093796 | 0.224757 | 0.194127 | 0.617247 | 0.591390 | 0.395189 | 0.458664 | 0.298388 | ... | 0.247190 | 0.279732 | 0.273162 | 0.310245 | 0.653655 | 0.759590 | 0.334608 | 0.393388 | 0.226893 | 0.303421 |
| **wtd\_range\_ElectronAffinity** | 0.241411 | 0.010235 | 0.204480 | 0.055316 | 0.246388 | 0.280781 | 0.267555 | 0.219772 | 0.241183 | 0.098522 | ... | 0.229217 | 0.169959 | 0.237823 | 0.164888 | 0.286828 | 0.133700 | 0.041614 | 0.132737 | 0.054814 | 0.080619 |
| **std\_ElectronAffinity** | 0.423738 | 0.164960 | 0.197729 | 0.249926 | 0.262822 | 0.402694 | 0.427352 | 0.316892 | 0.355561 | 0.553870 | ... | 0.244811 | 0.209334 | 0.280599 | 0.234673 | 0.441397 | 0.472159 | 0.310349 | 0.189659 | 0.253786 | 0.230442 |
| **wtd\_std\_ElectronAffinity** | 0.480813 | 0.133101 | 0.210757 | 0.238028 | 0.291153 | 0.458669 | 0.534124 | 0.366293 | 0.418981 | 0.453955 | ... | 0.259158 | 0.250939 | 0.293796 | 0.280100 | 0.494662 | 0.579242 | 0.310218 | 0.312077 | 0.258989 | 0.264354 |
| **mean\_FusionHeat** | 0.437624 | 0.137669 | 0.006730 | 0.092244 | 0.054752 | 0.484572 | 0.454171 | 0.203046 | 0.260348 | 0.001013 | ... | 0.488123 | 0.478450 | 0.494814 | 0.488039 | 0.434774 | 0.395375 | 0.035540 | 0.286468 | 0.008708 | 0.045541 |
| **wtd\_mean\_FusionHeat** | 0.449272 | 0.135429 | 0.014681 | 0.089139 | 0.072658 | 0.491876 | 0.464303 | 0.201149 | 0.283987 | 0.037151 | ... | 0.439682 | 0.499794 | 0.436508 | 0.503301 | 0.451595 | 0.449115 | 0.005202 | 0.365964 | 0.053801 | 0.092672 |
| **gmean\_FusionHeat** | 0.514252 | 0.014818 | 0.164239 | 0.086599 | 0.219751 | 0.531184 | 0.521330 | 0.261873 | 0.314796 | 0.135472 | ... | 0.604493 | 0.572989 | 0.622305 | 0.587671 | 0.514469 | 0.473427 | 0.110862 | 0.333871 | 0.060116 | 0.008076 |
| **wtd\_gmean\_FusionHeat** | 0.519109 | 0.043003 | 0.120044 | 0.024199 | 0.189353 | 0.544208 | 0.534181 | 0.251984 | 0.342816 | 0.130902 | ... | 0.518345 | 0.571290 | 0.519993 | 0.578353 | 0.526863 | 0.529789 | 0.034940 | 0.425550 | 0.022502 | 0.068536 |
| **entropy\_FusionHeat** | 0.900759 | 0.008499 | 0.225287 | 0.126798 | 0.313735 | 0.928251 | 0.835019 | 0.449821 | 0.487457 | 0.067640 | ... | 0.500919 | 0.559244 | 0.490564 | 0.560693 | 0.921445 | 0.824121 | 0.120887 | 0.342548 | 0.013797 | 0.016918 |
| **wtd\_entropy\_FusionHeat** | 0.860479 | 0.028541 | 0.237218 | 0.171928 | 0.349844 | 0.845089 | 0.873931 | 0.506630 | 0.575421 | 0.091153 | ... | 0.504138 | 0.589524 | 0.503019 | 0.598601 | 0.866068 | 0.908728 | 0.170917 | 0.547006 | 0.063734 | 0.056473 |
| **wtd\_range\_FusionHeat** | 0.371788 | 0.167528 | 0.070411 | 0.128758 | 0.006009 | 0.386431 | 0.392008 | 0.166545 | 0.270280 | 0.053585 | ... | 0.288012 | 0.384025 | 0.276334 | 0.385503 | 0.355642 | 0.440176 | 0.044222 | 0.480671 | 0.100086 | 0.076896 |
| **std\_FusionHeat** | 0.113361 | 0.337969 | 0.253911 | 0.361244 | 0.239074 | 0.178542 | 0.140514 | 0.000353 | 0.056823 | 0.175516 | ... | 0.096527 | 0.135557 | 0.077105 | 0.127758 | 0.090183 | 0.074018 | 0.162875 | 0.108712 | 0.173417 | 0.157327 |
| **wtd\_std\_FusionHeat** | 0.074796 | 0.335778 | 0.272806 | 0.372666 | 0.277760 | 0.133078 | 0.074441 | 0.040275 | 0.016325 | 0.190600 | ... | 0.084557 | 0.116743 | 0.069348 | 0.106550 | 0.036513 | 0.007079 | 0.150326 | 0.027364 | 0.151733 | 0.182440 |
| **mean\_ThermalConductivity** | 0.227656 | 0.158266 | 0.236418 | 0.190937 | 0.248849 | 0.216404 | 0.220697 | 0.136312 | 0.138042 | 0.067805 | ... | 0.355912 | 0.373348 | 0.330129 | 0.354064 | 0.240317 | 0.195642 | 0.195944 | 0.177460 | 0.192302 | 0.261168 |
| **wtd\_mean\_ThermalConductivity** | 0.206069 | 0.065989 | 0.058075 | 0.104940 | 0.056793 | 0.172703 | 0.184949 | 0.130974 | 0.058402 | 0.097611 | ... | 0.253874 | 0.276539 | 0.231307 | 0.260902 | 0.196164 | 0.218736 | 0.160914 | 0.189599 | 0.156601 | 0.206584 |
| **gmean\_ThermalConductivity** | 0.485324 | 0.006004 | 0.184990 | 0.110769 | 0.271083 | 0.472574 | 0.507619 | 0.309363 | 0.394396 | 0.285230 | ... | 0.406019 | 0.418818 | 0.411016 | 0.426053 | 0.502534 | 0.511838 | 0.077929 | 0.353427 | 0.025488 | 0.005427 |
| **entropy\_ThermalConductivity** | 0.501871 | 0.100077 | 0.076936 | 0.116455 | 0.076157 | 0.508056 | 0.381716 | 0.155560 | 0.091194 | 0.147664 | ... | 0.083684 | 0.065917 | 0.109405 | 0.086614 | 0.484029 | 0.389657 | 0.337554 | 0.042422 | 0.245604 | 0.255835 |
| **wtd\_entropy\_ThermalConductivity** | 0.207065 | 0.098221 | 0.025638 | 0.104644 | 0.020495 | 0.212688 | 0.182082 | 0.103910 | 0.066305 | 0.036971 | ... | 0.110553 | 0.141321 | 0.068933 | 0.105337 | 0.199033 | 0.254061 | 0.421079 | 0.038302 | 0.347741 | 0.447827 |
| **wtd\_range\_ThermalConductivity** | 0.316772 | 0.027790 | 0.108512 | 0.095661 | 0.129212 | 0.308654 | 0.302974 | 0.209846 | 0.155144 | 0.045948 | ... | 0.388703 | 0.413807 | 0.364107 | 0.394153 | 0.335672 | 0.302291 | 0.178406 | 0.189897 | 0.171831 | 0.274266 |
| **std\_ThermalConductivity** | 0.602018 | 0.110658 | 0.362512 | 0.233587 | 0.447236 | 0.597723 | 0.618855 | 0.386600 | 0.454069 | 0.072623 | ... | 0.671569 | 0.714405 | 0.640881 | 0.695269 | 0.641505 | 0.589997 | 0.174000 | 0.443612 | 0.209550 | 0.306211 |
| **wtd\_std\_ThermalConductivity** | 0.665580 | 0.110856 | 0.350993 | 0.232079 | 0.431027 | 0.650085 | 0.684497 | 0.383254 | 0.443604 | 0.063098 | ... | 0.678028 | 0.735073 | 0.643793 | 0.715135 | 0.691772 | 0.663885 | 0.187465 | 0.509148 | 0.232050 | 0.323640 |
| **mean\_Valence** | 0.609412 | 0.374099 | 0.534450 | 0.487021 | 0.599413 | 0.551641 | 0.583784 | 0.313584 | 0.372045 | 0.109749 | ... | 1.000000 | 0.937103 | 0.989911 | 0.940001 | 0.619428 | 0.558969 | 0.085892 | 0.554338 | 0.164058 | 0.203341 |
| **wtd\_mean\_Valence** | 0.648551 | 0.304683 | 0.545587 | 0.427961 | 0.614100 | 0.586855 | 0.644336 | 0.337815 | 0.394987 | 0.098992 | ... | 0.937103 | 1.000000 | 0.917905 | 0.994939 | 0.652105 | 0.636286 | 0.111249 | 0.683700 | 0.186270 | 0.258593 |
| **gmean\_Valence** | 0.618512 | 0.392153 | 0.539780 | 0.511508 | 0.608417 | 0.554477 | 0.586226 | 0.330630 | 0.386043 | 0.142754 | ... | 0.989911 | 0.917905 | 1.000000 | 0.933036 | 0.615971 | 0.563527 | 0.041651 | 0.532805 | 0.034589 | 0.088388 |
| **wtd\_gmean\_Valence** | 0.659268 | 0.321399 | 0.548981 | 0.450357 | 0.623261 | 0.592817 | 0.650247 | 0.352901 | 0.414098 | 0.122387 | ... | 0.940001 | 0.994939 | 0.933036 | 1.000000 | 0.656235 | 0.648708 | 0.033007 | 0.681415 | 0.108469 | 0.168962 |
| **entropy\_Valence** | 0.967832 | 0.156786 | 0.375718 | 0.306246 | 0.477785 | 0.963621 | 0.897636 | 0.518439 | 0.559955 | 0.156312 | ... | 0.619428 | 0.652105 | 0.615971 | 0.656235 | 1.000000 | 0.910822 | 0.155410 | 0.417283 | 0.034580 | 0.017031 |
| **wtd\_entropy\_Valence** | 0.892559 | 0.145610 | 0.331025 | 0.307662 | 0.448072 | 0.861479 | 0.918284 | 0.540601 | 0.606574 | 0.260380 | ... | 0.558969 | 0.636286 | 0.563527 | 0.648708 | 0.910822 | 1.000000 | 0.204411 | 0.639818 | 0.089774 | 0.081284 |
| **range\_Valence** | 0.231874 | 0.107450 | 0.039155 | 0.165010 | 0.078641 | 0.203207 | 0.161350 | 0.192347 | 0.177675 | 0.205324 | ... | 0.085892 | 0.111249 | 0.041651 | 0.033007 | 0.155410 | 0.204411 | 1.000000 | 0.114000 | 0.973788 | 0.867393 |
| **wtd\_range\_Valence** | 0.447770 | 0.168633 | 0.330904 | 0.272303 | 0.409674 | 0.350047 | 0.546910 | 0.273598 | 0.391024 | 0.235069 | ... | 0.554338 | 0.683700 | 0.532805 | 0.681415 | 0.417283 | 0.639818 | 0.114000 | 1.000000 | 0.184380 | 0.132162 |
| **std\_Valence** | 0.105365 | 0.080279 | 0.003681 | 0.124627 | 0.033313 | 0.087838 | 0.056309 | 0.144106 | 0.123636 | 0.161033 | ... | 0.164058 | 0.186270 | 0.034589 | 0.108469 | 0.034580 | 0.089774 | 0.973788 | 0.184380 | 1.000000 | 0.885856 |
| **wtd\_std\_Valence** | 0.035216 | 0.081253 | 0.077323 | 0.117336 | 0.030361 | 0.018758 | 0.006937 | 0.122547 | 0.155702 | 0.170006 | ... | 0.203341 | 0.258593 | 0.088388 | 0.168962 | 0.017031 | 0.081284 | 0.867393 | 0.132162 | 0.885856 | 1.000000 |

66 rows × 66 columns

In [9]:

*# Multi Colliniarity analysis on Independent variables*

upper\_tri **=** corr\_df**.**where(np**.**triu(np**.**ones(corr\_df**.**shape),k**=**1)**.**astype(np**.**bool))

print(upper\_tri)

number\_of\_elements mean\_atomic\_mass \

number\_of\_elements NaN 0.141923

mean\_atomic\_mass NaN NaN

wtd\_mean\_atomic\_mass NaN NaN

gmean\_atomic\_mass NaN NaN

wtd\_gmean\_atomic\_mass NaN NaN

entropy\_atomic\_mass NaN NaN

wtd\_entropy\_atomic\_mass NaN NaN

std\_atomic\_mass NaN NaN

wtd\_std\_atomic\_mass NaN NaN

mean\_fie NaN NaN

gmean\_fie NaN NaN

wtd\_gmean\_fie NaN NaN

entropy\_fie NaN NaN

wtd\_entropy\_fie NaN NaN

wtd\_range\_fie NaN NaN

std\_fie NaN NaN

mean\_atomic\_radius NaN NaN

gmean\_atomic\_radius NaN NaN

wtd\_gmean\_atomic\_radius NaN NaN

entropy\_atomic\_radius NaN NaN

wtd\_entropy\_atomic\_radius NaN NaN

wtd\_range\_atomic\_radius NaN NaN

std\_atomic\_radius NaN NaN

wtd\_std\_atomic\_radius NaN NaN

mean\_Density NaN NaN

wtd\_gmean\_Density NaN NaN

entropy\_Density NaN NaN

wtd\_entropy\_Density NaN NaN

wtd\_range\_Density NaN NaN

std\_Density NaN NaN

mean\_ElectronAffinity NaN NaN

wtd\_mean\_ElectronAffinity NaN NaN

gmean\_ElectronAffinity NaN NaN

wtd\_gmean\_ElectronAffinity NaN NaN

entropy\_ElectronAffinity NaN NaN

wtd\_entropy\_ElectronAffinity NaN NaN

wtd\_range\_ElectronAffinity NaN NaN

std\_ElectronAffinity NaN NaN

wtd\_std\_ElectronAffinity NaN NaN

mean\_FusionHeat NaN NaN

wtd\_mean\_FusionHeat NaN NaN

gmean\_FusionHeat NaN NaN

wtd\_gmean\_FusionHeat NaN NaN

entropy\_FusionHeat NaN NaN

wtd\_entropy\_FusionHeat NaN NaN

wtd\_range\_FusionHeat NaN NaN

std\_FusionHeat NaN NaN

wtd\_std\_FusionHeat NaN NaN

mean\_ThermalConductivity NaN NaN

wtd\_mean\_ThermalConductivity NaN NaN

gmean\_ThermalConductivity NaN NaN

entropy\_ThermalConductivity NaN NaN

wtd\_entropy\_ThermalConductivity NaN NaN

wtd\_range\_ThermalConductivity NaN NaN

std\_ThermalConductivity NaN NaN

wtd\_std\_ThermalConductivity NaN NaN

mean\_Valence NaN NaN

wtd\_mean\_Valence NaN NaN

gmean\_Valence NaN NaN

wtd\_gmean\_Valence NaN NaN

entropy\_Valence NaN NaN

wtd\_entropy\_Valence NaN NaN

range\_Valence NaN NaN

wtd\_range\_Valence NaN NaN

std\_Valence NaN NaN

wtd\_std\_Valence NaN NaN

wtd\_mean\_atomic\_mass gmean\_atomic\_mass \

number\_of\_elements 0.353064 0.292969

mean\_atomic\_mass 0.815977 0.940298

wtd\_mean\_atomic\_mass NaN 0.848242

gmean\_atomic\_mass NaN NaN

wtd\_gmean\_atomic\_mass NaN NaN

entropy\_atomic\_mass NaN NaN

wtd\_entropy\_atomic\_mass NaN NaN

std\_atomic\_mass NaN NaN

wtd\_std\_atomic\_mass NaN NaN

mean\_fie NaN NaN

gmean\_fie NaN NaN

wtd\_gmean\_fie NaN NaN

entropy\_fie NaN NaN

wtd\_entropy\_fie NaN NaN

wtd\_range\_fie NaN NaN

std\_fie NaN NaN

mean\_atomic\_radius NaN NaN

gmean\_atomic\_radius NaN NaN

wtd\_gmean\_atomic\_radius NaN NaN

entropy\_atomic\_radius NaN NaN

wtd\_entropy\_atomic\_radius NaN NaN

wtd\_range\_atomic\_radius NaN NaN

std\_atomic\_radius NaN NaN

wtd\_std\_atomic\_radius NaN NaN

mean\_Density NaN NaN

wtd\_gmean\_Density NaN NaN

entropy\_Density NaN NaN

wtd\_entropy\_Density NaN NaN

wtd\_range\_Density NaN NaN

std\_Density NaN NaN

mean\_ElectronAffinity NaN NaN

wtd\_mean\_ElectronAffinity NaN NaN

gmean\_ElectronAffinity NaN NaN

wtd\_gmean\_ElectronAffinity NaN NaN

entropy\_ElectronAffinity NaN NaN

wtd\_entropy\_ElectronAffinity NaN NaN

wtd\_range\_ElectronAffinity NaN NaN

std\_ElectronAffinity NaN NaN

wtd\_std\_ElectronAffinity NaN NaN

mean\_FusionHeat NaN NaN

wtd\_mean\_FusionHeat NaN NaN

gmean\_FusionHeat NaN NaN

wtd\_gmean\_FusionHeat NaN NaN

entropy\_FusionHeat NaN NaN

wtd\_entropy\_FusionHeat NaN NaN

wtd\_range\_FusionHeat NaN NaN

std\_FusionHeat NaN NaN

wtd\_std\_FusionHeat NaN NaN

mean\_ThermalConductivity NaN NaN

wtd\_mean\_ThermalConductivity NaN NaN

gmean\_ThermalConductivity NaN NaN

entropy\_ThermalConductivity NaN NaN

wtd\_entropy\_ThermalConductivity NaN NaN

wtd\_range\_ThermalConductivity NaN NaN

std\_ThermalConductivity NaN NaN

wtd\_std\_ThermalConductivity NaN NaN

mean\_Valence NaN NaN

wtd\_mean\_Valence NaN NaN

gmean\_Valence NaN NaN

wtd\_gmean\_Valence NaN NaN

entropy\_Valence NaN NaN

wtd\_entropy\_Valence NaN NaN

range\_Valence NaN NaN

wtd\_range\_Valence NaN NaN

std\_Valence NaN NaN

wtd\_std\_Valence NaN NaN

wtd\_gmean\_atomic\_mass entropy\_atomic\_mass \

number\_of\_elements 0.454525 0.939304

mean\_atomic\_mass 0.745841 0.104000

wtd\_mean\_atomic\_mass 0.964085 0.308046

gmean\_atomic\_mass 0.856975 0.190214

wtd\_gmean\_atomic\_mass NaN 0.370561

entropy\_atomic\_mass NaN NaN

wtd\_entropy\_atomic\_mass NaN NaN

std\_atomic\_mass NaN NaN

wtd\_std\_atomic\_mass NaN NaN

mean\_fie NaN NaN

gmean\_fie NaN NaN

wtd\_gmean\_fie NaN NaN

entropy\_fie NaN NaN

wtd\_entropy\_fie NaN NaN

wtd\_range\_fie NaN NaN

std\_fie NaN NaN

mean\_atomic\_radius NaN NaN

gmean\_atomic\_radius NaN NaN

wtd\_gmean\_atomic\_radius NaN NaN

entropy\_atomic\_radius NaN NaN

wtd\_entropy\_atomic\_radius NaN NaN

wtd\_range\_atomic\_radius NaN NaN

std\_atomic\_radius NaN NaN

wtd\_std\_atomic\_radius NaN NaN

mean\_Density NaN NaN

wtd\_gmean\_Density NaN NaN

entropy\_Density NaN NaN

wtd\_entropy\_Density NaN NaN

wtd\_range\_Density NaN NaN

std\_Density NaN NaN

mean\_ElectronAffinity NaN NaN

wtd\_mean\_ElectronAffinity NaN NaN

gmean\_ElectronAffinity NaN NaN

wtd\_gmean\_ElectronAffinity NaN NaN

entropy\_ElectronAffinity NaN NaN

wtd\_entropy\_ElectronAffinity NaN NaN

wtd\_range\_ElectronAffinity NaN NaN

std\_ElectronAffinity NaN NaN

wtd\_std\_ElectronAffinity NaN NaN

mean\_FusionHeat NaN NaN

wtd\_mean\_FusionHeat NaN NaN

gmean\_FusionHeat NaN NaN

wtd\_gmean\_FusionHeat NaN NaN

entropy\_FusionHeat NaN NaN

wtd\_entropy\_FusionHeat NaN NaN

wtd\_range\_FusionHeat NaN NaN

std\_FusionHeat NaN NaN

wtd\_std\_FusionHeat NaN NaN

mean\_ThermalConductivity NaN NaN

wtd\_mean\_ThermalConductivity NaN NaN

gmean\_ThermalConductivity NaN NaN

entropy\_ThermalConductivity NaN NaN

wtd\_entropy\_ThermalConductivity NaN NaN

wtd\_range\_ThermalConductivity NaN NaN

std\_ThermalConductivity NaN NaN

wtd\_std\_ThermalConductivity NaN NaN

mean\_Valence NaN NaN

wtd\_mean\_Valence NaN NaN

gmean\_Valence NaN NaN

wtd\_gmean\_Valence NaN NaN

entropy\_Valence NaN NaN

wtd\_entropy\_Valence NaN NaN

range\_Valence NaN NaN

wtd\_range\_Valence NaN NaN

std\_Valence NaN NaN

wtd\_std\_Valence NaN NaN

wtd\_entropy\_atomic\_mass std\_atomic\_mass \

number\_of\_elements 0.881845 0.513998

mean\_atomic\_mass 0.097609 0.196460

wtd\_mean\_atomic\_mass 0.412666 0.060739

gmean\_atomic\_mass 0.232183 0.121708

wtd\_gmean\_atomic\_mass 0.484664 0.274487

entropy\_atomic\_mass 0.889709 0.357951

wtd\_entropy\_atomic\_mass NaN 0.474542

std\_atomic\_mass NaN NaN

wtd\_std\_atomic\_mass NaN NaN

mean\_fie NaN NaN

gmean\_fie NaN NaN

wtd\_gmean\_fie NaN NaN

entropy\_fie NaN NaN

wtd\_entropy\_fie NaN NaN

wtd\_range\_fie NaN NaN

std\_fie NaN NaN

mean\_atomic\_radius NaN NaN

gmean\_atomic\_radius NaN NaN

wtd\_gmean\_atomic\_radius NaN NaN

entropy\_atomic\_radius NaN NaN

wtd\_entropy\_atomic\_radius NaN NaN

wtd\_range\_atomic\_radius NaN NaN

std\_atomic\_radius NaN NaN

wtd\_std\_atomic\_radius NaN NaN

mean\_Density NaN NaN

wtd\_gmean\_Density NaN NaN

entropy\_Density NaN NaN

wtd\_entropy\_Density NaN NaN

wtd\_range\_Density NaN NaN

std\_Density NaN NaN

mean\_ElectronAffinity NaN NaN

wtd\_mean\_ElectronAffinity NaN NaN

gmean\_ElectronAffinity NaN NaN

wtd\_gmean\_ElectronAffinity NaN NaN

entropy\_ElectronAffinity NaN NaN

wtd\_entropy\_ElectronAffinity NaN NaN

wtd\_range\_ElectronAffinity NaN NaN

std\_ElectronAffinity NaN NaN

wtd\_std\_ElectronAffinity NaN NaN

mean\_FusionHeat NaN NaN

wtd\_mean\_FusionHeat NaN NaN

gmean\_FusionHeat NaN NaN

wtd\_gmean\_FusionHeat NaN NaN

entropy\_FusionHeat NaN NaN

wtd\_entropy\_FusionHeat NaN NaN

wtd\_range\_FusionHeat NaN NaN

std\_FusionHeat NaN NaN

wtd\_std\_FusionHeat NaN NaN

mean\_ThermalConductivity NaN NaN

wtd\_mean\_ThermalConductivity NaN NaN

gmean\_ThermalConductivity NaN NaN

entropy\_ThermalConductivity NaN NaN

wtd\_entropy\_ThermalConductivity NaN NaN

wtd\_range\_ThermalConductivity NaN NaN

std\_ThermalConductivity NaN NaN

wtd\_std\_ThermalConductivity NaN NaN

mean\_Valence NaN NaN

wtd\_mean\_Valence NaN NaN

gmean\_Valence NaN NaN

wtd\_gmean\_Valence NaN NaN

entropy\_Valence NaN NaN

wtd\_entropy\_Valence NaN NaN

range\_Valence NaN NaN

wtd\_range\_Valence NaN NaN

std\_Valence NaN NaN

wtd\_std\_Valence NaN NaN

wtd\_std\_atomic\_mass mean\_fie ... \

number\_of\_elements 0.546391 0.167451 ...

mean\_atomic\_mass 0.130675 0.285782 ...

wtd\_mean\_atomic\_mass 0.089471 0.209296 ...

gmean\_atomic\_mass 0.166042 0.367690 ...

wtd\_gmean\_atomic\_mass 0.331657 0.276668 ...

entropy\_atomic\_mass 0.413647 0.058804 ...

wtd\_entropy\_atomic\_mass 0.495970 0.126955 ...

std\_atomic\_mass 0.919788 0.196162 ...

wtd\_std\_atomic\_mass NaN 0.253661 ...

mean\_fie NaN NaN ...

gmean\_fie NaN NaN ...

wtd\_gmean\_fie NaN NaN ...

entropy\_fie NaN NaN ...

wtd\_entropy\_fie NaN NaN ...

wtd\_range\_fie NaN NaN ...

std\_fie NaN NaN ...

mean\_atomic\_radius NaN NaN ...

gmean\_atomic\_radius NaN NaN ...

wtd\_gmean\_atomic\_radius NaN NaN ...

entropy\_atomic\_radius NaN NaN ...

wtd\_entropy\_atomic\_radius NaN NaN ...

wtd\_range\_atomic\_radius NaN NaN ...

std\_atomic\_radius NaN NaN ...

wtd\_std\_atomic\_radius NaN NaN ...

mean\_Density NaN NaN ...

wtd\_gmean\_Density NaN NaN ...

entropy\_Density NaN NaN ...

wtd\_entropy\_Density NaN NaN ...

wtd\_range\_Density NaN NaN ...

std\_Density NaN NaN ...

mean\_ElectronAffinity NaN NaN ...

wtd\_mean\_ElectronAffinity NaN NaN ...

gmean\_ElectronAffinity NaN NaN ...

wtd\_gmean\_ElectronAffinity NaN NaN ...

entropy\_ElectronAffinity NaN NaN ...

wtd\_entropy\_ElectronAffinity NaN NaN ...

wtd\_range\_ElectronAffinity NaN NaN ...

std\_ElectronAffinity NaN NaN ...

wtd\_std\_ElectronAffinity NaN NaN ...

mean\_FusionHeat NaN NaN ...

wtd\_mean\_FusionHeat NaN NaN ...

gmean\_FusionHeat NaN NaN ...

wtd\_gmean\_FusionHeat NaN NaN ...

entropy\_FusionHeat NaN NaN ...

wtd\_entropy\_FusionHeat NaN NaN ...

wtd\_range\_FusionHeat NaN NaN ...

std\_FusionHeat NaN NaN ...

wtd\_std\_FusionHeat NaN NaN ...

mean\_ThermalConductivity NaN NaN ...

wtd\_mean\_ThermalConductivity NaN NaN ...

gmean\_ThermalConductivity NaN NaN ...

entropy\_ThermalConductivity NaN NaN ...

wtd\_entropy\_ThermalConductivity NaN NaN ...

wtd\_range\_ThermalConductivity NaN NaN ...

std\_ThermalConductivity NaN NaN ...

wtd\_std\_ThermalConductivity NaN NaN ...

mean\_Valence NaN NaN ...

wtd\_mean\_Valence NaN NaN ...

gmean\_Valence NaN NaN ...

wtd\_gmean\_Valence NaN NaN ...

entropy\_Valence NaN NaN ...

wtd\_entropy\_Valence NaN NaN ...

range\_Valence NaN NaN ...

wtd\_range\_Valence NaN NaN ...

std\_Valence NaN NaN ...

wtd\_std\_Valence NaN NaN ...

mean\_Valence wtd\_mean\_Valence \

number\_of\_elements 0.609412 0.648551

mean\_atomic\_mass 0.374099 0.304683

wtd\_mean\_atomic\_mass 0.534450 0.545587

gmean\_atomic\_mass 0.487021 0.427961

wtd\_gmean\_atomic\_mass 0.599413 0.614100

entropy\_atomic\_mass 0.551641 0.586855

wtd\_entropy\_atomic\_mass 0.583784 0.644336

std\_atomic\_mass 0.313584 0.337815

wtd\_std\_atomic\_mass 0.372045 0.394987

mean\_fie 0.109749 0.098992

gmean\_fie 0.076551 0.075439

wtd\_gmean\_fie 0.458337 0.479924

entropy\_fie 0.596644 0.624606

wtd\_entropy\_fie 0.340607 0.340946

wtd\_range\_fie 0.442376 0.460419

std\_fie 0.721737 0.696705

mean\_atomic\_radius 0.052578 0.003385

gmean\_atomic\_radius 0.342367 0.295607

wtd\_gmean\_atomic\_radius 0.632103 0.651223

entropy\_atomic\_radius 0.579872 0.610735

wtd\_entropy\_atomic\_radius 0.623502 0.660216

wtd\_range\_atomic\_radius 0.413485 0.439858

std\_atomic\_radius 0.713658 0.688617

wtd\_std\_atomic\_radius 0.713816 0.733805

mean\_Density 0.709641 0.626874

wtd\_gmean\_Density 0.787529 0.804085

entropy\_Density 0.438001 0.479754

wtd\_entropy\_Density 0.319051 0.389292

wtd\_range\_Density 0.452547 0.503307

std\_Density 0.047196 0.012949

mean\_ElectronAffinity 0.321216 0.330667

wtd\_mean\_ElectronAffinity 0.063204 0.029280

gmean\_ElectronAffinity 0.500995 0.504130

wtd\_gmean\_ElectronAffinity 0.111913 0.154391

entropy\_ElectronAffinity 0.491108 0.523597

wtd\_entropy\_ElectronAffinity 0.247190 0.279732

wtd\_range\_ElectronAffinity 0.229217 0.169959

std\_ElectronAffinity 0.244811 0.209334

wtd\_std\_ElectronAffinity 0.259158 0.250939

mean\_FusionHeat 0.488123 0.478450

wtd\_mean\_FusionHeat 0.439682 0.499794

gmean\_FusionHeat 0.604493 0.572989

wtd\_gmean\_FusionHeat 0.518345 0.571290

entropy\_FusionHeat 0.500919 0.559244

wtd\_entropy\_FusionHeat 0.504138 0.589524

wtd\_range\_FusionHeat 0.288012 0.384025

std\_FusionHeat 0.096527 0.135557

wtd\_std\_FusionHeat 0.084557 0.116743

mean\_ThermalConductivity 0.355912 0.373348

wtd\_mean\_ThermalConductivity 0.253874 0.276539

gmean\_ThermalConductivity 0.406019 0.418818

entropy\_ThermalConductivity 0.083684 0.065917

wtd\_entropy\_ThermalConductivity 0.110553 0.141321

wtd\_range\_ThermalConductivity 0.388703 0.413807

std\_ThermalConductivity 0.671569 0.714405

wtd\_std\_ThermalConductivity 0.678028 0.735073

mean\_Valence NaN 0.937103

wtd\_mean\_Valence NaN NaN

gmean\_Valence NaN NaN

wtd\_gmean\_Valence NaN NaN

entropy\_Valence NaN NaN

wtd\_entropy\_Valence NaN NaN

range\_Valence NaN NaN

wtd\_range\_Valence NaN NaN

std\_Valence NaN NaN

wtd\_std\_Valence NaN NaN

gmean\_Valence wtd\_gmean\_Valence \

number\_of\_elements 0.618512 0.659268

mean\_atomic\_mass 0.392153 0.321399

wtd\_mean\_atomic\_mass 0.539780 0.548981

gmean\_atomic\_mass 0.511508 0.450357

wtd\_gmean\_atomic\_mass 0.608417 0.623261

entropy\_atomic\_mass 0.554477 0.592817

wtd\_entropy\_atomic\_mass 0.586226 0.650247

std\_atomic\_mass 0.330630 0.352901

wtd\_std\_atomic\_mass 0.386043 0.414098

mean\_fie 0.142754 0.122387

gmean\_fie 0.046003 0.054233

wtd\_gmean\_fie 0.465677 0.480221

entropy\_fie 0.608303 0.637874

wtd\_entropy\_fie 0.369296 0.375613

wtd\_range\_fie 0.429217 0.438015

std\_fie 0.740587 0.713240

mean\_atomic\_radius 0.086126 0.032473

gmean\_atomic\_radius 0.372333 0.322096

wtd\_gmean\_atomic\_radius 0.636740 0.654187

entropy\_atomic\_radius 0.590868 0.623531

wtd\_entropy\_atomic\_radius 0.635244 0.677313

wtd\_range\_atomic\_radius 0.423345 0.455683

std\_atomic\_radius 0.722552 0.696988

wtd\_std\_atomic\_radius 0.711929 0.736105

mean\_Density 0.721242 0.643125

wtd\_gmean\_Density 0.795337 0.814657

entropy\_Density 0.430860 0.479727

wtd\_entropy\_Density 0.326816 0.402037

wtd\_range\_Density 0.451274 0.507844

std\_Density 0.016267 0.028931

mean\_ElectronAffinity 0.289160 0.312905

wtd\_mean\_ElectronAffinity 0.074342 0.031283

gmean\_ElectronAffinity 0.480164 0.495592

wtd\_gmean\_ElectronAffinity 0.110341 0.161498

entropy\_ElectronAffinity 0.505040 0.537412

wtd\_entropy\_ElectronAffinity 0.273162 0.310245

wtd\_range\_ElectronAffinity 0.237823 0.164888

std\_ElectronAffinity 0.280599 0.234673

wtd\_std\_ElectronAffinity 0.293796 0.280100

mean\_FusionHeat 0.494814 0.488039

wtd\_mean\_FusionHeat 0.436508 0.503301

gmean\_FusionHeat 0.622305 0.587671

wtd\_gmean\_FusionHeat 0.519993 0.578353

entropy\_FusionHeat 0.490564 0.560693

wtd\_entropy\_FusionHeat 0.503019 0.598601

wtd\_range\_FusionHeat 0.276334 0.385503

std\_FusionHeat 0.077105 0.127758

wtd\_std\_FusionHeat 0.069348 0.106550

mean\_ThermalConductivity 0.330129 0.354064

wtd\_mean\_ThermalConductivity 0.231307 0.260902

gmean\_ThermalConductivity 0.411016 0.426053

entropy\_ThermalConductivity 0.109405 0.086614

wtd\_entropy\_ThermalConductivity 0.068933 0.105337

wtd\_range\_ThermalConductivity 0.364107 0.394153

std\_ThermalConductivity 0.640881 0.695269

wtd\_std\_ThermalConductivity 0.643793 0.715135

mean\_Valence 0.989911 0.940001

wtd\_mean\_Valence 0.917905 0.994939

gmean\_Valence NaN 0.933036

wtd\_gmean\_Valence NaN NaN

entropy\_Valence NaN NaN

wtd\_entropy\_Valence NaN NaN

range\_Valence NaN NaN

wtd\_range\_Valence NaN NaN

std\_Valence NaN NaN

wtd\_std\_Valence NaN NaN

entropy\_Valence wtd\_entropy\_Valence \

number\_of\_elements 0.967832 0.892559

mean\_atomic\_mass 0.156786 0.145610

wtd\_mean\_atomic\_mass 0.375718 0.331025

gmean\_atomic\_mass 0.306246 0.307662

wtd\_gmean\_atomic\_mass 0.477785 0.448072

entropy\_atomic\_mass 0.963621 0.861479

wtd\_entropy\_atomic\_mass 0.897636 0.918284

std\_atomic\_mass 0.518439 0.540601

wtd\_std\_atomic\_mass 0.559955 0.606574

mean\_fie 0.156312 0.260380

gmean\_fie 0.009336 0.125304

wtd\_gmean\_fie 0.431439 0.382479

entropy\_fie 0.992726 0.907923

wtd\_entropy\_fie 0.730266 0.862061

wtd\_range\_fie 0.379288 0.137482

std\_fie 0.689995 0.689920

mean\_atomic\_radius 0.009756 0.065894

gmean\_atomic\_radius 0.243188 0.310648

wtd\_gmean\_atomic\_radius 0.535898 0.485474

entropy\_atomic\_radius 0.989546 0.898930

wtd\_entropy\_atomic\_radius 0.919184 0.951463

wtd\_range\_atomic\_radius 0.332892 0.512876

std\_atomic\_radius 0.670556 0.660613

wtd\_std\_atomic\_radius 0.738201 0.730072

mean\_Density 0.425309 0.396193

wtd\_gmean\_Density 0.657725 0.638629

entropy\_Density 0.900579 0.779776

wtd\_entropy\_Density 0.780306 0.837233

wtd\_range\_Density 0.328885 0.413359

std\_Density 0.223375 0.275794

mean\_ElectronAffinity 0.110131 0.033251

wtd\_mean\_ElectronAffinity 0.214876 0.176687

gmean\_ElectronAffinity 0.346279 0.279780

wtd\_gmean\_ElectronAffinity 0.022912 0.094571

entropy\_ElectronAffinity 0.904659 0.823543

wtd\_entropy\_ElectronAffinity 0.653655 0.759590

wtd\_range\_ElectronAffinity 0.286828 0.133700

std\_ElectronAffinity 0.441397 0.472159

wtd\_std\_ElectronAffinity 0.494662 0.579242

mean\_FusionHeat 0.434774 0.395375

wtd\_mean\_FusionHeat 0.451595 0.449115

gmean\_FusionHeat 0.514469 0.473427

wtd\_gmean\_FusionHeat 0.526863 0.529789

entropy\_FusionHeat 0.921445 0.824121

wtd\_entropy\_FusionHeat 0.866068 0.908728

wtd\_range\_FusionHeat 0.355642 0.440176

std\_FusionHeat 0.090183 0.074018

wtd\_std\_FusionHeat 0.036513 0.007079

mean\_ThermalConductivity 0.240317 0.195642

wtd\_mean\_ThermalConductivity 0.196164 0.218736

gmean\_ThermalConductivity 0.502534 0.511838

entropy\_ThermalConductivity 0.484029 0.389657

wtd\_entropy\_ThermalConductivity 0.199033 0.254061

wtd\_range\_ThermalConductivity 0.335672 0.302291

std\_ThermalConductivity 0.641505 0.589997

wtd\_std\_ThermalConductivity 0.691772 0.663885

mean\_Valence 0.619428 0.558969

wtd\_mean\_Valence 0.652105 0.636286

gmean\_Valence 0.615971 0.563527

wtd\_gmean\_Valence 0.656235 0.648708

entropy\_Valence NaN 0.910822

wtd\_entropy\_Valence NaN NaN

range\_Valence NaN NaN

wtd\_range\_Valence NaN NaN

std\_Valence NaN NaN

wtd\_std\_Valence NaN NaN

range\_Valence wtd\_range\_Valence \

number\_of\_elements 0.231874 0.447770

mean\_atomic\_mass 0.107450 0.168633

wtd\_mean\_atomic\_mass 0.039155 0.330904

gmean\_atomic\_mass 0.165010 0.272303

wtd\_gmean\_atomic\_mass 0.078641 0.409674

entropy\_atomic\_mass 0.203207 0.350047

wtd\_entropy\_atomic\_mass 0.161350 0.546910

std\_atomic\_mass 0.192347 0.273598

wtd\_std\_atomic\_mass 0.177675 0.391024

mean\_fie 0.205324 0.235069

gmean\_fie 0.191984 0.131647

wtd\_gmean\_fie 0.036008 0.340174

entropy\_fie 0.260018 0.389817

wtd\_entropy\_fie 0.361073 0.454246

wtd\_range\_fie 0.104309 0.061797

std\_fie 0.151535 0.500329

mean\_atomic\_radius 0.238393 0.109244

gmean\_atomic\_radius 0.213301 0.304293

wtd\_gmean\_atomic\_radius 0.042693 0.456791

entropy\_atomic\_radius 0.259130 0.375326

wtd\_entropy\_atomic\_radius 0.226873 0.576320

wtd\_range\_atomic\_radius 0.069213 0.722955

std\_atomic\_radius 0.084398 0.469035

wtd\_std\_atomic\_radius 0.039561 0.543134

mean\_Density 0.083691 0.395385

wtd\_gmean\_Density 0.087742 0.571594

entropy\_Density 0.156710 0.261252

wtd\_entropy\_Density 0.250488 0.425261

wtd\_range\_Density 0.008209 0.634471

std\_Density 0.259181 0.052541

mean\_ElectronAffinity 0.193715 0.104693

wtd\_mean\_ElectronAffinity 0.052746 0.029781

gmean\_ElectronAffinity 0.082682 0.247611

wtd\_gmean\_ElectronAffinity 0.050487 0.143140

entropy\_ElectronAffinity 0.265024 0.324851

wtd\_entropy\_ElectronAffinity 0.334608 0.393388

wtd\_range\_ElectronAffinity 0.041614 0.132737

std\_ElectronAffinity 0.310349 0.189659

wtd\_std\_ElectronAffinity 0.310218 0.312077

mean\_FusionHeat 0.035540 0.286468

wtd\_mean\_FusionHeat 0.005202 0.365964

gmean\_FusionHeat 0.110862 0.333871

wtd\_gmean\_FusionHeat 0.034940 0.425550

entropy\_FusionHeat 0.120887 0.342548

wtd\_entropy\_FusionHeat 0.170917 0.547006

wtd\_range\_FusionHeat 0.044222 0.480671

std\_FusionHeat 0.162875 0.108712

wtd\_std\_FusionHeat 0.150326 0.027364

mean\_ThermalConductivity 0.195944 0.177460

wtd\_mean\_ThermalConductivity 0.160914 0.189599

gmean\_ThermalConductivity 0.077929 0.353427

entropy\_ThermalConductivity 0.337554 0.042422

wtd\_entropy\_ThermalConductivity 0.421079 0.038302

wtd\_range\_ThermalConductivity 0.178406 0.189897

std\_ThermalConductivity 0.174000 0.443612

wtd\_std\_ThermalConductivity 0.187465 0.509148

mean\_Valence 0.085892 0.554338

wtd\_mean\_Valence 0.111249 0.683700

gmean\_Valence 0.041651 0.532805

wtd\_gmean\_Valence 0.033007 0.681415

entropy\_Valence 0.155410 0.417283

wtd\_entropy\_Valence 0.204411 0.639818

range\_Valence NaN 0.114000

wtd\_range\_Valence NaN NaN

std\_Valence NaN NaN

wtd\_std\_Valence NaN NaN

std\_Valence wtd\_std\_Valence

number\_of\_elements 0.105365 0.035216

mean\_atomic\_mass 0.080279 0.081253

wtd\_mean\_atomic\_mass 0.003681 0.077323

gmean\_atomic\_mass 0.124627 0.117336

wtd\_gmean\_atomic\_mass 0.033313 0.030361

entropy\_atomic\_mass 0.087838 0.018758

wtd\_entropy\_atomic\_mass 0.056309 0.006937

std\_atomic\_mass 0.144106 0.122547

wtd\_std\_atomic\_mass 0.123636 0.155702

mean\_fie 0.161033 0.170006

gmean\_fie 0.166488 0.194419

wtd\_gmean\_fie 0.000583 0.105682

entropy\_fie 0.139747 0.077940

wtd\_entropy\_fie 0.249305 0.332966

wtd\_range\_fie 0.107349 0.307760

std\_fie 0.063530 0.000063

mean\_atomic\_radius 0.220036 0.291992

gmean\_atomic\_radius 0.170579 0.200041

wtd\_gmean\_atomic\_radius 0.006524 0.096146

entropy\_atomic\_radius 0.139544 0.077988

wtd\_entropy\_atomic\_radius 0.114904 0.093575

wtd\_range\_atomic\_radius 0.018608 0.061663

std\_atomic\_radius 0.019095 0.056573

wtd\_std\_atomic\_radius 0.031263 0.109204

mean\_Density 0.024797 0.016377

wtd\_gmean\_Density 0.014400 0.056154

entropy\_Density 0.052318 0.001252

wtd\_entropy\_Density 0.146199 0.152149

wtd\_range\_Density 0.055460 0.048841

std\_Density 0.257379 0.198795

mean\_ElectronAffinity 0.200917 0.218167

wtd\_mean\_ElectronAffinity 0.051092 0.010135

gmean\_ElectronAffinity 0.116236 0.143877

wtd\_gmean\_ElectronAffinity 0.027394 0.079913

entropy\_ElectronAffinity 0.150735 0.094690

wtd\_entropy\_ElectronAffinity 0.226893 0.303421

wtd\_range\_ElectronAffinity 0.054814 0.080619

std\_ElectronAffinity 0.253786 0.230442

wtd\_std\_ElectronAffinity 0.258989 0.264354

mean\_FusionHeat 0.008708 0.045541

wtd\_mean\_FusionHeat 0.053801 0.092672

gmean\_FusionHeat 0.060116 0.008076

wtd\_gmean\_FusionHeat 0.022502 0.068536

entropy\_FusionHeat 0.013797 0.016918

wtd\_entropy\_FusionHeat 0.063734 0.056473

wtd\_range\_FusionHeat 0.100086 0.076896

std\_FusionHeat 0.173417 0.157327

wtd\_std\_FusionHeat 0.151733 0.182440

mean\_ThermalConductivity 0.192302 0.261168

wtd\_mean\_ThermalConductivity 0.156601 0.206584

gmean\_ThermalConductivity 0.025488 0.005427

entropy\_ThermalConductivity 0.245604 0.255835

wtd\_entropy\_ThermalConductivity 0.347741 0.447827

wtd\_range\_ThermalConductivity 0.171831 0.274266

std\_ThermalConductivity 0.209550 0.306211

wtd\_std\_ThermalConductivity 0.232050 0.323640

mean\_Valence 0.164058 0.203341

wtd\_mean\_Valence 0.186270 0.258593

gmean\_Valence 0.034589 0.088388

wtd\_gmean\_Valence 0.108469 0.168962

entropy\_Valence 0.034580 0.017031

wtd\_entropy\_Valence 0.089774 0.081284

range\_Valence 0.973788 0.867393

wtd\_range\_Valence 0.184380 0.132162

std\_Valence NaN 0.885856

wtd\_std\_Valence NaN NaN

[66 rows x 66 columns]

In [10]:

to\_drop **=** [column **for** column **in** upper\_tri**.**columns **if** any(upper\_tri[column] **>** 0.95)]

print(); print((to\_drop))

['wtd\_gmean\_atomic\_mass', 'gmean\_fie', 'entropy\_fie', 'entropy\_atomic\_radius', 'wtd\_entropy\_atomic\_radius', 'wtd\_gmean\_FusionHeat', 'wtd\_std\_ThermalConductivity', 'gmean\_Valence', 'wtd\_gmean\_Valence', 'entropy\_Valence', 'wtd\_entropy\_Valence', 'std\_Valence']

In [11]:

*# drop variables which are collinear to each other*

train\_df\_cleaned **=** train\_df**.**drop(to\_drop, axis**=**1)

train\_df\_cleaned**.**head()

Out[11]:

|  | **number\_of\_elements** | **mean\_atomic\_mass** | **wtd\_mean\_atomic\_mass** | **gmean\_atomic\_mass** | **entropy\_atomic\_mass** | **wtd\_entropy\_atomic\_mass** | **std\_atomic\_mass** | **wtd\_std\_atomic\_mass** | **mean\_fie** | **wtd\_gmean\_fie** | **...** | **gmean\_ThermalConductivity** | **entropy\_ThermalConductivity** | **wtd\_entropy\_ThermalConductivity** | **wtd\_range\_ThermalConductivity** | **std\_ThermalConductivity** | **mean\_Valence** | **wtd\_mean\_Valence** | **range\_Valence** | **wtd\_range\_Valence** | **wtd\_std\_Valence** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 4 | 88.944468 | 57.862692 | 66.361592 | 1.181795 | 1.062396 | 51.968828 | 53.622535 | 775.425 | 938.016780 | ... | 7.062488 | 0.308148 | 0.262848 | 57.127669 | 168.854244 | 2.25 | 2.257143 | 1 | 1.085714 | 0.437059 |
| **1** | 5 | 92.729214 | 58.518416 | 73.132787 | 1.449309 | 1.057755 | 47.094633 | 53.979870 | 766.440 | 938.745413 | ... | 16.064228 | 0.847404 | 0.567706 | 51.413383 | 198.554600 | 2.00 | 2.257143 | 2 | 1.128571 | 0.468606 |
| **2** | 4 | 88.944468 | 57.885242 | 66.361592 | 1.181795 | 0.975980 | 51.968828 | 53.656268 | 775.425 | 939.009036 | ... | 7.062488 | 0.308148 | 0.250477 | 57.127669 | 168.854244 | 2.25 | 2.271429 | 1 | 1.114286 | 0.444697 |
| **3** | 4 | 88.944468 | 57.873967 | 66.361592 | 1.181795 | 1.022291 | 51.968828 | 53.639405 | 775.425 | 938.512777 | ... | 7.062488 | 0.308148 | 0.257045 | 57.127669 | 168.854244 | 2.25 | 2.264286 | 1 | 1.100000 | 0.440952 |
| **4** | 4 | 88.944468 | 57.840143 | 66.361592 | 1.181795 | 1.129224 | 51.968828 | 53.588771 | 775.425 | 937.025573 | ... | 7.062488 | 0.308148 | 0.272820 | 57.127669 | 168.854244 | 2.25 | 2.242857 | 1 | 1.057143 | 0.428809 |

5 rows × 54 columns

In [12]:

*#heatmap - correlation matrix*

plt**.**figure(figsize**=**(55, 50)) *#code reference (5-1)*

sns**.**heatmap(train\_df\_cleaned**.**corr(), annot**=True**)

plt**.**title('HeatMap-Correlation Matrix')

Out[12]:

Text(0.5, 1.0, 'HeatMap-Correlation Matrix')

A picture containing text, colorful, mosaic, fabric

Description automatically generated

In [13]:

*#Visualizing the hist of data*

train\_df\_cleaned**.**hist(bins**=**50,figsize**=**(25,30))

plt**.**show()

A picture containing window

Description automatically generated

#### Read and analyze unique\_m csv file

In [14]:

*# read unique\_m csv files*

unique\_m\_df **=** pd**.**read\_csv('unique\_m.csv')

The "material" column which comprises of all elements used as materials, it is not useful to us so we will drop it from our data frame. This will also help to reduce the size of any pairwise comparisons without additional restricting.

In [15]:

unique\_m\_df**.**drop(['material'], inplace**=True**, axis**=**1)

unique\_m\_df**.**head()

Out[15]:

|  | **H** | **He** | **Li** | **Be** | **B** | **C** | **N** | **O** | **F** | **Ne** | **...** | **Pt** | **Au** | **Hg** | **Tl** | **Pb** | **Bi** | **Po** | **At** | **Rn** | **critical\_temp** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0.0 | 0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 4.0 | 0.0 | 0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 | 29.0 |
| **1** | 0.0 | 0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 4.0 | 0.0 | 0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 | 26.0 |
| **2** | 0.0 | 0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 4.0 | 0.0 | 0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 | 19.0 |
| **3** | 0.0 | 0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 4.0 | 0.0 | 0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 | 22.0 |
| **4** | 0.0 | 0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 4.0 | 0.0 | 0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 | 23.0 |

5 rows × 87 columns

In [16]:

*# Validate null values in unique\_m csv file*

unique\_m\_df**.**isnull()**.**sum()**.**sum()

Out[16]:

0

In [17]:

*# Passing a value of zero for the parameter will filter all the features with zero variance.*

constant\_filter **=** VarianceThreshold(threshold**=**0)

*# apply this filter to our dataframe*

constant\_filter**.**fit(unique\_m\_df)

*# get\_support() method will give the number of non-constant features.*

len(unique\_m\_df**.**columns[constant\_filter**.**get\_support()])

*# get columns names of non constant columns*

non\_constant\_columns **=** [column **for** column **in** unique\_m\_df**.**columns

**if** column **in** unique\_m\_df**.**columns[constant\_filter**.**get\_support()]]

print(len(non\_constant\_columns))

unique\_m\_df\_cleaned **=** pd**.**DataFrame(constant\_filter**.**transform(unique\_m\_df), columns**=**non\_constant\_columns)

unique\_m\_df\_cleaned**.**shape

78

Out[17]:

(21263, 78)

In [18]:

*# get columns names of non constant columns*

constant\_columns **=** [column **for** column **in** unique\_m\_df**.**columns

**if** **not** column **in** unique\_m\_df**.**columns[constant\_filter**.**get\_support()]]

print(len(constant\_columns))

print(constant\_columns)

9

['He', 'Ne', 'Ar', 'Kr', 'Xe', 'Pm', 'Po', 'At', 'Rn']

In [19]:

*#heatmap - correlation matrix*

plt**.**figure(figsize**=**(55, 50)) *#code reference (5-1)*

sns**.**heatmap(unique\_m\_df\_cleaned**.**corr(), annot**=True**)

plt**.**title('HeatMap-Correlation Matrix')

Out[19]:

Text(0.5, 1.0, 'HeatMap-Correlation Matrix')

A picture containing text

Description automatically generated

In [20]:

*#Visualizing the hist of data*

unique\_m\_df\_cleaned**.**hist(bins**=**50,figsize**=**(25,30))

plt**.**show()

A picture containing window, shoji, building

Description automatically generated

Most of the columns in unique\_m data frame having value ZERO for most of the observations which is the reason for high multicollinearity among independent features

#### Concatenate both csv files

In [21]:

df **=** pd**.**concat([train\_df\_cleaned,unique\_m\_df\_cleaned],axis**=**1)

df**.**shape

Out[21]:

(21263, 132)

In [22]:

*#data frame with all variables*

df\_all **=** pd**.**concat([train\_orin\_df,unique\_m\_df],axis**=**1)

df\_all**.**shape

Out[22]:

(21263, 168)

In [23]:

*# Create X (Independent variables) and y(target) from the dataframe with reduced variables*

X **=** df**.**drop(['critical\_temp'],axis**=**1)

ind\_columns **=** df**.**drop('critical\_temp',axis**=**1)**.**columns

y **=** df['critical\_temp']

In [24]:

*# Create X (Independent variables) and y(target) from the dataframe with all variables*

X\_all **=** df\_all**.**drop(['critical\_temp'],axis**=**1)

ind\_columns\_all **=** df\_all**.**drop('critical\_temp',axis**=**1)**.**columns

y\_all **=** df\_all['critical\_temp']

In [25]:

*# Normalize the data*

scaler **=** StandardScaler()

X\_scaled **=** scaler**.**fit\_transform(X)

In [26]:

*# Normalize the data (for all variables)*

X\_all\_scaled **=** scaler**.**fit\_transform(X\_all)

In [27]:

*# cross validation with k=10*

cv **=** KFold(n\_splits**=**10, random\_state**=**1234, shuffle**=True**)

#### Multiple Linear Regression

#### Method - 1 (dataset with all variable)

In [28]:

*# Regression Model*

mlr\_all **=** linear\_model**.**LinearRegression()**.**fit(X\_all\_scaled, y\_all)

*# score the regression model based on Root Mean Squared Error*

mlr\_all\_preds **=** cross\_val\_predict(mlr\_all, X\_all\_scaled, y\_all, cv**=**cv)

mlr\_all\_scores **=** cross\_val\_score(mlr\_all, X\_all\_scaled, y\_all, scoring**=**'neg\_root\_mean\_squared\_error', cv**=**cv)

print('Average of RMSE for 10 folds: ', np**.**mean(mlr\_all\_scores))

print('Standard Deviation of RMSE for 10 folds: ', np**.**std(mlr\_all\_scores))

Average of RMSE for 10 folds: -22.567912627438222

Standard Deviation of RMSE for 10 folds: 12.42901305546248

In [29]:

mlr\_all\_scores

Out[29]:

array([-16.66222827, -17.09389124, -16.916193 , -16.61280737,

-16.51606893, -36.6341099 , -16.51435072, -16.97374952,

-16.43790687, -55.31782046])

Some of the observations in 7th and 10th folds are not same as most of the observations, which is causing high RMSE in those two folds.

#### Method - 2 (dataset with reduced variables)

In [30]:

*# Regression Model*

mlr **=** linear\_model**.**LinearRegression()**.**fit(X\_scaled, y)

*# score the regression model based on Root Mean Squared Error*

mlr\_preds **=** cross\_val\_predict(mlr, X\_scaled, y, cv**=**cv)

mlr\_scores **=** cross\_val\_score(mlr, X\_scaled, y, scoring**=**'neg\_root\_mean\_squared\_error', cv**=**cv)

print('Average of RMSE for 10 folds: ', np**.**mean(mlr\_scores))

print('Standard Deviation of RMSE for 10 folds: ', np**.**std(mlr\_scores))

Average of RMSE for 10 folds: -22.67985342685359

Standard Deviation of RMSE for 10 folds: 10.941256856189279

In [31]:

mlr\_scores

Out[31]:

array([-17.11372273, -17.75714692, -17.50256374, -17.31075339,

-17.2494668 , -38.71287444, -17.22180329, -17.69255286,

-16.88444786, -49.35320223])

With respect to RMSE Linear regression model with all variables is little less than RMSE with reduced variables. But standard deviation is more with all variables than reduced variables. Plot the residuals with reduced variables

In [32]:

rezdf **=** df

rezdf['preds'] **=** mlr\_preds

rezdf['resids'] **=** (rezdf['preds'] **-** y)

*# residual plot*

fig **=** plt**.**figure(figsize **=** (12, 8))

sns**.**relplot(x**=**'preds', y**=**'resids', data**=**rezdf, color**=**'red')

plt**.**title('Residual plot', size**=**20)

plt**.**xlim([**-**200, 200])

plt**.**ylim([**-**200, 200])

plt**.**xlabel('Predictions', size**=**14)

plt**.**ylabel('Residuals', size**=**14);

<Figure size 864x576 with 0 Axes>

Chart, scatter chart

Description automatically generated

In [33]:

*# Plot the histogram of the error terms*

fig **=** plt**.**figure()

sns**.**distplot(rezdf['resids'], bins **=** 20)

fig**.**suptitle('Error Terms', fontsize **=** 20) *# Plot heading*

plt**.**xlabel('Errors', fontsize **=** 18)

plt**.**xlim([**-**200, 200])

Out[33]:

(-200.0, 200.0)

Chart, histogram

Description automatically generated

In [34]:

*# intercept and regression model coefficients*

coef2 **=** mlr**.**coef\_**.**tolist()

coefdf2 **=** pd**.**DataFrame(ind\_columns, columns**=**['Variable'])

coefdf2['Coefficient'] **=** coef2

print('Intercept: \n', mlr**.**intercept\_)

coefdf2

Intercept:

34.421219135352445

Out[34]:

|  | **Variable** | **Coefficient** |
| --- | --- | --- |
| **0** | number\_of\_elements | 1.972091 |
| **1** | mean\_atomic\_mass | 7.769634 |
| **2** | wtd\_mean\_atomic\_mass | -7.277440 |
| **3** | gmean\_atomic\_mass | -0.923948 |
| **4** | entropy\_atomic\_mass | -9.686440 |
| **...** | ... | ... |
| **126** | Au | -0.528809 |
| **127** | Hg | 1.048743 |
| **128** | Tl | 1.434886 |
| **129** | Pb | 0.385434 |
| **130** | Bi | 4.914820 |

131 rows × 2 columns

#### LASSO Regression

#### Method - 1 (Dataset with all variables)

In [35]:

*# hyperparameter tuning with grid search*

lasso\_all\_reg **=** Lasso()

param\_grid **=** {'alpha':[0.001, 0.005, 0.01, 0.03, 0.1, 0.3, 0.6, 1]}

grid\_search\_lasso\_all **=** GridSearchCV(estimator**=**lasso\_all\_reg, param\_grid**=**param\_grid, n\_jobs**=-**1,

cv**=**cv, scoring**=**'neg\_root\_mean\_squared\_error')

In [36]:

**%%time**

grid\_result\_lasso\_all **=** grid\_search\_lasso\_all**.**fit(X\_all\_scaled, y\_all)

*# summarize results*

print("Best: %f using %s" **%** (grid\_result\_lasso\_all**.**best\_score\_, grid\_result\_lasso\_all**.**best\_params\_))

means\_all **=** grid\_result\_lasso\_all**.**cv\_results\_['mean\_test\_score']

stds\_all **=** grid\_result\_lasso\_all**.**cv\_results\_['std\_test\_score']

params\_all **=** grid\_result\_lasso\_all**.**cv\_results\_['params']

**for** mean, stdev, param **in** zip(means\_all, stds\_all, params\_all):

print("%f (%f) with: %r" **%** (mean, stdev, param))

Best: -17.957879 using {'alpha': 0.3}

-22.231906 (11.430279) with: {'alpha': 0.001}

-21.851813 (10.649407) with: {'alpha': 0.005}

-21.482859 (9.857907) with: {'alpha': 0.01}

-20.563842 (7.779756) with: {'alpha': 0.03}

-18.608007 (3.511564) with: {'alpha': 0.1}

-17.957879 (0.284759) with: {'alpha': 0.3}

-18.495503 (0.267034) with: {'alpha': 0.6}

-19.006189 (0.254651) with: {'alpha': 1}

Wall time: 1min 26s

#### Method - 2 (Reduced dataset)

In [37]:

*# hyperparameter tuning with grid search*

lasso\_reg **=** Lasso()

grid\_search\_lasso **=** GridSearchCV(estimator**=**lasso\_reg, param\_grid**=**param\_grid, n\_jobs**=-**1,

cv**=**cv, scoring**=**'neg\_root\_mean\_squared\_error')

In [38]:

**%%time**

grid\_result\_lasso **=** grid\_search\_lasso**.**fit(X\_scaled, y)

*# summarize results*

print("Best: %f using %s" **%** (grid\_result\_lasso**.**best\_score\_, grid\_result\_lasso**.**best\_params\_))

means **=** grid\_result\_lasso**.**cv\_results\_['mean\_test\_score']

stds **=** grid\_result\_lasso**.**cv\_results\_['std\_test\_score']

params **=** grid\_result\_lasso**.**cv\_results\_['params']

**for** mean, stdev, param **in** zip(means, stds, params):

print("%f (%f) with: %r" **%** (mean, stdev, param))

Best: -18.345913 using {'alpha': 0.3}

-22.553782 (10.699814) with: {'alpha': 0.001}

-22.221466 (10.110054) with: {'alpha': 0.005}

-21.834902 (9.429106) with: {'alpha': 0.01}

-20.662639 (7.582365) with: {'alpha': 0.03}

-19.177542 (4.434427) with: {'alpha': 0.1}

-18.345913 (0.329406) with: {'alpha': 0.3}

-18.715247 (0.302464) with: {'alpha': 0.6}

-19.121818 (0.267632) with: {'alpha': 1}

Wall time: 1min 4s

With respect to RMSE, Lasso regression model with all variables is better than reduced variables at alpha of 0.3

In [39]:

*# bset model with all variables*

lasso\_reg\_best **=** Lasso(alpha **=** 0.3)

*# Lasso Regression Model*

lasso **=** lasso\_reg\_best**.**fit(X\_all\_scaled, y\_all)

*# score the regression model based on Root Mean Squared Error*

lasso\_preds **=** cross\_val\_predict(lasso, X\_all\_scaled, y\_all, cv**=**cv)

lasso\_scores **=** cross\_val\_score(lasso, X\_all\_scaled, y\_all, scoring**=**'neg\_root\_mean\_squared\_error', cv**=**cv)

print('Average of RMSE for 10 folds: ', np**.**mean(lasso\_scores))

print('Standard Deviation of RMSE for 10 folds: ', np**.**std(lasso\_scores))

Average of RMSE for 10 folds: -17.95787853561992

Standard Deviation of RMSE for 10 folds: 0.28475898113408626

In [40]:

lasso\_scores

Out[40]:

array([-17.49501274, -18.18279108, -18.04697072, -17.9540785 ,

-17.99042669, -18.24794041, -17.76672689, -18.01578225,

-17.48547188, -18.39358418])

In [41]:

rezdf\_lasso **=** df

rezdf\_lasso['preds'] **=** lasso\_preds

rezdf\_lasso['resids'] **=** (rezdf\_lasso['preds'] **-** y)

*# residual plot*

fig **=** plt**.**figure(figsize **=** (12, 8))

sns**.**relplot(x**=**'preds', y**=**'resids', data**=**rezdf\_lasso, color**=**'red')

plt**.**title('Residual plot', size**=**20)

plt**.**xlim([**-**200, 200])

plt**.**ylim([**-**200, 200])

plt**.**xlabel('Predictions', size**=**14)

plt**.**ylabel('Residuals', size**=**14);

<Figure size 864x576 with 0 Axes>

Scatter chart

Description automatically generated

In [42]:

*# Plot the histogram of the error terms*

fig **=** plt**.**figure()

sns**.**distplot(rezdf\_lasso['resids'], bins **=** 20)

fig**.**suptitle('Error Terms', fontsize **=** 20) *# Plot heading*

plt**.**xlabel('Errors', fontsize **=** 18)

plt**.**xlim([**-**200, 200])

Out[42]:

(-200.0, 200.0)

Histogram

Description automatically generated with low confidence

In [43]:

*# intercept and regression model coefficients*

coef2\_lasso **=** lasso**.**coef\_**.**tolist()

coefdf2\_lasso **=** pd**.**DataFrame(ind\_columns\_all, columns**=**['Variable'])

coefdf2\_lasso['Coefficient'] **=** coef2\_lasso

print('Intercept: \n', lasso**.**intercept\_)

coefdf2\_lasso

Intercept:

34.421219135352494

Out[43]:

|  | **Variable** | **Coefficient** |
| --- | --- | --- |
| **0** | number\_of\_elements | 0.000000 |
| **1** | mean\_atomic\_mass | -0.000000 |
| **2** | wtd\_mean\_atomic\_mass | -0.117541 |
| **3** | gmean\_atomic\_mass | -0.000000 |
| **4** | wtd\_gmean\_atomic\_mass | -0.000000 |
| **...** | ... | ... |
| **162** | Pb | 0.000000 |
| **163** | Bi | 3.851411 |
| **164** | Po | 0.000000 |
| **165** | At | 0.000000 |
| **166** | Rn | 0.000000 |

167 rows × 2 columns

#### Lasso Features with Coefficients

In [44]:

pd**.**set\_option('display.max\_rows', 150)

coeff**=** pd**.**DataFrame(coefdf2\_lasso[coefdf2\_lasso**.**Coefficient**!=**0])

coeff['Coefficient Absolute'] **=** coeff['Coefficient']**.**abs()

coeff**.**sort\_values('Coefficient Absolute', ascending**=False**)

Out[44]:

|  | **Variable** | **Coefficient** | **Coefficient Absolute** |
| --- | --- | --- | --- |
| **62** | wtd\_mean\_ThermalConductivity | 10.603133 | 10.603133 |
| **64** | wtd\_gmean\_ThermalConductivity | -8.911775 | 8.911775 |
| **136** | Ba | 8.757244 | 8.757244 |
| **7** | range\_atomic\_mass | 5.437564 | 5.437564 |
| **80** | wtd\_std\_Valence | -4.053058 | 4.053058 |
| **163** | Bi | 3.851411 | 3.851411 |
| **10** | wtd\_std\_atomic\_mass | -3.343854 | 3.343854 |
| **94** | Si | -3.238540 | 3.238540 |
| **44** | wtd\_gmean\_ElectronAffinity | -2.960786 | 2.960786 |
| **6** | wtd\_entropy\_atomic\_mass | 2.960010 | 2.960010 |
| **70** | wtd\_std\_ThermalConductivity | 2.754461 | 2.754461 |
| **46** | wtd\_entropy\_ElectronAffinity | -2.750278 | 2.750278 |
| **66** | wtd\_entropy\_ThermalConductivity | 2.658724 | 2.658724 |
| **100** | Ca | 2.564098 | 2.564098 |
| **33** | gmean\_Density | -1.821968 | 1.821968 |
| **27** | range\_atomic\_radius | 1.814414 | 1.814414 |
| **11** | mean\_fie | 1.598674 | 1.598674 |
| **160** | Hg | 1.453957 | 1.453957 |
| **60** | wtd\_std\_FusionHeat | -1.444920 | 1.444920 |
| **161** | Tl | 1.118754 | 1.118754 |
| **113** | As | -1.048332 | 1.048332 |
| **8** | wtd\_range\_atomic\_mass | -1.022679 | 1.022679 |
| **127** | Ag | -0.990668 | 0.990668 |
| **96** | S | -0.967576 | 0.967576 |
| **97** | Cl | -0.865930 | 0.865930 |
| **56** | wtd\_entropy\_FusionHeat | 0.730239 | 0.730239 |
| **49** | std\_ElectronAffinity | 0.723023 | 0.723023 |
| **118** | Sr | 0.665290 | 0.665290 |
| **138** | Ce | -0.650766 | 0.650766 |
| **140** | Nd | -0.612786 | 0.612786 |
| **112** | Ge | -0.605506 | 0.605506 |
| **31** | mean\_Density | -0.359760 | 0.359760 |
| **38** | wtd\_range\_Density | 0.354665 | 0.354665 |
| **143** | Eu | -0.323738 | 0.323738 |
| **93** | Al | -0.299284 | 0.299284 |
| **121** | Nb | 0.289820 | 0.289820 |
| **117** | Rb | 0.262275 | 0.262275 |
| **78** | wtd\_range\_Valence | 0.238612 | 0.238612 |
| **84** | Be | -0.225422 | 0.225422 |
| **83** | Li | 0.224590 | 0.224590 |
| **87** | N | -0.215999 | 0.215999 |
| **137** | La | -0.215895 | 0.215895 |
| **114** | Se | -0.189324 | 0.189324 |
| **95** | P | -0.167144 | 0.167144 |
| **40** | wtd\_std\_Density | -0.165014 | 0.165014 |
| **151** | Lu | 0.163652 | 0.163652 |
| **86** | C | 0.159113 | 0.159113 |
| **144** | Gd | -0.152539 | 0.152539 |
| **107** | Co | -0.133933 | 0.133933 |
| **54** | wtd\_gmean\_FusionHeat | 0.133314 | 0.133314 |
| **108** | Ni | -0.124371 | 0.124371 |
| **35** | entropy\_Density | -0.121573 | 0.121573 |
| **103** | V | 0.121085 | 0.121085 |
| **2** | wtd\_mean\_atomic\_mass | -0.117541 | 0.117541 |
| **102** | Ti | -0.088072 | 0.088072 |
| **99** | K | 0.067738 | 0.067738 |
| **85** | B | -0.054637 | 0.054637 |
| **65** | entropy\_ThermalConductivity | 0.021905 | 0.021905 |
| **110** | Zn | -0.021174 | 0.021174 |
| **150** | Yb | 0.005876 | 0.005876 |
| **123** | Tc | 0.003704 | 0.003704 |

Recursive feature elimination (RFE) is a feature selection method that fits a model and removes the weakest feature (or features) until the specified number of features is reached

In [45]:

*# Instance of Recursive Feature Elimination*

rfe **=** RFE(estimator**=**lasso\_reg\_best, n\_features\_to\_select**=**10, step**=**1)

*# Fit RFE*

rfe**.**fit(X\_all\_scaled, y\_all)

Out[45]:

RFE(estimator=Lasso(alpha=0.3), n\_features\_to\_select=10)

In [46]:

*# get column names*

columns **=** pd**.**DataFrame(list(ind\_columns\_all))

*# Get the ranking of features*

ranking **=** pd**.**DataFrame(rfe**.**ranking\_)

*# ranking and their importance*

rfe\_selected **=** pd**.**DataFrame()

rfe\_selected **=** pd**.**concat([columns, ranking], axis**=**1)

rfe\_selected**.**columns **=** ['Feature Name', 'Ranking']

rfe\_selected\_sorted **=** rfe\_selected**.**sort\_values(by**=**['Ranking'], ascending**=True**)

print(rfe\_selected\_sorted)

Feature Name Ranking

163 Bi 1

62 wtd\_mean\_ThermalConductivity 1

33 gmean\_Density 1

64 wtd\_gmean\_ThermalConductivity 1

66 wtd\_entropy\_ThermalConductivity 1

.. ... ...

164 Po 154

165 At 155

166 Rn 156

1 mean\_atomic\_mass 157

0 number\_of\_elements 158

[167 rows x 2 columns]

In [47]:

*# Top 10 features*

df\_top\_10 **=** rfe\_selected\_sorted[(rfe\_selected\_sorted["Ranking"] **<** 11)]

df\_top\_10

Out[47]:

|  | **Feature Name** | **Ranking** |
| --- | --- | --- |
| **163** | Bi | 1 |
| **62** | wtd\_mean\_ThermalConductivity | 1 |
| **33** | gmean\_Density | 1 |
| **64** | wtd\_gmean\_ThermalConductivity | 1 |
| **66** | wtd\_entropy\_ThermalConductivity | 1 |
| **80** | wtd\_std\_Valence | 1 |
| **7** | range\_atomic\_mass | 1 |
| **136** | Ba | 1 |
| **10** | wtd\_std\_atomic\_mass | 1 |
| **46** | wtd\_entropy\_ElectronAffinity | 1 |
| **6** | wtd\_entropy\_atomic\_mass | 2 |
| **94** | Si | 3 |
| **44** | wtd\_gmean\_ElectronAffinity | 4 |
| **100** | Ca | 5 |
| **70** | wtd\_std\_ThermalConductivity | 6 |
| **11** | mean\_fie | 7 |
| **27** | range\_atomic\_radius | 8 |
| **60** | wtd\_std\_FusionHeat | 9 |
| **113** | As | 10 |

In [48]:

*# Visulization of important features*

ax **=** sns**.**barplot(x **=**'Ranking', y **=** 'Feature Name',data**=**df\_top\_10, orient**=** 'h')

ax**.**set\_title("Feature Importance")

ax**.**set\_xlabel("Ranking")

ax**.**set\_ylabel("Feature Names")

Out[48]:

Text(0, 0.5, 'Feature Names')

Chart, bar chart

Description automatically generated

#### Ridge Regression

#### Method - 1 (Dataset with all variables)

In [49]:

*# hyperparameter tuning with grid search*

ridge\_reg\_all **=** Ridge()

param\_grid **=** {'alpha': [0.001, 0.01, 0.1, 1, 10, 100, 200, 500, 1000]}

grid\_search\_ridge\_all **=** GridSearchCV(estimator**=**ridge\_reg\_all, param\_grid**=**param\_grid, n\_jobs**=-**1, cv**=**cv,

scoring**=**'neg\_root\_mean\_squared\_error')

In [50]:

**%%time**

grid\_result\_ridge\_all **=** grid\_search\_ridge\_all**.**fit(X\_all\_scaled, y\_all)

*# summarize results*

print("Best: %f using %s" **%** (grid\_result\_ridge\_all**.**best\_score\_, grid\_result\_ridge\_all**.**best\_params\_))

means\_all **=** grid\_result\_ridge\_all**.**cv\_results\_['mean\_test\_score']

stds\_all **=** grid\_result\_ridge\_all**.**cv\_results\_['std\_test\_score']

params\_all **=** grid\_result\_ridge\_all**.**cv\_results\_['params']

**for** mean, stdev, param **in** zip(means\_all, stds\_all, params\_all):

print("%f (%f) with: %r" **%** (mean, stdev, param))

Best: -17.897377 using {'alpha': 1000}

-22.567645 (12.428285) with: {'alpha': 0.001}

-22.565251 (12.421771) with: {'alpha': 0.01}

-22.542670 (12.360181) with: {'alpha': 0.1}

-22.391410 (11.947036) with: {'alpha': 1}

-21.836797 (10.594398) with: {'alpha': 10}

-19.851100 (6.203182) with: {'alpha': 100}

-19.006787 (4.275562) with: {'alpha': 200}

-18.166797 (2.089707) with: {'alpha': 500}

-17.897377 (1.037348) with: {'alpha': 1000}

Wall time: 3.34 s

#### Method - 2 (Reduced dataset)

In [51]:

*# hyperparameter tuning with grid search*

ridge\_reg **=** Ridge()

grid\_search\_ridge **=** GridSearchCV(estimator**=**ridge\_reg, param\_grid**=**param\_grid, n\_jobs**=-**1, cv**=**cv,

scoring**=**'neg\_root\_mean\_squared\_error')

In [52]:

**%%time**

grid\_result\_ridge **=** grid\_search\_ridge**.**fit(X\_scaled, y)

*# summarize results*

print("Best: %f using %s" **%** (grid\_result\_ridge**.**best\_score\_, grid\_result\_ridge**.**best\_params\_))

means **=** grid\_result\_ridge**.**cv\_results\_['mean\_test\_score']

stds **=** grid\_result\_ridge**.**cv\_results\_['std\_test\_score']

params **=** grid\_result\_ridge**.**cv\_results\_['params']

**for** mean, stdev, param **in** zip(means, stds, params):

print("%f (%f) with: %r" **%** (mean, stdev, param))

Best: -18.214993 using {'alpha': 1000}

-22.679788 (10.941132) with: {'alpha': 0.001}

-22.679199 (10.940006) with: {'alpha': 0.01}

-22.673323 (10.928771) with: {'alpha': 0.1}

-22.615849 (10.818889) with: {'alpha': 1}

-22.138575 (9.907089) with: {'alpha': 10}

-20.123965 (6.006532) with: {'alpha': 100}

-19.295693 (4.250282) with: {'alpha': 200}

-18.476290 (2.152504) with: {'alpha': 500}

-18.214993 (1.092863) with: {'alpha': 1000}

Wall time: 2.48 s

With respect to RMSE, Ridge regression model with all variables is better than reduced variables at alpha of 1000

In [53]:

ridge\_reg\_best **=** Ridge(alpha **=** 1000)

*# Lasso Regression Model*

ridge **=** ridge\_reg\_best**.**fit(X\_all\_scaled, y\_all)

*# score the regression model based on Root Mean Squared Error*

ridge\_preds **=** cross\_val\_predict(ridge, X\_all\_scaled, y\_all, cv**=**cv)

ridge\_scores **=** cross\_val\_score(ridge, X\_all\_scaled, y\_all, scoring**=**'neg\_root\_mean\_squared\_error', cv**=**cv)

print('Average of RMSE for 10 folds: ', np**.**mean(ridge\_scores))

print('Standard Deviation of RMSE for 10 folds: ', np**.**std(ridge\_scores))

Average of RMSE for 10 folds: -17.897376751849187

Standard Deviation of RMSE for 10 folds: 1.037348231646988

In [54]:

ridge\_scores

Out[54]:

array([-17.20146094, -17.76609165, -17.70643718, -17.55575516,

-17.52395924, -18.28908214, -17.33259556, -17.65596932,

-17.0831061 , -20.85931023])

In [55]:

rezdf\_ridge **=** df

rezdf\_ridge['preds'] **=** ridge\_preds

rezdf\_ridge['resids'] **=** (rezdf\_ridge['preds'] **-** y\_all)

*# residual plot*

fig **=** plt**.**figure(figsize **=** (12, 8))

sns**.**relplot(x**=**'preds', y**=**'resids', data**=**rezdf\_ridge, color**=**'red')

plt**.**title('Residual plot', size**=**20)

plt**.**xlim([**-**200, 200])

plt**.**ylim([**-**200, 200])

plt**.**xlabel('Predictions', size**=**14)

plt**.**ylabel('Residuals', size**=**14);

<Figure size 864x576 with 0 Axes>

Chart, scatter chart

Description automatically generated

In [56]:

*# Plot the histogram of the error terms*

fig **=** plt**.**figure()

sns**.**distplot(rezdf\_ridge['resids'], bins **=** 20)

fig**.**suptitle('Error Terms', fontsize **=** 20) *# Plot heading*

plt**.**xlabel('Errors', fontsize **=** 18)

plt**.**xlim([**-**200, 200])

Out[56]:

(-200.0, 200.0)

Histogram

Description automatically generated with low confidence

In [57]:

*# intercept and regression model coefficients*

coef2\_ridge **=** ridge**.**coef\_**.**tolist()

coefdf2\_ridge **=** pd**.**DataFrame(ind\_columns\_all, columns**=**['Variable'])

coefdf2\_ridge['Coefficient'] **=** coef2\_ridge

print('Intercept: \n', ridge**.**intercept\_)

coefdf2\_ridge

Intercept:

34.42121913535249

Out[57]:

|  | **Variable** | **Coefficient** |
| --- | --- | --- |
| **0** | number\_of\_elements | 1.662024 |
| **1** | mean\_atomic\_mass | 1.530949 |
| **2** | wtd\_mean\_atomic\_mass | -2.061641 |
| **3** | gmean\_atomic\_mass | 0.765496 |
| **4** | wtd\_gmean\_atomic\_mass | -0.383023 |
| **...** | ... | ... |
| **162** | Pb | 0.329784 |
| **163** | Bi | 3.752301 |
| **164** | Po | 0.000000 |
| **165** | At | 0.000000 |
| **166** | Rn | 0.000000 |

167 rows × 2 columns

Recursive feature elimination (RFE) is a feature selection method that fits a model and removes the weakest feature (or features) until the specified number of features is reached

In [58]:

*# Instance of Recursive Feature Elimination*

rfe **=** RFE(estimator**=**ridge\_reg\_best, n\_features\_to\_select**=**5, step**=**1)

*# Fit RFE*

rfe**.**fit(X\_all\_scaled, y\_all)

Out[58]:

RFE(estimator=Ridge(alpha=1000), n\_features\_to\_select=5)

In [59]:

*# get column names*

columns **=** pd**.**DataFrame(list(ind\_columns\_all))

*# Get the ranking of features*

ranking **=** pd**.**DataFrame(rfe**.**ranking\_)

*# ranking and their importance*

rfe\_selected **=** pd**.**DataFrame()

rfe\_selected **=** pd**.**concat([columns, ranking], axis**=**1)

rfe\_selected**.**columns **=** ['Feature Name', 'Ranking']

rfe\_selected\_sorted **=** rfe\_selected**.**sort\_values(by**=**['Ranking'], ascending**=True**)

print(rfe\_selected\_sorted)

Feature Name Ranking

136 Ba 1

62 wtd\_mean\_ThermalConductivity 1

64 wtd\_gmean\_ThermalConductivity 1

10 wtd\_std\_atomic\_mass 1

7 range\_atomic\_mass 1

.. ... ...

141 Pm 159

164 Po 160

165 At 161

82 He 162

166 Rn 163

[167 rows x 2 columns]

In [60]:

*# Top 5 features*

df\_top\_10 **=** rfe\_selected\_sorted[(rfe\_selected\_sorted["Ranking"] **<** 11)]

df\_top\_10

Out[60]:

|  | **Feature Name** | **Ranking** |
| --- | --- | --- |
| **136** | Ba | 1 |
| **62** | wtd\_mean\_ThermalConductivity | 1 |
| **64** | wtd\_gmean\_ThermalConductivity | 1 |
| **10** | wtd\_std\_atomic\_mass | 1 |
| **7** | range\_atomic\_mass | 1 |
| **70** | wtd\_std\_ThermalConductivity | 2 |
| **44** | wtd\_gmean\_ElectronAffinity | 3 |
| **0** | number\_of\_elements | 4 |
| **35** | entropy\_Density | 5 |
| **6** | wtd\_entropy\_atomic\_mass | 6 |
| **80** | wtd\_std\_Valence | 7 |
| **163** | Bi | 8 |
| **27** | range\_atomic\_radius | 9 |
| **66** | wtd\_entropy\_ThermalConductivity | 10 |

In [61]:

*# Visulization of important features*

ax **=** sns**.**barplot(x **=**'Ranking', y **=** 'Feature Name',data**=**df\_top\_10, orient**=** 'h')

ax**.**set\_title("Feature Importance")

ax**.**set\_xlabel("Ranking")

ax**.**set\_ylabel("Feature Names")

Out[61]:

Text(0, 0.5, 'Feature Names')

Chart, bar chart

Description automatically generated

#### ElasticNet Regression

#### Method - 1 (Dataset with all variables)

In [62]:

*# ElasticNet Regression Model*

elastic\_net\_all **=** linear\_model**.**ElasticNet()**.**fit(X\_all\_scaled, y\_all)

*# score the regression model based on Root Mean Squared Error*

elastic\_net\_preds\_all **=** cross\_val\_predict(elastic\_net\_all, X\_all\_scaled, y\_all, cv**=**cv)

elastic\_net\_scores\_all **=** cross\_val\_score(elastic\_net\_all, X\_all\_scaled, y\_all, scoring**=**'neg\_root\_mean\_squared\_error', cv**=**cv)

print('Average of RMSE for 10 folds: ', np**.**mean(elastic\_net\_scores\_all))

print('Standard Deviation of RMSE for 10 folds: ', np**.**std(elastic\_net\_scores\_all))

Average of RMSE for 10 folds: -19.463285824164597

Standard Deviation of RMSE for 10 folds: 0.2705539932896067

In [63]:

elastic\_net\_scores\_all

Out[63]:

array([-19.41353864, -19.85481209, -19.90531346, -19.51629704,

-19.32952465, -19.65027818, -19.10810117, -19.41211302,

-19.02217056, -19.42070943])

#### Method - 2 (Reduced dataset)

In [64]:

*# ElasticNet Regression Model*

elastic\_net **=** linear\_model**.**ElasticNet()**.**fit(X\_scaled, y)

*# score the regression model based on Root Mean Squared Error*

elastic\_net\_preds **=** cross\_val\_predict(elastic\_net, X\_scaled, y, cv**=**cv)

elastic\_net\_scores **=** cross\_val\_score(elastic\_net, X\_scaled, y, scoring**=**'neg\_root\_mean\_squared\_error', cv**=**cv)

print('Average of RMSE for 10 folds: ', np**.**mean(elastic\_net\_scores))

print('Standard Deviation of RMSE for 10 folds: ', np**.**std(elastic\_net\_scores))

Average of RMSE for 10 folds: -19.670809400897863

Standard Deviation of RMSE for 10 folds: 0.2943780608568159

In [65]:

elastic\_net\_scores

Out[65]:

array([-19.62085556, -20.04501339, -20.18080974, -19.76583607,

-19.55486832, -19.85011789, -19.28952289, -19.6022496 ,

-19.1628291 , -19.63599143])

With respect to RMSE, ElasticNet regression model with all variables is better than reduced variables

In [66]:

rezdf\_elastic\_net **=** df\_all

rezdf\_elastic\_net['preds'] **=** elastic\_net\_preds\_all

rezdf\_elastic\_net['resids'] **=** (rezdf\_elastic\_net['preds'] **-** y)

*# residual plot*

fig **=** plt**.**figure(figsize **=** (12, 8))

sns**.**relplot(x**=**'preds', y**=**'resids', data**=**rezdf\_elastic\_net, color**=**'red')

plt**.**title('Residual plot', size**=**20)

plt**.**xlabel('Predictions', size**=**14)

plt**.**ylabel('Residuals', size**=**14);

<Figure size 864x576 with 0 Axes>

Chart, scatter chart

Description automatically generated

In [67]:

*# Plot the histogram of the error terms*

fig **=** plt**.**figure()

sns**.**distplot(rezdf\_elastic\_net['resids'], bins **=** 20)

fig**.**suptitle('Error Terms', fontsize **=** 20) *# Plot heading*

plt**.**xlabel('Errors', fontsize **=** 18)

Out[67]:

Text(0.5, 0, 'Errors')

A picture containing histogram

Description automatically generated

In [68]:

*# intercept and regression model coefficients*

coef2\_elastic\_net **=** elastic\_net\_all**.**coef\_**.**tolist()

coefdf2\_elastic\_net **=** pd**.**DataFrame(ind\_columns\_all, columns**=**['Variable'])

coefdf2\_elastic\_net['Coefficient'] **=** coef2\_elastic\_net

print('Intercept: \n', elastic\_net**.**intercept\_)

Intercept:

34.42121913535249

In [69]:

pd**.**set\_option('display.max\_rows', 170)

print(coefdf2\_elastic\_net)

Variable Coefficient

0 number\_of\_elements 0.984486

1 mean\_atomic\_mass -0.000000

2 wtd\_mean\_atomic\_mass -0.000000

3 gmean\_atomic\_mass -0.000000

4 wtd\_gmean\_atomic\_mass -0.000000

5 entropy\_atomic\_mass 0.000000

6 wtd\_entropy\_atomic\_mass 1.588463

7 range\_atomic\_mass 1.154617

8 wtd\_range\_atomic\_mass -0.845391

9 std\_atomic\_mass 0.422909

10 wtd\_std\_atomic\_mass -0.000000

11 mean\_fie 0.289722

12 wtd\_mean\_fie 0.000000

13 gmean\_fie 0.092132

14 wtd\_gmean\_fie -0.000000

15 entropy\_fie 0.053338

16 wtd\_entropy\_fie 0.604340

17 range\_fie 0.933960

18 wtd\_range\_fie -0.000000

19 std\_fie 0.099647

20 wtd\_std\_fie 0.000000

21 mean\_atomic\_radius 0.000000

22 wtd\_mean\_atomic\_radius 0.364845

23 gmean\_atomic\_radius -0.000000

24 wtd\_gmean\_atomic\_radius 0.000000

25 entropy\_atomic\_radius 0.000000

26 wtd\_entropy\_atomic\_radius 0.633045

27 range\_atomic\_radius 1.432005

28 wtd\_range\_atomic\_radius -0.024386

29 std\_atomic\_radius 0.506948

30 wtd\_std\_atomic\_radius 0.396864

31 mean\_Density -0.434999

32 wtd\_mean\_Density -0.000000

33 gmean\_Density -0.823879

34 wtd\_gmean\_Density -0.000000

35 entropy\_Density 0.000000

36 wtd\_entropy\_Density 0.000000

37 range\_Density 0.000000

38 wtd\_range\_Density 0.000000

39 std\_Density -0.000000

40 wtd\_std\_Density -0.201977

41 mean\_ElectronAffinity -0.000000

42 wtd\_mean\_ElectronAffinity -0.000000

43 gmean\_ElectronAffinity -1.139041

44 wtd\_gmean\_ElectronAffinity -1.922188

45 entropy\_ElectronAffinity -0.000000

46 wtd\_entropy\_ElectronAffinity -0.417492

47 range\_ElectronAffinity 0.000000

48 wtd\_range\_ElectronAffinity -0.000000

49 std\_ElectronAffinity 0.236956

50 wtd\_std\_ElectronAffinity 0.120894

51 mean\_FusionHeat -0.000000

52 wtd\_mean\_FusionHeat -0.000000

53 gmean\_FusionHeat -0.000000

54 wtd\_gmean\_FusionHeat -0.000000

55 entropy\_FusionHeat 0.172367

56 wtd\_entropy\_FusionHeat 0.752248

57 range\_FusionHeat -0.173889

58 wtd\_range\_FusionHeat 0.000000

59 std\_FusionHeat -0.122728

60 wtd\_std\_FusionHeat -1.124035

61 mean\_ThermalConductivity 0.817599

62 wtd\_mean\_ThermalConductivity 2.496734

63 gmean\_ThermalConductivity -0.747381

64 wtd\_gmean\_ThermalConductivity -1.313422

65 entropy\_ThermalConductivity 0.000000

66 wtd\_entropy\_ThermalConductivity 0.000000

67 range\_ThermalConductivity 1.241433

68 wtd\_range\_ThermalConductivity 1.456685

69 std\_ThermalConductivity 0.902571

70 wtd\_std\_ThermalConductivity 2.543067

71 mean\_Valence -0.462457

72 wtd\_mean\_Valence -0.371290

73 gmean\_Valence -0.285867

74 wtd\_gmean\_Valence -0.131099

75 entropy\_Valence 0.155054

76 wtd\_entropy\_Valence 0.570728

77 range\_Valence -0.179742

78 wtd\_range\_Valence -0.000000

79 std\_Valence -0.628608

80 wtd\_std\_Valence -2.483201

81 H -0.000000

82 He 0.000000

83 Li 0.000000

84 Be -0.117178

85 B -0.000784

86 C 0.077238

87 N -0.019883

88 O 1.223456

89 F 0.000000

90 Ne 0.000000

91 Na -0.000000

92 Mg 0.000000

93 Al -0.257660

94 Si -1.296059

95 P -0.000000

96 S -0.196525

97 Cl -0.271938

98 Ar 0.000000

99 K 0.000000

100 Ca 2.362060

101 Sc -0.000000

102 Ti -0.000000

103 V 0.000000

104 Cr 0.000000

105 Mn 0.000000

106 Fe -0.030994

107 Co -0.003608

108 Ni -0.132154

109 Cu 0.967124

110 Zn -0.000000

111 Ga 0.000000

112 Ge -0.175748

113 As -0.512833

114 Se -0.000000

115 Br -0.000000

116 Kr 0.000000

117 Rb 0.000000

118 Sr 0.520270

119 Y 0.350626

120 Zr -0.000000

121 Nb 0.000000

122 Mo -0.000000

123 Tc 0.000000

124 Ru -0.000000

125 Rh -0.000000

126 Pd -0.000000

127 Ag -0.465381

128 Cd -0.000000

129 In -0.000000

130 Sn -0.000000

131 Sb -0.000000

132 Te -0.000000

133 I 0.000000

134 Xe 0.000000

135 Cs 0.000000

136 Ba 4.682539

137 La -0.363840

138 Ce -0.924071

139 Pr -0.000000

140 Nd -0.685764

141 Pm 0.000000

142 Sm -0.000000

143 Eu -0.192075

144 Gd -0.000000

145 Tb -0.000000

146 Dy -0.000000

147 Ho -0.000000

148 Er 0.000000

149 Tm -0.000000

150 Yb 0.000000

151 Lu -0.000000

152 Hf -0.000000

153 Ta -0.000000

154 W -0.000000

155 Re 0.000000

156 Os -0.000000

157 Ir -0.000000

158 Pt -0.000000

159 Au -0.000000

160 Hg 1.511780

161 Tl 1.093075

162 Pb 0.000000

163 Bi 1.695102

164 Po 0.000000

165 At 0.000000

166 Rn 0.000000

Recursive feature elimination (RFE) is a feature selection method that fits a model and removes the weakest feature (or features) until the specified number of features is reached.

In [70]:

*# Instance of Recursive Feature Elimination*

rfe **=** RFE(estimator**=**elastic\_net\_all, n\_features\_to\_select**=**10, step**=**1)

*# Fit RFE*

rfe**.**fit(X\_all\_scaled, y\_all)

Out[70]:

RFE(estimator=ElasticNet(), n\_features\_to\_select=10)

In [71]:

*# get column names*

columns **=** pd**.**DataFrame(list(ind\_columns\_all))

*# Get the ranking of features*

ranking **=** pd**.**DataFrame(rfe**.**ranking\_)

*# ranking and their importance*

rfe\_selected **=** pd**.**DataFrame()

rfe\_selected **=** pd**.**concat([columns, ranking], axis**=**1)

rfe\_selected**.**columns **=** ['Feature Name', 'Ranking']

rfe\_selected\_sorted **=** rfe\_selected**.**sort\_values(by**=**['Ranking'], ascending**=True**)

print(rfe\_selected\_sorted)

Feature Name Ranking

163 Bi 1

17 range\_fie 1

44 wtd\_gmean\_ElectronAffinity 1

67 range\_ThermalConductivity 1

70 wtd\_std\_ThermalConductivity 1

80 wtd\_std\_Valence 1

27 range\_atomic\_radius 1

100 Ca 1

136 Ba 1

6 wtd\_entropy\_atomic\_mass 1

62 wtd\_mean\_ThermalConductivity 2

88 O 3

64 wtd\_gmean\_ThermalConductivity 4

56 wtd\_entropy\_FusionHeat 5

68 wtd\_range\_ThermalConductivity 6

160 Hg 7

33 gmean\_Density 8

43 gmean\_ElectronAffinity 9

7 range\_atomic\_mass 10

94 Si 11

69 std\_ThermalConductivity 12

0 number\_of\_elements 13

109 Cu 14

161 Tl 15

138 Ce 16

60 wtd\_std\_FusionHeat 17

8 wtd\_range\_atomic\_mass 18

76 wtd\_entropy\_Valence 19

71 mean\_Valence 20

63 gmean\_ThermalConductivity 21

61 mean\_ThermalConductivity 22

26 wtd\_entropy\_atomic\_radius 23

140 Nd 24

79 std\_Valence 25

29 std\_atomic\_radius 26

16 wtd\_entropy\_fie 27

118 Sr 28

113 As 29

30 wtd\_std\_atomic\_radius 30

72 wtd\_mean\_Valence 31

127 Ag 32

31 mean\_Density 33

137 La 34

46 wtd\_entropy\_ElectronAffinity 35

9 std\_atomic\_mass 36

119 Y 37

11 mean\_fie 38

22 wtd\_mean\_atomic\_radius 39

73 gmean\_Valence 40

93 Al 41

49 std\_ElectronAffinity 42

97 Cl 43

57 range\_FusionHeat 44

96 S 45

40 wtd\_std\_Density 46

143 Eu 47

55 entropy\_FusionHeat 48

112 Ge 49

77 range\_Valence 50

75 entropy\_Valence 51

108 Ni 52

74 wtd\_gmean\_Valence 53

50 wtd\_std\_ElectronAffinity 54

84 Be 55

59 std\_FusionHeat 56

19 std\_fie 57

13 gmean\_fie 58

86 C 59

15 entropy\_fie 60

106 Fe 61

28 wtd\_range\_atomic\_radius 62

87 N 63

107 Co 64

85 B 65

159 Au 66

99 K 67

5 entropy\_atomic\_mass 68

4 wtd\_gmean\_atomic\_mass 69

12 wtd\_mean\_fie 70

3 gmean\_atomic\_mass 71

2 wtd\_mean\_atomic\_mass 72

34 wtd\_gmean\_Density 73

1 mean\_atomic\_mass 74

111 Ga 75

123 Tc 76

10 wtd\_std\_atomic\_mass 77

158 Pt 78

32 wtd\_mean\_Density 79

130 Sn 80

14 wtd\_gmean\_fie 81

134 Xe 82

135 Cs 83

103 V 84

110 Zn 85

139 Pr 86

157 Ir 87

20 wtd\_std\_fie 88

91 Na 89

21 mean\_atomic\_radius 90

95 P 91

102 Ti 92

25 entropy\_atomic\_radius 93

23 gmean\_atomic\_radius 94

122 Mo 95

58 wtd\_range\_FusionHeat 96

41 mean\_ElectronAffinity 97

42 wtd\_mean\_ElectronAffinity 98

18 wtd\_range\_fie 99

47 range\_ElectronAffinity 100

92 Mg 101

117 Rb 102

24 wtd\_gmean\_atomic\_radius 103

48 wtd\_range\_ElectronAffinity 104

162 Pb 105

51 mean\_FusionHeat 106

101 Sc 107

98 Ar 108

131 Sb 109

35 entropy\_Density 110

105 Mn 111

36 wtd\_entropy\_Density 112

104 Cr 113

37 range\_Density 114

90 Ne 115

38 wtd\_range\_Density 116

89 F 117

39 std\_Density 118

45 entropy\_ElectronAffinity 119

52 wtd\_mean\_FusionHeat 120

114 Se 121

53 gmean\_FusionHeat 122

115 Br 123

54 wtd\_gmean\_FusionHeat 124

116 Kr 125

120 Zr 126

121 Nb 127

154 W 128

155 Re 129

124 Ru 130

125 Rh 131

126 Pd 132

153 Ta 133

128 Cd 134

129 In 135

144 Gd 136

152 Hf 137

132 Te 138

133 I 139

145 Tb 140

151 Lu 141

146 Dy 142

65 entropy\_ThermalConductivity 143

147 Ho 144

66 wtd\_entropy\_ThermalConductivity 145

148 Er 146

150 Yb 147

149 Tm 148

78 wtd\_range\_Valence 149

141 Pm 150

142 Sm 151

156 Os 152

164 Po 153

165 At 154

81 H 155

166 Rn 156

82 He 157

83 Li 158

In [72]:

df\_top\_10 **=** rfe\_selected\_sorted[(rfe\_selected\_sorted["Ranking"] **<** 11)]

df\_top\_10

Out[72]:

|  | **Feature Name** | **Ranking** |
| --- | --- | --- |
| **163** | Bi | 1 |
| **17** | range\_fie | 1 |
| **44** | wtd\_gmean\_ElectronAffinity | 1 |
| **67** | range\_ThermalConductivity | 1 |
| **70** | wtd\_std\_ThermalConductivity | 1 |
| **80** | wtd\_std\_Valence | 1 |
| **27** | range\_atomic\_radius | 1 |
| **100** | Ca | 1 |
| **136** | Ba | 1 |
| **6** | wtd\_entropy\_atomic\_mass | 1 |
| **62** | wtd\_mean\_ThermalConductivity | 2 |
| **88** | O | 3 |
| **64** | wtd\_gmean\_ThermalConductivity | 4 |
| **56** | wtd\_entropy\_FusionHeat | 5 |
| **68** | wtd\_range\_ThermalConductivity | 6 |
| **160** | Hg | 7 |
| **33** | gmean\_Density | 8 |
| **43** | gmean\_ElectronAffinity | 9 |
| **7** | range\_atomic\_mass | 10 |

In [73]:

*# Visulization of important features*

ax **=** sns**.**barplot(x **=**'Ranking', y **=** 'Feature Name',data**=**df\_top\_10, orient**=** 'h')

ax**.**set\_title("Feature Importance")

ax**.**set\_xlabel("Ranking")

ax**.**set\_ylabel("Feature Names")

Out[73]:

Text(0, 0.5, 'Feature Names')

Chart, bar chart

Description automatically generated

In [ ]: