

# PREDICTION OF CRITICAL TEMPERATURE FOR SUPERCONDUCTIVITY USING REGRESSION

Submitted by

Shivam Thakur

Nupur Bhavesh Shah

Pengxiang Zhuo

Rajaraghunathan Palanisamy

Submitted To

Prof. Ramin Mohammadi

## **Introduction**

Superconducting materials - materials that conduct current with zero resistance - have significant practical applications. Perhaps the best-known application is in the Magnetic Resonance Imaging (MRI) systems widely employed by health care professionals for detailed internal body imaging. Other prominent applications include the superconducting coils used to maintain high magnetic fields in the Large Hadron Collider at CERN, where the existence of Higgs Boson was recently confirmed, and the extremely sensitive magnetic field measuring devices called SQUIDs (Superconducting Quantum Interference Devices). Furthermore, superconductors could revolutionize the energy industry as frictionless (zero resistance) superconducting wires and electrical system may transport and deliver electricity with no energy loss.

In this project we have a statistical model for predicting the superconducting critical temperature on the bases of the features extracted from the superconductor's chemical formula. The model gives reasonable out-of-sample predictions:  $\pm 9.5$  K based on root-mean-squared error.

The features are extracted on the bases of thermal conductivity, atomic radius, valence, electron affinity, and atomic mass contribute the most to the model's predictive accuracy. This is crucial part of the model that it does not predict whether a material is a superconductor or not; but it can only give predictions for superconductors.

## Data Description

The superconductivity data set contains 21263 rows and 82 columns, which included 81 features variable and 1 target variable “Critical temperature”. The goal of this project is to predict the critical temperature of the superconductor by using their relevant features.

Among all the 81 features variable, that are some variables that are foundation of this data set and used to creating the features to predict the critical temperature, which shown on the table below, and rest of the variables are calculation of Mean, Weighted mean, Geometric mean, weighted geometric mean, Entropy, Weighted entropy, Range, Weighted range, Standard deviation, Weighted standard deviation respectively.

Variable	Units	Description
Atomic Mass	atomic mass units (AMU)	total proton and neutron rest masses
First Ionization Energy	kilo-Joules per mole (kJ/mol)	energy required to remove a valence electron
Atomic Radius	picometer (pm)	calculated atomic radius
Density	kilograms per meters cubed (kg/m <sup>3</sup> )	density at standard temperature and pressure
Electron Affinity	kilo-Joules per mole (kJ/mol)	energy required to add an electron to a neutral atom
Fusion Heat	kilo-Joules per mole (kJ/mol)	energy to change from solid to liquid without temperature change
Thermal Conductivity	watts per meter-Kelvin (W/(m × K))	thermal conductivity coefficient $\kappa$
Valence	no units	typical number of chemical bonds formed by the element

Feature & Description	Formula	Sample Value
Mean	$= \mu = (t_1 + t_2)/2$	35.5
Weighted mean	$= \nu = (p_1 t_1) + (p_2 t_2)$	44.43
Geometric mean	$= (t_1 t_2)^{1/2}$	33.23
Weighted geometric mean	$= (t_1)^{p_1} (t_2)^{p_2}$	43.21
Entropy	$= -w_1 \ln(w_1) - w_2 \ln(w_2)$	0.63
Weighted entropy	$= -A \ln(A) - B \ln(B)$	0.26
Range	$= t_1 - t_2 \ (t_1 > t_2)$	25
Weighted range	$= p_1 t_1 - p_2 t_2$	37.86
Standard deviation	$= [(1/2)((t_1 - \mu)^2 + (t_2 - \mu)^2)]^{1/2}$	12.5
Weighted standard deviation	$= [p_1(t_1 - \nu)^2 + p_2(t_2 - \nu)^2]^{1/2}$	8.75

## Data Exploration and Cleaning

Data cleaning and exploration involves dealing with raw data and converting it into a form which is easy to access and analyze.

```
df = pd.read_csv("train.csv")
dataset = pd.read_csv('train.csv')
df.head()
```

	number_of_elements	mean_atomic_mass	wtd_mean_atomic_mass	gmean_atomic_mass	wtd_gmean_atomic_mass	entropy_atomic_mass	wtd_entropy_atomic_mass
0	4	88.944468	57.862692	66.361592	36.116612	1.181795	1.062396
1	5	92.729214	58.518416	73.132787	36.396602	1.449309	1.057755
2	4	88.944468	57.885242	66.361592	36.122509	1.181795	0.975980
3	4	88.944468	57.873967	66.361592	36.119560	1.181795	1.022291
4	4	88.944468	57.840143	66.361592	36.110716	1.181795	1.129224

5 rows x 82 columns

Initially we first load the dataset and then obtain the first five rows in order to get an overview of what the columns are and their respective datatypes.

We then find the information to display the number of columns, column labels, column data types, memory usage, range index, and the number of cells in each column (non-null values)

```
df.info()
```

<class 'pandas.core.frame.DataFrame'>			
RangeIndex: 21263 entries, 0 to 21262			
Data columns (total 82 columns):			
number_of_elements	21263	non-null	int64
mean_atomic_mass	21263	non-null	float64
wtd_mean_atomic_mass	21263	non-null	float64
gmean_atomic_mass	21263	non-null	float64
wtd_gmean_atomic_mass	21263	non-null	float64
entropy_atomic_mass	21263	non-null	float64
wtd_entropy_atomic_mass	21263	non-null	float64
range_atomic_mass	21263	non-null	float64
wtd_range_atomic_mass	21263	non-null	float64
std_atomic_mass	21263	non-null	float64
wtd_std_atomic_mass	21263	non-null	float64
mean_fie	21263	non-null	float64
wtd_mean_fie	21263	non-null	float64
gmean_fie	21263	non-null	float64
wtd_gmean_fie	21263	non-null	float64
entropy_fie	21263	non-null	float64
wtd_entropy_fie	21263	non-null	float64
range_fie	21263	non-null	float64
wtd_range_fie	21263	non-null	float64
std_fie	21263	non-null	float64
wtd_std_fie	21263	non-null	float64
mean_atomic_radius	21263	non-null	float64
wtd_mean_atomic_radius	21263	non-null	float64
gmean_atomic_radius	21263	non-null	float64
wtd_gmean_atomic_radius	21263	non-null	float64
entropy_atomic_radius	21263	non-null	float64
wtd_entropy_atomic_radius	21263	non-null	float64
range_atomic_radius	21263	non-null	int64
wtd_range_atomic_radius	21263	non-null	float64

mean_Valence	21263	non-null	float64
wtd_mean_Valence	21263	non-null	float64
gmean_Valence	21263	non-null	float64
wtd_gmean_Valence	21263	non-null	float64
entropy_Valence	21263	non-null	float64
wtd_entropy_Valence	21263	non-null	float64
range_Valence	21263	non-null	int64
wtd_range_Valence	21263	non-null	float64
std_Valence	21263	non-null	float64
wtd_std_Valence	21263	non-null	float64
critical_temp	21263	non-null	float64

dtypes: float64(79), int64(3)  
memory usage: 13.3 MB

## Data Exploration and Visualization

Exploring data and displaying it in a visual form is an important tool to help tell us a story, making it easy to understand by highlighting trends and outliers. Removing excess noise, gives us a clear picture and helps enable us to draw coherent conclusions about the data.

To understand the unique features in the model we get the results column wise to understand the nature of the model.

```
df.nunique()
```

number_of_elements	9
mean_atomic_mass	3365
wtd_mean_atomic_mass	15164
gmean_atomic_mass	3365
wtd_gmean_atomic_mass	15165
...	
range_Valence	7
wtd_range_Valence	5908
std_Valence	125
wtd_std_Valence	7082
critical_temp	3007
Length: 82, dtype: int64	

We obtain the details of the dataset and we can see that there are extreme values between in the Atomic Mass column : 6 and 21263.

```
df.describe()
```


	number_of_elements	mean_atomic_mass	wtd_mean_atomic_mass	gmean_atomic_mass	wtd_gmean_atomic_mass	entropy_atomic_mass	wtd_entropy_atomic_mass
count	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
std	1.439295	29.676497	33.490406	31.030272	36.651067	0.364930	0.401423
min	1.000000	6.941000	6.423452	5.320573	1.960849	0.000000	0.000000
25%	3.000000	72.458076	52.143839	58.041225	35.248990	0.966676	0.775363
50%	4.000000	84.922750	60.696571	66.361592	39.918385	1.199541	1.146783
75%	5.000000	100.404410	86.103540	78.116681	73.113234	1.444537	1.359418
max	9.000000	208.980400	208.980400	208.980400	208.980400	1.983797	1.958203


8 rows x 8 columns

Page Break

Since we aim to clean the data so that it becomes much easier to analyze we need to make sure that there are no null values as that would cause poor reporting and skewed analysis.

We could check the missing in the dataset by the following method and it shows that there are no null values in this dataset. In this case since it is a 0 or 1 i.e., hence we do not need to make any data changes to this dataset.

 `df.isnull().sum()`

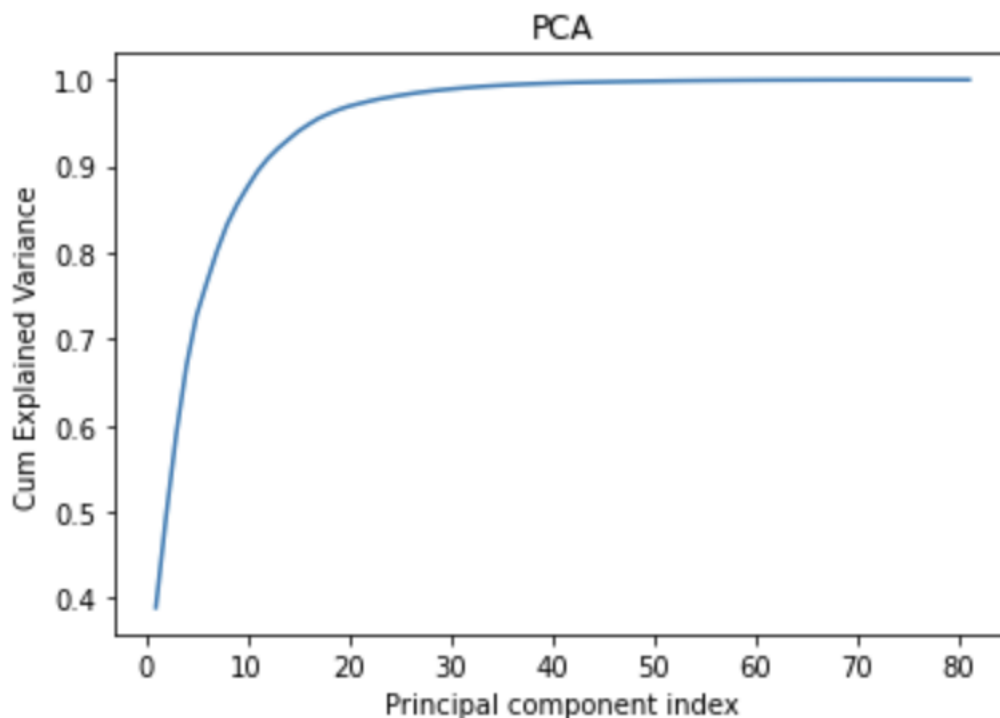


number_of_elements	0
mean_atomic_mass	0
wtd_mean_atomic_mass	0
gmean_atomic_mass	0
wtd_gmean_atomic_mass	0
..	
range_Valence	0
wtd_range_Valence	0
std_Valence	0
wtd_std_Valence	0
critical_temp	0
Length: 82, dtype: int64	

## Dimension reduction

Before we apply any learning algorithms to our dataset, we are going to perform the principal component analysis to reduce the computational complexity to our dataset, especially for the random forest tree and decision tree we will implement later which involved a lot of computation.

We first standardized the dataset, performed the algorithm, found the dimensions with the greatest variation to maximize the distinction between different data points. We calculated the cumulative explained variance and visualized it by the line chart in order to get a better sense of how many components to choose from.



What we see from the chart above is that the first few components captured a great amount of variation, and gradually decreased. The goal is to reduce the dimensions and not lose too much variance. Thinking of tradeoff between these two aspects, we decided to keep principal component as 15 which captured 94% of the cumulative explained variance.

Now After PCA was performed and we had identified our important 15 features, we decided to perform different kinds of regressions and compare the result.



## Linear Regression

The first method we are going to implement is linear regression, one of the most used algorithms and we serve it as the baseline model to compare the performance with other methods we choose. We apply all possible techniques that we covered from class and set the learning with 0.00001, and tolerance with 0.00005 for gradient descent, L1 & L2 regularization, also the normal equation as well to check if there is any improvement.

Gradient descent:

**Mean squared error(MSE): 477.75834388256084**  
**Root-mean-square error(RMSE): 21.857683863633877**

Gradient descent with L1 Regularization:

**Mean squared error(MSE): 477.75879836224505**  
**Root-mean-square error(RMSE): 21.85769425996816**

Gradient descent with L2 Regularization:

**Mean squared error(MSE): 477.75879836224505**  
**Root-mean-square error(RMSE): 21.85769425996816**

Normal Equation:

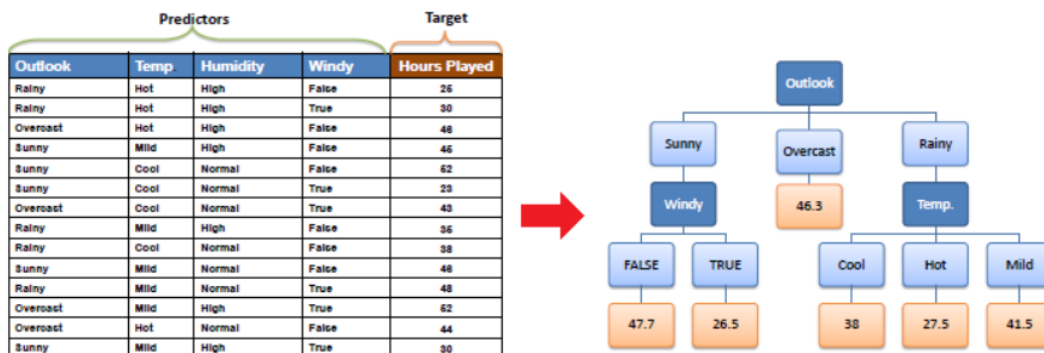
**Mean squared error(MSE): 477.69162513157426**  
**Root-mean-square error(RMSE): 21.856157602185576**

All the results are very similar to each other, but Normal equation perform the best, with lowest **MSE 477.6916** and **RMSE 21.8561** respectively.

## Decision Trees

As we saw above that the Linear Regression does a fairly nice job in minimizing the MSE of the predicted results, we still wanted to experiment with methods like Decision trees and Random Forest regression methods to see if there are any further improvements in the prediction.

Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.



In general, decision tree is a more flexible method compared to Linear Regression and it should be able to reduce the bias considerably by making improved predictions. Below are the error stats calculated after performing the Decision trees regression. The parameter values were: max\_depth – 10, min\_samples\_split – 1000.

**MSE 342.4764 & RMSE 1850612**

It is a slight improvement over the Linear Regression model.

Max\_depth – Maximum number of depth or branch nodes the root node can have

Min\_samples\_split – Split function will be applied if the node size(no of observations) is greater than or equal to this value

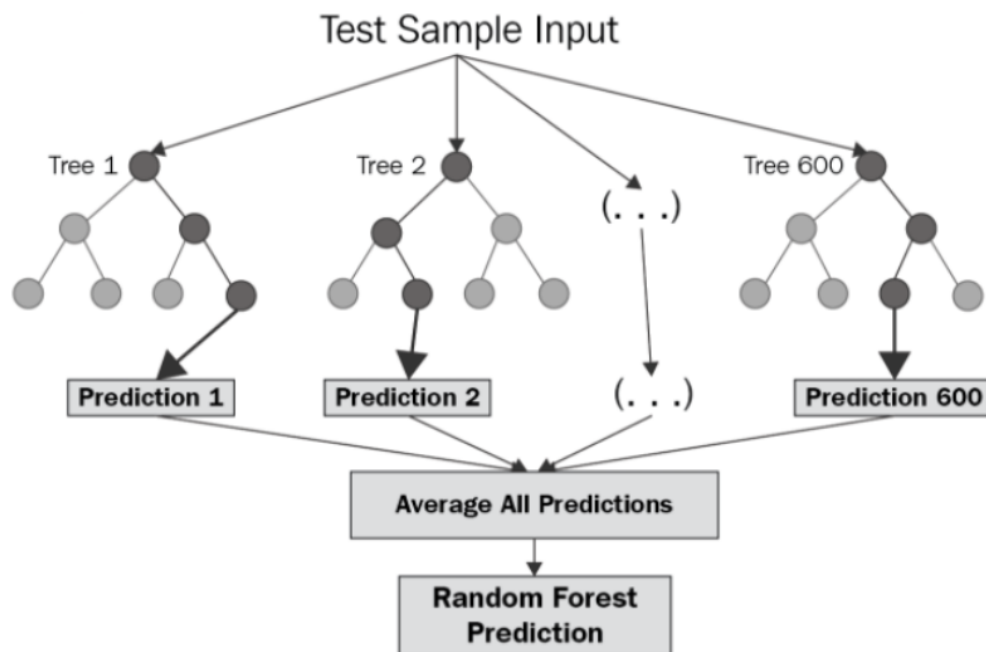
When we used decision trees to print the tree, it was as follows:

---

```
Root
| MSE of the node: 1177.15
| Count of observations in node: 17010
| Prediction of node: 34.508
|----- Split rule: PC1 <= 0.783
|       | MSE of the node: 284.55
|       | Count of observations in node: 10033
|       | Prediction of node: 13.225
|----- Split rule: PC1 <= 0.357
|       | MSE of the node: 103.72
|       | Count of observations in node: 8093
|       | Prediction of node: 8.83
|----- Split rule: PC1 <= -0.599
|       | MSE of the node: 72.34
|       | Count of observations in node: 5723
|       | Prediction of node: 7.102
|----- Split rule: PC1 > -0.599
|       | MSE of the node: 154.85
|       | Count of observations in node: 2370
|       | Prediction of node: 13.004
|----- Split rule: PC1 > 0.357
|       | MSE of the node: 622.31
|       | Count of observations in node: 1940
|       | Prediction of node: 31.557
|----- Split rule: PC1 > 0.783
|       | MSE of the node: 872.71
|       | Count of observations in node: 6977
|       | Prediction of node: 65.112
|----- Split rule: PC1 <= 1.136
|       | MSE of the node: 813.13
|       | Count of observations in node: 5553
|       | Prediction of node: 61.922
|----- Split rule: PC2 <= -0.427
|       | MSE of the node: 468.89
|       | Count of observations in node: 1672
|       | Prediction of node: 74.363
|----- Split rule: PC2 > -0.427
|       | MSE of the node: 866.02
|       | Count of observations in node: 3881
|       | Prediction of node: 56.562
|----- Split rule: PC1 > 1.136
|       | MSE of the node: 910.53
|       | Count of observations in node: 1424
|       | Prediction of node: 77.555
```

## Random Forest Regression

After Observing the much better results using Decision Tree Regression, we decided to try Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model. the more complicated, yet better ensemble approach knows as Random Forest Regression.

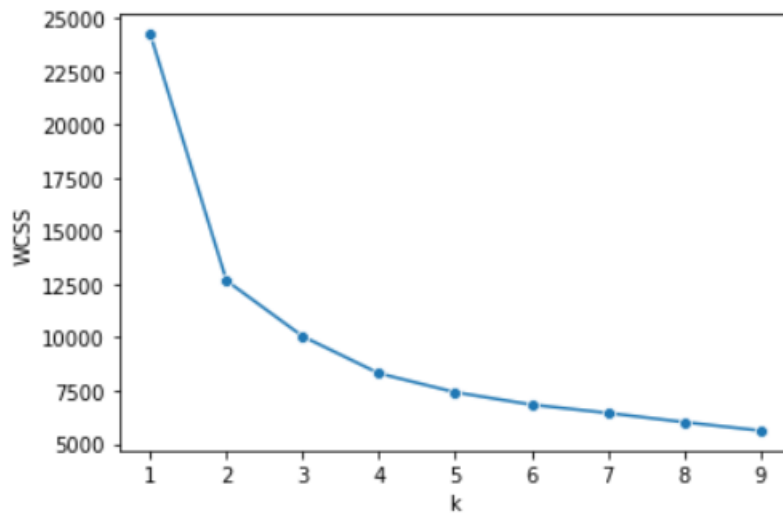


Here we set the number of trees as 2 and the seed value to 42 to perform then ran the Model.

The random Forest Regression after training was used to predict against the test model and it was able to do so successfully with a MSE of 177.23.

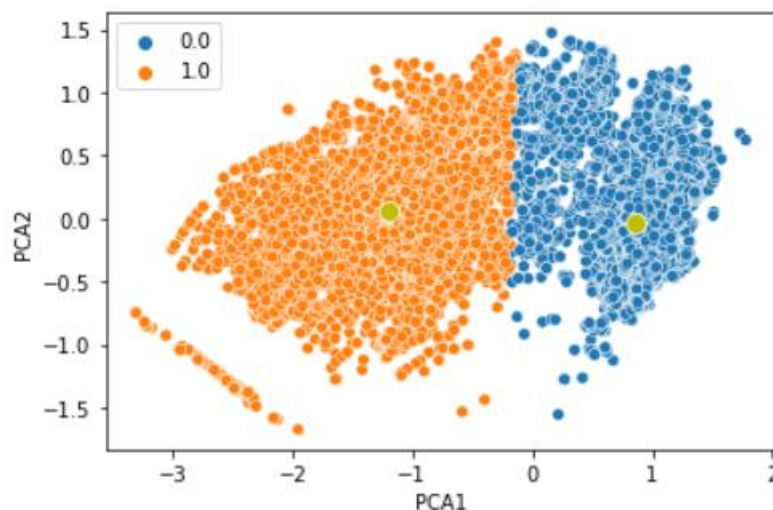
## K-means clustering

Though the main problem here is to predict a response variable, we were just curious to see if there are any clusters found in the data, and wanted to study the K-means algorithm by coding it from scratch. Using the elbow method, we plotted the below chart to find the optimal k-value for the clusters.



From the above chart, we can see that an elbow is created at the k-value 2. Hence we have chosen k-value as for creating the clusters.

Then the k-value is fed to the clustering algorithm to produce the clusters by minimizing the within cluster variation(distance) and maximizing the inter-cluster distances for each observation. After clustering the data points, the below chart is created in-order to visualize the clusters.



[**Note:** In order to visualize the clusters, we applied PCA to reduce the dimensions to 2 from 81].

## **Conclusion**

In conclusion, we had a dataset which had 82 features which we thought were used in determining the critical temperature of the superconductor. After the initial data exploration and going through the correlations, we realised that not all features are used and hence we can use the Dimension Reduction Methods to decrease the number of features used to prediction.

After applying the PCA Technique, we came up with 15 features which were having more impact in determining the actual temperature. Now we started with various regression models to perform a comparative analysis which one will be working best.

After training the algorithms, we tested them against the test data which we had split out from the original data.

The average value for MSE for all Linear Regression Models was 477.758

The average value of MSE for Decision Tree Regression Model was 342.47

The average value for MSE for Random Forest Regression Model was 177.23

Which gives enough evidence that the ensemble Random Forest Regression Model was able to determine the temperature with least error.