Superconductors Temperature Prediction

Kris Ghimire, Rajesh Satluri, Suchismita Moharana

May 9, 2021

Abstract

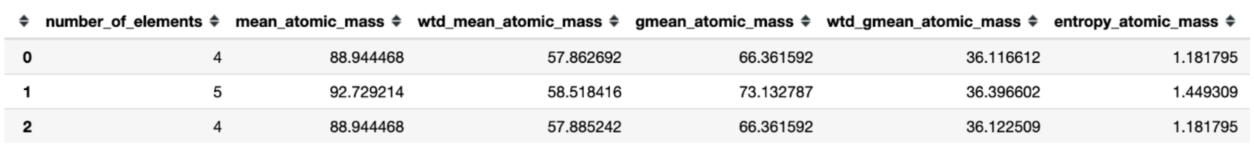
In this case study, researchers are going to build a simple regression model to predict temperature that makes new superconductors become superconductors. Superconductors are materials that give little or no resistance to electrical current. So, this is of pretty big importance to the scientific community.

1 Introduction:

Superconductor is a material that has the capability to conduct electricity or transport electron from one atom to another with no resistance. It means that superconductor has the power to carry a current indefinitely without making any loss of energy in other forms like heat, sounds etc. One of the most unusual properties of superconductor is that material must be in an extremely low temperature. Some of the most recent studies have found that critical low temperature is not so necessary to make superconductor. In either case appropriate temperature is crucial for producing superconductor. In this case study, researcher is going to implement simple machine learning model called Linear Regression (LR) to predict the approximate temperature that is required to make superconductor. Provided dataset has a size of 169 features and 21263 samples.

2 Methods

Dataset: There were two separated datasets. There are no missing values in the datasets but there are few columns with only zeros. We have joined them to merge all variables into single dataset and removed feature columns from the joined dataset with just the 0's as instances as they don't provide any values in building the model. Simple statistics performed on critical temperature shows that average critical temperature required to make superconductor is 34.42. The min and max critical temperature were 0.0002 and 185.0, respectively.



	number_of_elements	mean_atomic_mass	wtd_mean_atomic_mass	gmean_atomic_mass	wtd_gmean_atomic_mass	entropy_atomic_mass
0	4	88.944468	57.862692	66.361592	36.116612	1.181795
1	5	92.729214	58.518416	73.132787	36.396602	1.449309
2	4	88.944468	57.885242	66.361592	36.122509	1.181795

Table 1

Dataset contains mostly the numeric continuous features. The plot below (figure 1) shows that the dataset doesn't have any categorical data.

Train Test Split: Dataset was split into training and test set (80/20 split) to keep test data separate and to perform modeling and tuning only on training set.

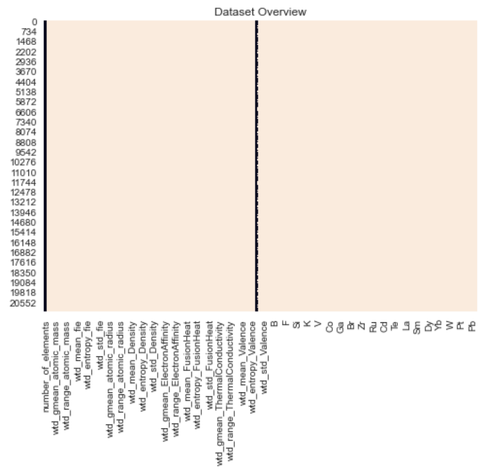
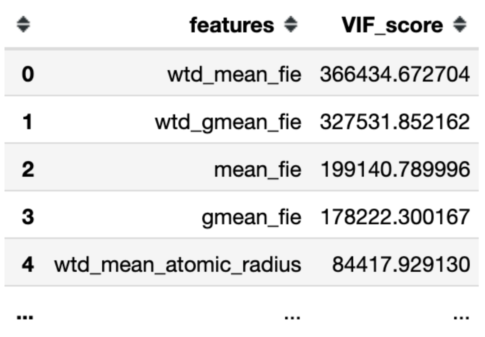


Figure 1

Multicollinearity: One of the key assumptions of linear model is to make sure that features are not highly correlated. High correlation between features can be a problem while explaining which features contributes most to the target feature and to the validity of model in general. We plotted correlation heat map to find all correlated features. Going through each correlated feature and picking one among two is a time-consuming manual process. We tried correlation heatmap. However, since the features are many correlation heatmap is not showing all at once properly. Hence, we used VIF (figure 2) but it is associated with a risk of losing important information. Also, given that none of us are SME in the field of superconductor, we do not know which features to keep and which to discard. Hence, we are using L1 or LASSO regularization to get the most important features.



	features	VIF_score
0	wtd_mean_fie	366434.672704
1	wtd_gmean_fie	327531.852162
2	mean_fie	199140.789996
3	gmean_fie	178222.300167
4	wtd_mean_atomic_radius	84417.929130
...	...	...

Table 2

L1 or LASSO regularization: LASSO stands for Least Absolute Shrinkage and Selection Operator. For L1 regularization, we use a penalty term where the function of the penalty is just the absolute value of the coefficients as it is shown in figure formula below:

$$\lambda \sum_{j=0}^{k} |m_j|$$

$$J = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \sum_{j=0}^{k} m_j x_{ij})^2 + \lambda \sum_{j=0}^{k} |m_j|$$

In the loss function formula above with the L1 penalty on the right-hand side, we can see that the penalty term starts to inhibit the growth of the slopes (coefficients). On the left-hand side, we have our traditional term for the loss function with our target features and our prediction. Initially, if we assume that all the slopes are 0 i.e. m=0, we can see that model will have a large error. Because the error will be basically the sum of all the target. Similarly, on the right-hand side of the equation, if all m=0, everything will be 0 this contributes nothing. Hence, our initial loss is simply the sum of all the targets. But as our slopes start to grow, in other words, we start to make more accurate predictions. We will have two competing terms, one that is decreasing as our slopes grow and the other that is increasing as our slopes grow.

So, the balance between these two terms will help find a model that is a general fit and not a specific fit to our data.

L1 regularization introduces sparsity to coefficients. Some of those low weight slopes or coefficients are forced to 0 giving us the most important features.

Scaling: Another key assumption is that all independent features should be normalized or scaled. As algorithm makes decision based on distance metrics, feature scaling is very important. We used *StandardScaler* to scale all the independent features. It is not a good practice to scale categorical data and our dataset does not have any categorical features. However, there are some features with many zeros and some values here and there. For example, feature 'Ca' has max values 24 while there are also many 0s, similarly feature 'Mn' has max value 14 and many 0 values.

Mathematically scaling is done using formula below:

$$z = \frac{(x-u)}{s}$$

Subtract the mean (called centering) and divide by the standard deviation to shift the distribution to have a mean of zero and a standard deviation of one.

Just to verify the importance of scaling, initially we used linear model on the given dataset without scaling. And we used linear model with scaled data. We saw some change in feature importance when we fit our model using scaled data.

We will use the scaled dataset and features that we get from L1 regularization to fit our final model using Ridge or L2 regularization.

Method of choosing Alpha for L1: We selected 100 evenly spaced numbers from an interval of 1 and 0 and we used this array as alpha in LassoCV function to get the best alpha value. Here we got best alpha value as 0.23232 as shown in the (figure 3)

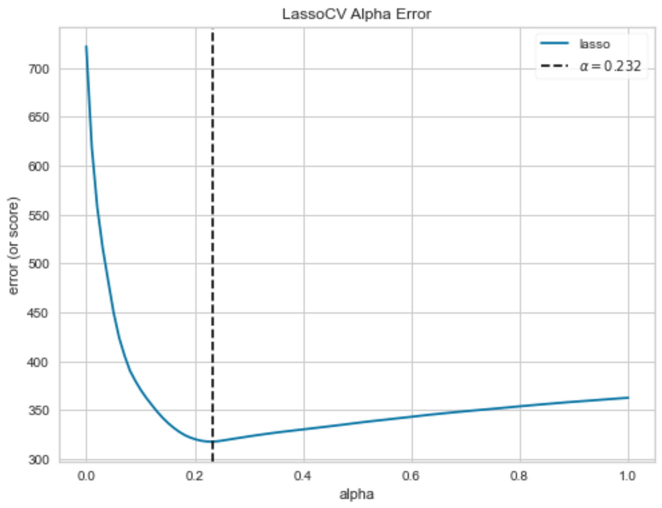


Figure 2

Using above alpha values, our L1 model selected the below features (figure 3) as most important. We decided to keep the number of features to 15 so that the model is a generic model.

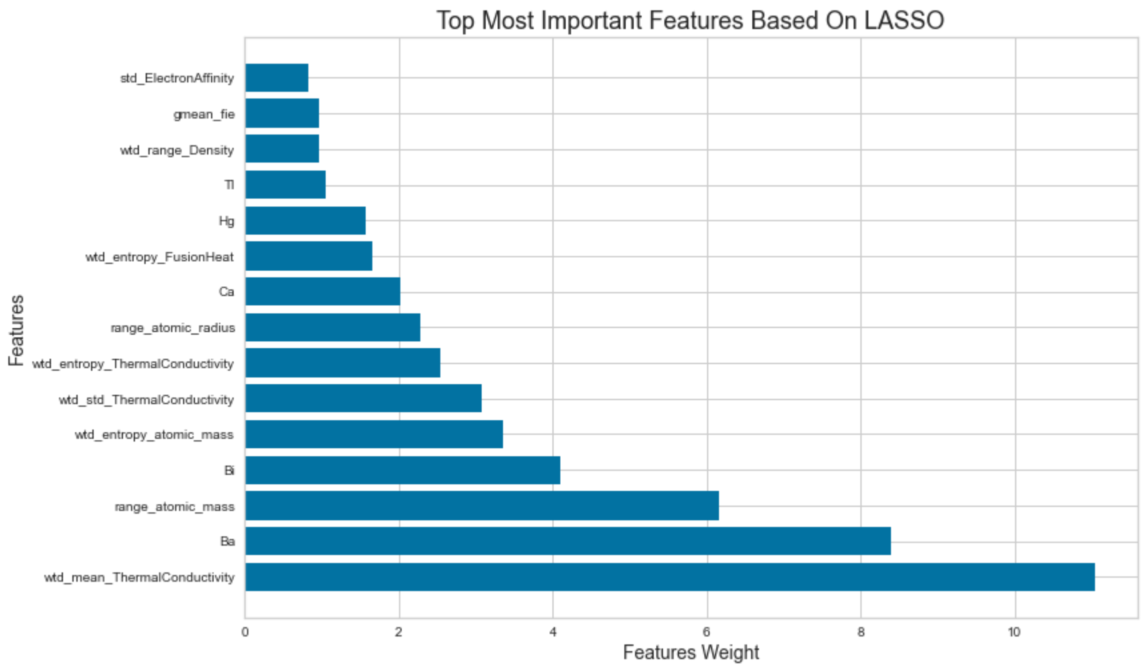


Figure 3

Here we see that features related to Thermal Conductivity is having importance in the model. It does make sense based on our basic science knowledge.

L2 regularization: L2 or RIDGE regularization uses a penalty term that is based on the size of the coefficient squared. Hence, it's called second order regularization. Unlike L1, L2 provides a general method to prevent overfitting. With L2 the coefficients are not driven to 0. Thus, weak contributions while being suppressed are not suppressed completely to 0s before strong interactions. All the interactions are inhibited or suppressed together, which makes L2 an excellent tool for overfitting for this linear regression model and as well as other more advanced models.

$$\lambda \sum_{j=0}^{k} {m_j}^2$$

Alpha selection Ridge: We performed 5-fold cross validation to get the best value of alpha to use for final model using Ridge regularization. Alpha value of 0.1 obtained from cross-validation was used to build ridge model.

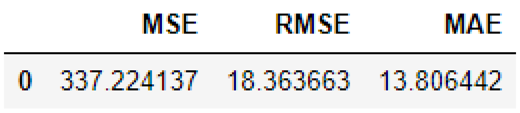
Final ridge model was built using important features obtained from L1 and best alpha value.

Cross-Validation: We will use K-Fold cross-validator with shuffle = true, to shuffle the data before splitting into batches to validate our model. With K-Folds, our full dataset is first randomly partitioned into a user-specified k number of subsets of data, in our case we will specify k=5.

Dataset of 21263 instances was divided into five separate subsets of data, and then run through five successive cross-validation implementations, in which four of the five subsets represent the training set, and the remaining set the test. Cross-validation provides an iterative look at our model's performance. So, implementing 5-fold cross-validation will produce the five metrics representing the negative mean of the errors of Ridge model. These values are manipulated to find MAE, MSE and RSME.

3. Result:

After the model was trained on important features obtained through L1 regularization and best alpha value, we used three different evaluation metrics MAE (Mean Absolute Error), MSE (Mean Square Error) and RMSE (Root Mean Square Error) to evaluate our model. Scaled test dataset was used to make prediction using the model. Using the original target value (critical temperature) and predicted critical temperature we ran our metric function to generate evaluation results. The table (table 3) below shows the model evaluation metric.



	MSE	RMSE	MAE
0	337.224137	18.363663	13.806442

Table 3

Another popular way to validate regression model is to check for the residual homoscedasticity. The assumption is that after fitting the model on training data set, the residual errors should be independent and identically distributed randomly or normally distributed should have constant variance. The plot below (figure 4) shows that the model meets the assumption as we can see that the residuals are randomly distributed.

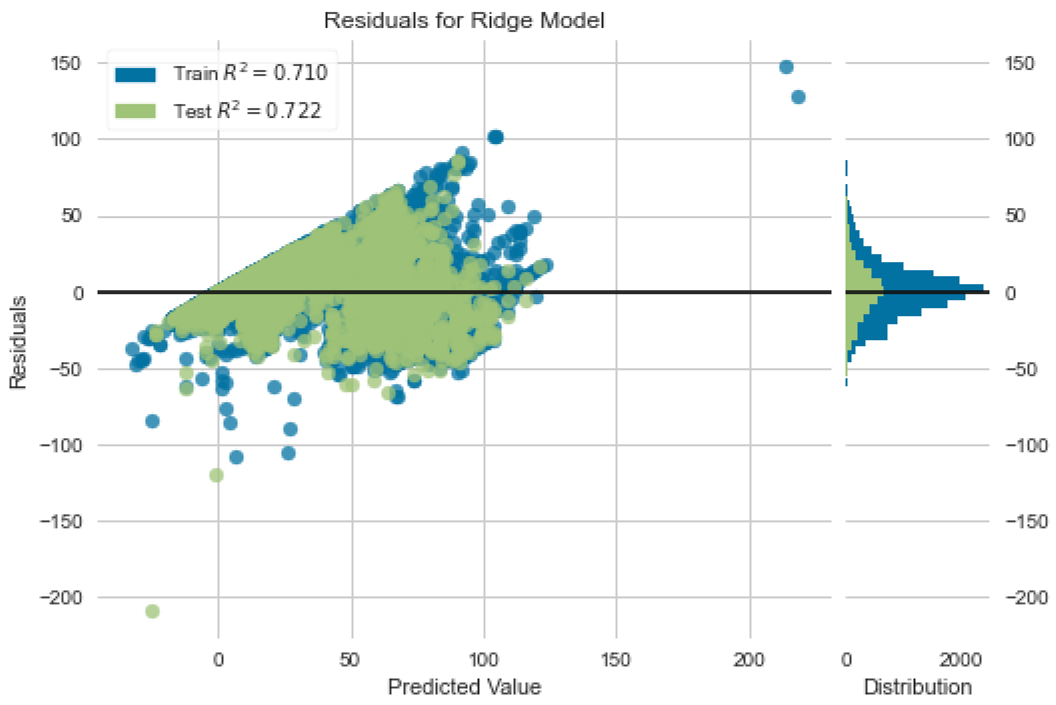


Figure 4

Weight of variables used to make final prediction is shown is figure 5 below.

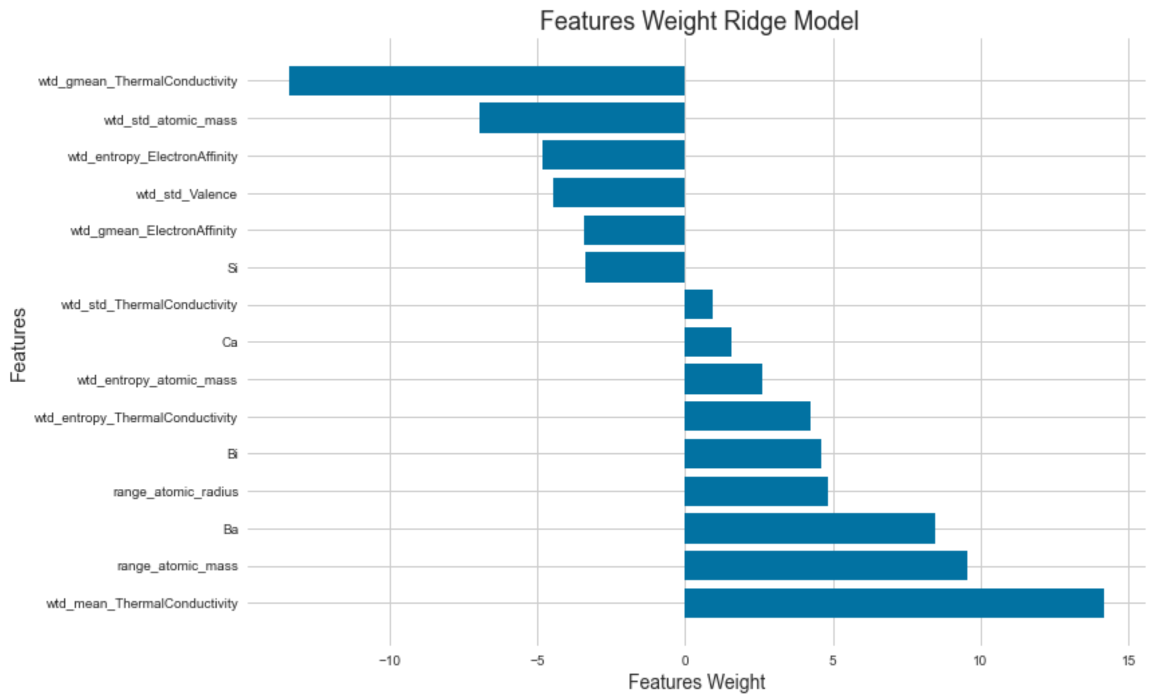
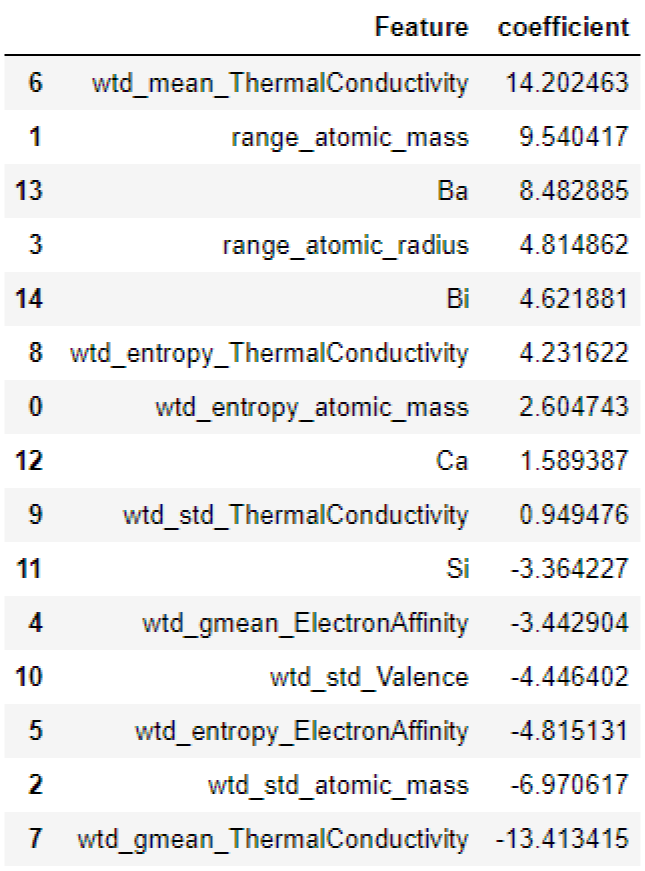


Figure 5



	Feature	coefficient
6	wtd_mean_ThermalConductivity	14.202463
1	range_atomic_mass	9.540417
13	Ba	8.482885
3	range_atomic_radius	4.814862
14	Bi	4.621881
8	wtd_entropy_ThermalConductivity	4.231622
0	wtd_entropy_atomic_mass	2.604743
12	Ca	1.589387
9	wtd_std_ThermalConductivity	0.949476
11	Si	-3.364227
4	wtd_gmean_ElectronAffinity	-3.442904
10	wtd_std_Valence	-4.446402
5	wtd_entropy_ElectronAffinity	-4.815131
2	wtd_std_atomic_mass	-6.970617
7	wtd_gmean_ThermalConductivity	-13.413415

From the above plot we can see feature **wtd_mean_ThermalConductivity,** is one that is contributing most to the prediction of critical temperature. Hence by keeping all features constant, each unit difference in **wtd_mean_ThermalConductivity** will have effect of 14.20 unit of critical temperature difference to make superconductor. Similarly, **range_atomic_mass, Ba, range_atomic_radius** etc. will positively affect the critical temperature.

On the other hand, **wtd_gmean_ThermalConductivity, wtd_std_atomic_mass, wtd_entropy_EntropyAffinity** etc. has negative effect on critical temperature.

4. Conclusion:

We were able to predict critical temperatures for super conductors based on our limited domain knowledge and linear regression techniques. However, further analysis can be done on this dataset by using Principal Component Analysis (PCA) so that important features can be selected in a more reliable way out of the provided vast range of features.