Case Study 1: Predicting Critical Temperature

By: Allen Hoskins

Introduction

Superconductors are materials that conduct electricity with little or no resistance without onset or buildup of heat. Due to this process, superconductors can create a magnetic field and generate a constant flow of electricity. These materials have a property called `critical temperature`. This property is the temperature at which this material acts as a superconductor. While most materials have an extremely low critical temperature (between 0 and 10 Kelvin), research has been ongoing to find materials with higher critical temperatures.

In this case study, we will utilize Linear Regression with both L1 and L2 regularization to predict the critical temperature of a compound to potentially identify superconductors.

METHODS

DATA PREPROCESSING:

The original data is composed of two separate CSV files, named `train.csv` and `unique_m.csv`. Once importing the needed packages for the case study, I read in the data and determined the shape and size of both datasets. The `train.csv` dataset consisted of 21,263 rows with 82 columns, and the `unique_m.csv` dataset consisted of 21,263 rows with 88 columns. The two datasets were then able to be merged on the index, and the duplicate response variable of `critical_temp` was able to be dropped, resulting in a final dataframe consisting of 21,263 rows with 168 columns. Before proceeding to Exploratory Data Analysis (EDA), I checked variable datatypes to determine if any other preprocessing needed to be done. With every datatype consisting of a float or integer, I was able to move on to EDA.

EXPLORATORY DATA ANALYSIS (EDA):

With little information about the 168 explanatory variables and response variable within the data set, I needed to determine if the data needed to be scaled. I first plotted a histogram of the response variable to see the distribution (Fig. 1). The histogram showed that the `critical_temp` was heavily right skewed. To determine if any other variables in the data were heavily skewed, I ran quick descriptive statistics and plotted histograms of the following explanatory variables: `number_of_elements`, `entropy_atomic_mass`, `wtd_mean_atomic_mass`, `critical_temp`. All of these variables appeared to have a large variances and non-standard distributions (Fig 2).

Fig. 1

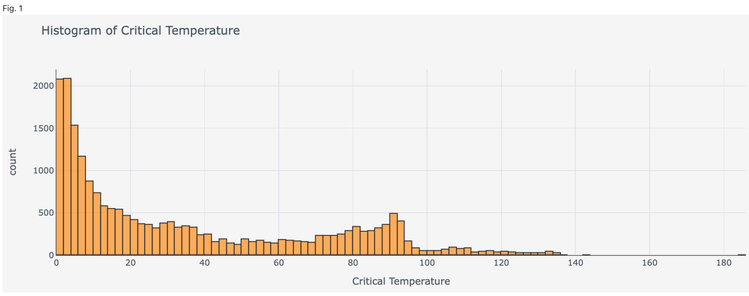
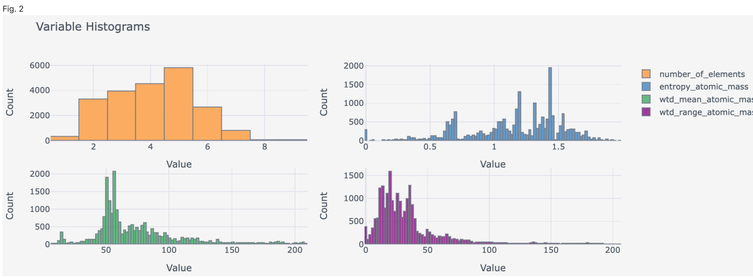


Fig. 2



LINEAR REGRESSION ASSUMPTIONS:

Before performing a linear regssion model, I needed to check the assumptions. These consisted of:

> Linearity: PASS

> Homoscedasticity: FAIL

> Independence: PASS

> Normality: FAIL

After testing the assumptions and determinign that they did not all pass, I needed to prepare the data with transformations below.

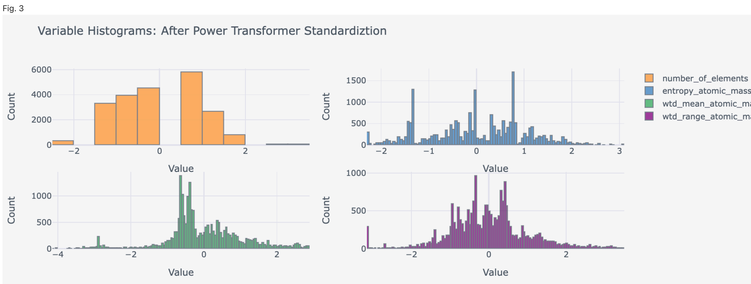
PREP DATA TO MODEL:

Before scaling any of the explanatory variables, the data was separated into explanatory variables (X) and response variable (y). Once separated, I was able to scale the data. Sklearn provides multiple scalers to transform the data, but only `StandardScaler` and `PowerTransformer` were considered at this time. According to Sklearn's website, the definitions of the scalers are as follows:

> StandardScaler: Standardize features by removing the mean and scaling to unit variance. PowerTransformer: Apply a power transform feature wise to make data more Gaussian-like. Power transforms are a family of parametric, monotonic transformations that are applied to make data more Gaussian-like. This is useful for modeling issues related to heteroscedasticity (non-constant variance), or other situations where normality is desired. Supports Yeo-Johnson and Box-Cox.

Both StandardScaler and PowerTransformer were run though the ElasticNet model, and since PowerTransformer produced the lowest mean RMSE, this will be the data set used moving forward. Note that since there are positive and negative integers in the data set, Yeo-Johnson was used instead of Box-Cox. Fig. 3 contains histograms of variables in Fig. 2 after utilizing `PowerTransformer`.

Fig. 3



RESULTS

For modeling, I determined that using Sklearn's `ElasticNetCV` model was appropriate as it combines the `l1` and `l2` regularization of `Lasso` and `Ridge` models.

Models use: 10-fold Cross validation (`Kfold`), `random_seed = 0`, and `max_iter = 20000`.

Model GridSearchCV:

After preprocessing, EDA, and scaling the data, modeling was able to begin. Utilizing the power of Sklearn's `GridSearchCV`, the hyperparameters of `l1_ratio`, `tol`, and `eps` were run though the model and the best output using the `neg_root_mean_squared_error` scoring were output to be used in the final model.

Grid Search Parameters:

```
"l1_ratio": np.arange(0.0, 1.0, 0.1),
"tol":      [1e-9, 1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1],
"eps":      [1e-3, 1e-2, 1e-1, 1, 10, 100]
```

Best Model Output:

```
"l1_ratio": 0.2
"tol":      0.01
"eps":      0.001
```

ElasticNetCV with GridSearchCV Tuned Parameters:

After performing GridSearchCV to tune the model parameters, Sklearn's `cross_validate` was used to validate the model and determine final performance. The results of all 10 folds are below with a mean RMSE of 16.4637.

	fit_time	score_time	estimator	test_score	train_score
0	2.19916	0.00117993	ElasticNetCV(l1_ratio=0.2, max_iter=10000, random_state=0, tol=0.01)	16.7378	16.3605
1	2.15888	0.00172806	ElasticNetCV(l1_ratio=0.2, max_iter=10000, random_state=0, tol=0.01)	16.7811	16.3389
2	2.09171	0.00159836	ElasticNetCV(l1_ratio=0.2, max_iter=10000, random_state=0, tol=0.01)	16.2488	16.4508
3	2.13909	0.00134301	ElasticNetCV(l1_ratio=0.2, max_iter=10000, random_state=0, tol=0.01)	16.4203	16.4222
4	2.04526	0.00174427	ElasticNetCV(l1_ratio=0.2, max_iter=10000, random_state=0, tol=0.01)	16.5608	16.4041
5	2.05884	0.00163698	ElasticNetCV(l1_ratio=0.2, max_iter=10000, random_state=0, tol=0.01)	15.7617	16.4887
6	2.1159	0.00138688	ElasticNetCV(l1_ratio=0.2, max_iter=10000, random_state=0, tol=0.01)	16.2624	16.446
7	2.06758	0.00157595	ElasticNetCV(l1_ratio=0.2, max_iter=10000, random_state=0, tol=0.01)	16.7527	16.397
8	2.10658	0.00192094	ElasticNetCV(l1_ratio=0.2, max_iter=10000, random_state=0, tol=0.01)	16.4909	16.4127
9	2.11754	0.00175214	ElasticNetCV(l1_ratio=0.2, max_iter=10000, random_state=0, tol=0.01)	16.6208	16.4005
MEAN	2.11005	0.00158665		16.4637	16.4121

Feature Importance:

After completing the ElasticNetCV modeling, I wanted to determine which features were the most important in predicting `critical_temp`.

The top 10 features in the model were:

	Feature	Coefficient
19	Cu	4.99197
7	Ba	4.88638
12	Ca	4.83552
38	La	-3.48937
163	wtd_std_Valence	-3.24568
9	Bi	2.40671
162	wtd_std_ThermalConductivity	2.29054
57	Pr	-2.27065
31	Hg	2.2417
146	wtd_mean_ThermalConductivity	1.89972

CONCLUSION

In conclusion, the best model that was chosen used a combination of L1 and L2 regularization (`l1_ratio = 0.2`). This model produced a mean RMSE of 16.4637. While not all models are useful, the output of this model can be used as a base in determining superconductors.

CODE:

Attached in file CS1_CODE.ipynb