Timothy Cabaza

May 2024

Case Study1

**Problem Statement**:

The aim of this case study is to build a linear regression model using L1 (Lasso), L2 (Ridge), and ElasticNet. The task is to predict the Critical Temperature as closely as possible using the Super-Conductor dataset. In addition, the top 5 features that are most important and the best regularization parameter for each model.

**Exploratory Data Analysis:**

The super-conductor data was loaded. There were two files that the client has asked that we group together. While rare, in this case the data is clean with no missing data and thus no imputations to consider.

The traincsv was loaded and inspected. The column dytpes are numeric, there are no null values and there 82 columns with 21263 rows of data (Figure 1). The describe function was also used but given the 82 columns it would be an inefficient use of space to include in our table below.

|  |
| --- |
|  |
| **Figure 1: train dataset loaded and inspected**  Utilized pandas functions such as info(), shape, and describe to inspect the first of two datasets for Super Conductors. |

The second dataset was loaded, unique\_m csv, and inspected in a similar fashion.

|  |
| --- |
|  |
| **Figure 2: unique\_m dataset loaded and inspected.**  Utilized pandas functions such as info(), shape, and describe to inspect the second of two datasets for Super Conductors. |

Once the data was loaded and inspected for any missing values, appropriate data type, and shape, the columns for each file were identified. Before combining and splitting the super conductor datasets into target variable and features, duplicate columns in both datasets (‘critical\_temp’) were found and one was dropped before joining. The 'material' column was dropped from the unique csv as it is a composite of all the other features in the data and it would be redundant to include as a feature in the models. To perform drops

|  |
| --- |
|  |
| **Figure 3: Identifying duplicative and redundant columns**  Before joining the train csv and the unique csv the drop and columns functions were utilized to identify duplicative and redundant columns. |

The client has asked that both datasets be joined so that we have one joined dataset to train and evaluate our models on. The pandas concat function was called to join the two dataframes. The columns and head functions was then used to once again verify the data was joined appropriately. The shape function was utilized to check the number of rows and columns in the dataframe (21263, 168).

|  |
| --- |
|  |
| **Figure 4: Joining datasets and checking columns and shape** |

The joined dataframe (joined\_df) still had the target column (‘critical\_temp) in the dataframe, so the target variable was created and the target column 'critical\_temp' isolated and dropped using the drop function.

|  |
| --- |
|  |
| **Figure 5: Target Variable dropped from the joined dataframe.**  The target variable we use double brackets on the line *target = joined\_df[['critical\_temp']]* so that the target is a pandas dataframe rather than pandas series, which allows us to use the columns function to print the column name 'critical\_temp'. |

The target and the features (joined\_features) are now in their own respective dataframes. Before moving forward, the data is then visualized to view the distribution (normal, skewed, bimodal, central tendancy, spread, outliers) and extract insights into the feature relationships.

|  |
| --- |
|  |
| **Figure 6: Distribution of target variable**  The distribution is right skewed. Utilized matplotlib histogram to create visualization. |

The histogram function was called with 50 bins and the histogram for the target was generated. The histogram indicates that the target 'critical\_temp' is right skewed with most of the temperatures clustered at the lower end with a long tail. The takeaway here is that that the critical temperature for most of the materials are usually on the lower end with a only a few higher critical temperature values. The temperatures above ~90 Kelvin on the right end of the tail of the distribution could possibly impact the performance of the regression models. However, we will proceed with no transformation on the target (dependent) variable to preserve interpretability, any transformation would make it difficult to understand the predictions of the model in the context of the problem statement.

Given that our joined features dataframe has a shape of (21263, 167), histograms or any other type of visual would be inefficient in the report, instead the correlation of the features versus the target variable and the description of the data is utilized to assess our features and the next steps.

|  |
| --- |
|  |
| **Figure 7: Top 20 feature correlations with the target variable and description of features 10 out of the 167 features**  Tabulate package was imported to create organized tables. |

The resulting table (Figure 7) illustrates that the features need to be normalized as we have a wide range between min and max values as well as high standard deviations. To address the spread in values and variation in averages the standard scaler package was utilized so that the models will not give undue importance to features with larger values. The scaling will transform the features into a comparable range with a mean of zero and a standard deviation of one. By scaling the data our models should theoretically perform better as the models assumes that the features are centered around zero and have a similar scale.

|  |
| --- |
|  |
| **Figure 8: Features after scaling using Standard Scaler.** |

**Modeling:**

|  |
| --- |
| **L1-Lasso** |
| **Figure 9: The alpha parameter is grid searched over a log space to find the optimal regularization parameter.**  The top 5 features are listed and the Best Model are captured in the tables. |

|  |
| --- |
| **L2-Ridge** |
| **Figure 10: The alpha parameter is grid searched over a log space to find the optimal regularization parameter.**  The top 5 features are listed and the Best Model are captured in the tables.  **Adjsted alpha since the l1 alpah logspace used came back with 10 as our best alpha, so the logspace was shifted**  **# alphas = np.logspace(-6, 1, 50) # going from 10^6 to 10^1 with 50**  **alphas = np.logspace(-6, 2, 50)** |

|  |
| --- |
| **Elastic Net** |
| **Figure 11: The alpha parameter is grid searched over a log space to find the optimal regularization parameter.**  The top 5 features are listed and the Best Model are captured in the tables. |

**Summary:**

In conclusion, the best performing model \_\_\_ with a regularlization parameter \_\_\_ and MSE of \_\_\_. The model ranked the following as the top 5 most important variables \_\_\_\_\_.

Describe alpha ranges scanned and how you adjusted for each model – ridge started with the same l1 but 10 was the best alpha so the alpha grid search was altered.

|  |
| --- |
|  |
| **Figure 12: Summary tables from each model with best regularization parameter and top 5 most important features for each model.** |

**Appendix:**

|  |  |
| --- | --- |
| Code: | Dfddf |