Machine Learning

Course Project Report

(Draft - 01, Team No: 01)

Title of the project: Superconductivity Data

Student 1 : Sarang Galada, sarang.g-25@scds.saiuniversity.edu.in
Student 2 : R Sai Pranav Aadhitya, saipranavaadhitya.r-25@scds.saiuniversity.edu.in

ML Category: Regression

---

1. Introduction

Superconductivity is the property of certain materials to exhibit zero electrical resistance, ie. ability to conduct direct current electricity without energy loss, when cooled below their critical temperature. Due to its unique advantages, a lot of research has been directed towards superconductivity and its applications. Despite that, the relationship between Critical Temperature (below which the material behaves as a superconductor) and the material's chemical properties is poorly understood. This has propelled efforts to use Machine Learning techniques to model critical temperature and understand its relationship with the material's chemical properties.

- Problem Statement:

  Predicting the Critical Temperature of the Superconductor compounds based on the values of their various chemical properties.

2. Dataset and Features

- Details of the dataset:

  This project makes use of the Superconductivity dataset from University of California, Irvine's Machine Learning Repository, sourced from Dr. Kam Hamidieh. The dataset consists of 21,263 rows and 82 columns. The rows represent different superconductor compounds (eg. La_2CuO_4), while the columns depict various statistical measures of their chemical properties (eg. mean Atomic Mass, standard deviation of Thermal Conductivity, range of Heat of Fusion and so on). Of these 82 columns, 81 are the input features for our learning model, with the 82nd column `critical_temp` (Critical Temperature) being the target variable we wish to predict.

- <u>Exploratory Data Analysis</u>:

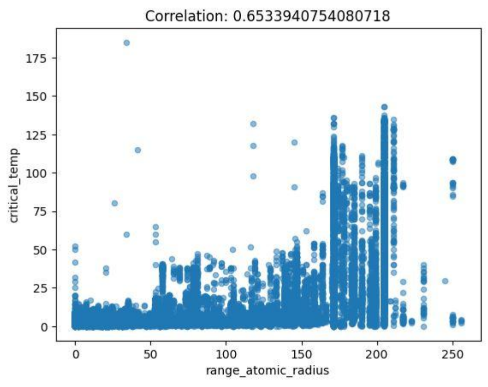
  As mentioned earlier, the dataset contains 21,263 rows and 82 columns.

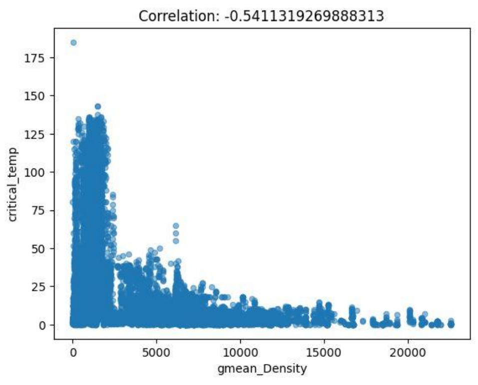
	number_of_elements	mean_atomic_mass	wtd_mean_atomic_mass	gmean_atomic_mass	wtd_gmean_atomic_mass
0	4	88.944468	57.862692	66.361592	36.116612
1	5	92.729214	58.518416	73.132787	36.396602
2	4	88.944468	57.885242	66.361592	36.122509
3	4	88.944468	57.873967	66.361592	36.119560
4	4	88.944468	57.840143	66.361592	36.110716

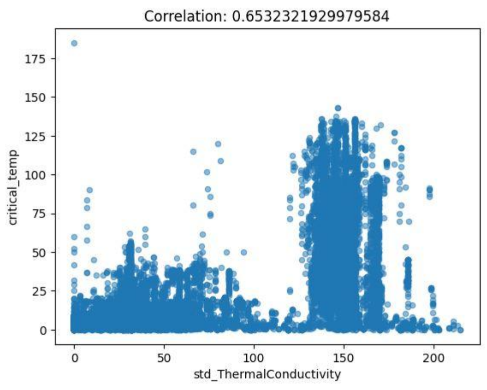
Peak into the first 5 rows of the dataset

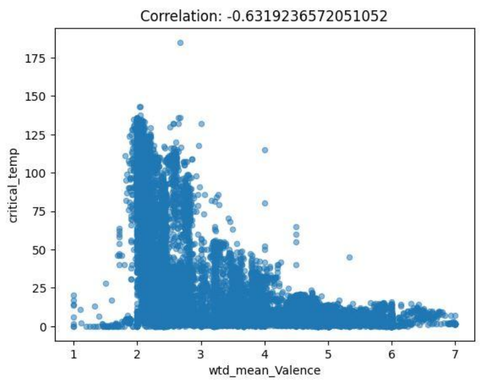
Majority of the columns are of type float64, with the exception of two (`number_of_elements`, `range_Valence`), which are of type int64 and hence take discrete values. The dataset doesn't contain any missing values, although it was found to contain 66 duplicate rows, which had to be dropped as part of data cleaning.

To understand the relationship between the features of the dataset and the target variable `critical_temp`, we make use of scatterplots and correlation matrices. Samples of some scatterplots are shown below.



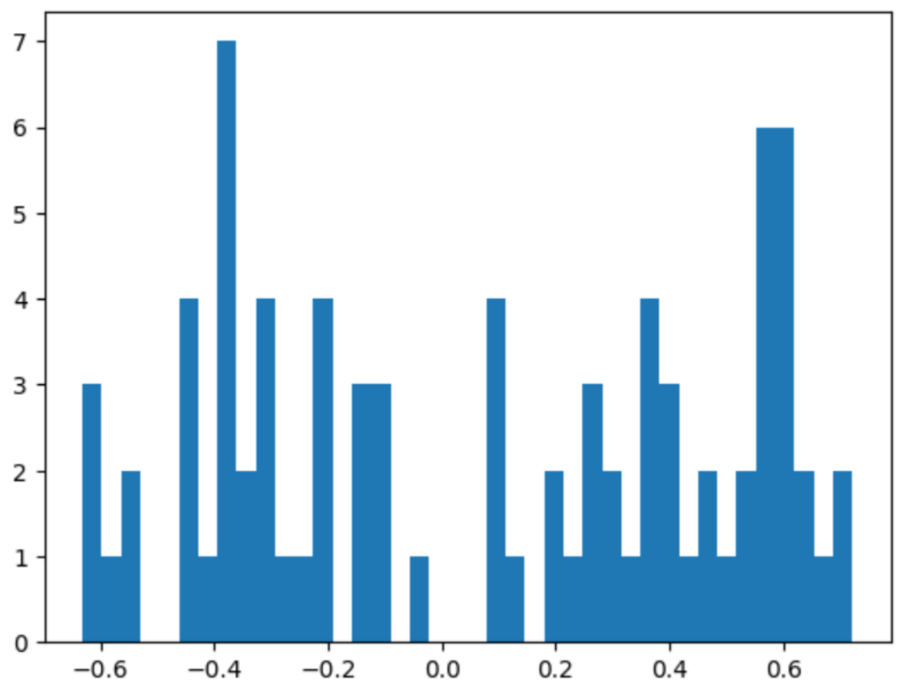






As can be seen, we do not observe distinct linear patterns emerge between the features with our target variable `critical_temp`, but still do observe an increasing or decreasing trend in the case of some features. The values in the correlation matrix support this fact, as those features show a reasonably high absolute value of correlation with `critical_temp`, indicating some amount of linear dependence.

Here's a histogram showing distribution of correlation values of all the features with target variable `critical_temp`.



3. Methods

3.1 Baseline - Linear Regression

- The Method:

  We begin by implementing the baseline model, which in this case is a multivariate linear regression model. The critical_temp column is stored as our target variable (represented by y) and the remaining are used as input features of the model (represented as X). Using Scikit Learn's inbuilt functions, we split the data into 80% training and 20% testing as is usually recommended. After using Scikit Learn's inbuilt LinearRegression class to fit our data, we obtain the intercept and coefficients of the multidimensional hyperplane that best represents the point cloud of the data instances.

Next, we evaluate the performance of this model using the R^2 score, which tells us how much better the model performs compared to just using the mean value as the model's prediction.

$$R^2 = 1 - \left(\frac{SS_{res}}{SS_{tot}}\right)$$

To further understand the accuracy of our model, we visualise:

- ➔ Scatterplot of the actual values against the predicted values
- ➔ Histogram of the residual errors, ie. difference between the actual and predicted values

Finally, to evaluate the model's ability to generalise new data, we calculate the Cross Validation score. Cross Validation splits the training set into k equal parts (in our case, we chose k=5) and alternates between the parts in each round selecting one as the testing set and the remaining combined as the training set. The score consists of the mean and standard deviation in R^2 values across the k folds of the Cross Validation. The mean tells us the average performance of the model compared to the mean model, while the standard deviation tells us the model's sensitivity in performance to changes in data. It is therefore a more comprehensive metric to evaluate a model compared to the R^2 score.

Results and graphs obtained by applying this method and other experiments can be found under the section 'Results'.

- We implement the above method with three variations. In the first, we directly apply the Linear Regression technique to the cleaned dataset. In the second, we first apply Feature Selection to our dataset and then proceed with the Linear Regression as before. More details regarding the Feature Selection can be found in the 'Protocol' section. In the final variation, in addition to the Feature Selection, we apply Feature Scaling by standardising data values so that for all features, the mean is 0 and variance is 1 . This 'standardises' all the features to a common scale.

4. Experiments & Results

4.1 Protocol

- Data Splitting:

  The data was split into 80% training and 20% testing, and reproducibility was added with a seed value of 42.

- Data Preprocessing.

  - Data cleaning:

    The dataset was searched for missing values and duplicates, which were dropped if found. Although the Superconductivity dataset had no missing values, it had 66 duplicate row instances, which had to be dropped.

  - Feature Selection:

    From the EDA, we notice that most of the columns (about 3/4th) have an absolute correlation above 0.261 with `critical_temp` upon observing the correlation matrix. We hence decided to apply feature reduction to the dataset in order to boost the speed of computation by removing less relevant features. We hence got the absolute values of the correlation values with `critical_temp`, and eliminated about 25% of the columns that were correlated by less than ~0.261.

  - Feature Scaling:

    In addition to feature selection, we were interested in reducing the error further by applying standard scaling. We hence scaled the values of X, and applied the the regression fit function on the scaled dataset. We observed that, this did not make too much of an impact on the R^2 score - it was the same as before up to the 12th decimal, and the model continued to be almost exactly as accurate as it was before standard scaling was applied. Since we're not working with polynomial terms in this case, this was quite an expected result.

4.2 Results

- Baseline results:

1) **R^2 Score**:

Cleaned original data	After feature selection	After feature selection and standardisation
0.7353599049008277	0.7232760543024528	0.7232760543024641

The R^2 values in the range 0.72-0.74 tells us that the model is able to make moderately accurate predictions on the dataset.

2) **Cross Validation Score**:

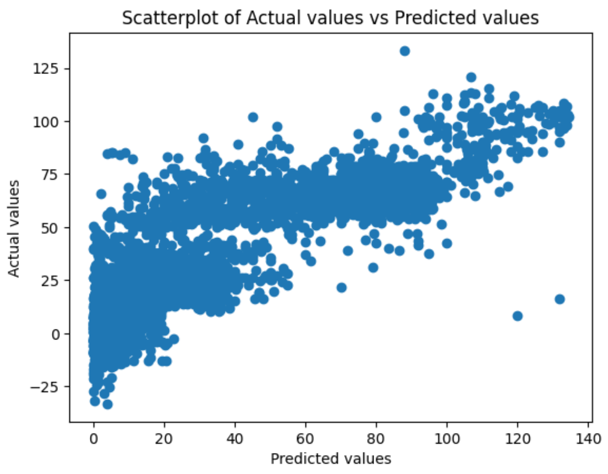
	Cleaned original data	After feature selection	After feature selection and standardisation
Mean	0.7337205416664668	0.7165469130217834	0.7165469130217896
Std. Deviation	0.01813007522420121	0.019210403416915323	0.01921040341692294

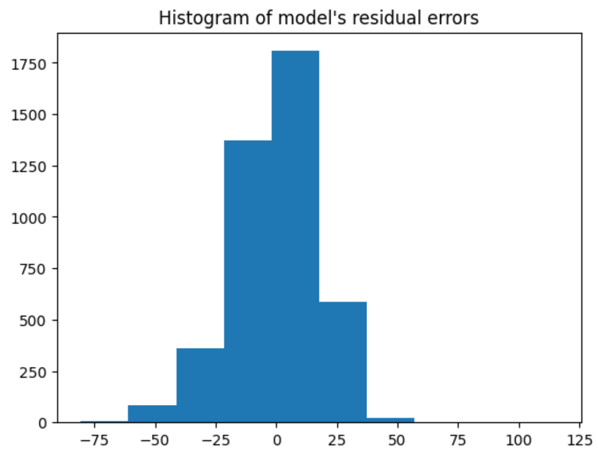
The Cross Validation scores very closely match the R^2 scores calculated previously, indicating neither overfitting nor underfitting. The low standard deviation captures this, conveying that the model isn't sensitive to small changes in data and can adapt well to new data instances.

What is interesting to note is that Feature Selection actually reduces the performance and increases sensitivity marginally. This may be because the 21 features that have been dropped actually contribute in predicting the target variable, although by a small magnitude. Feature Scaling evidently has no impact on the model's performance or sensitivity. This is understandable, as standardisation only brings the features down to a common scale, without making any change to the relation between the features. Thus, we see that standardisation doesn't actually impact the performance of our model.

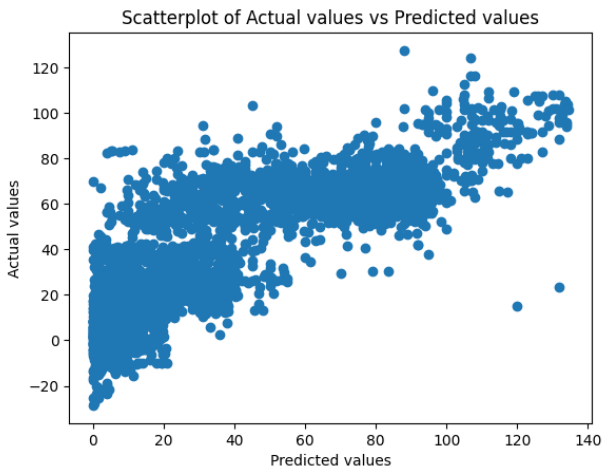
3) **Graphs**: Scatterplots & Histograms comparing Actual and Predicted values:

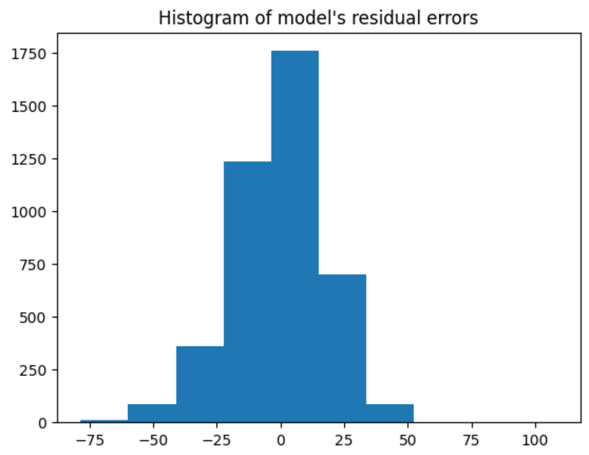
a) Original dataset



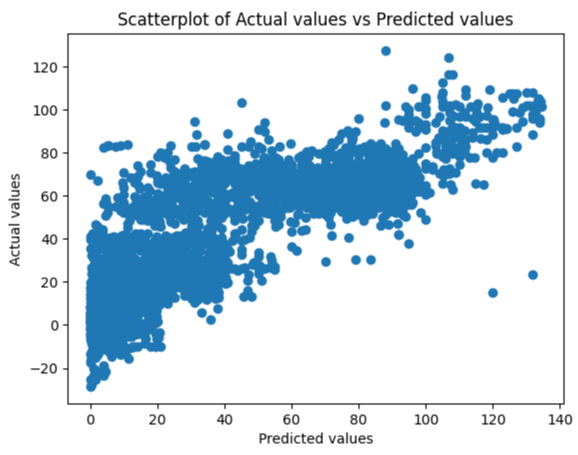


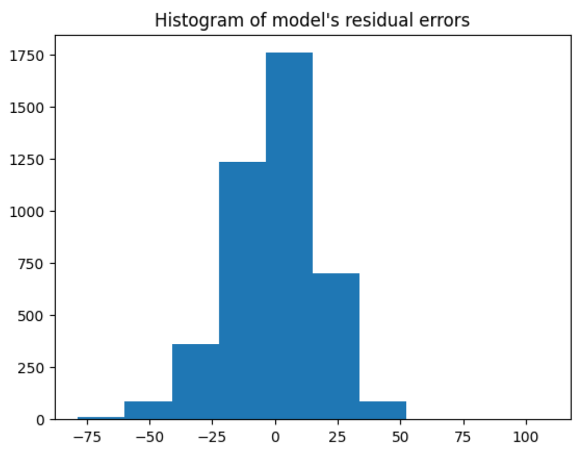
b) After Feature Selection





c) After Feature Selection & Scaling





Once again, there are minimal differences between the graphs for the three variations of the dataset. This hints that other techniques need to be explored for dropping features and improving performance.

5. Discussion

- Not required in first draft

6. Conclusion

- Not required in first draft

7. References

[1] UCI Machine Learning Repository: Superconductivty Data Data Set
[2] Machine learning modeling of superconducting critical temperature | npj Computational Materials
[3] 2.7.3: Scatter Plots and Linear Correlation - K12 LibreTexts
[4] A data-driven statistical model for predicting the critical temperature of a superconductor - ScienceDirect
[5] What and why behind fit_transform() and transform() in scikit-learn!
[6] When and why to standardize a variable