Predicting Superconductivity Critical Temperature

CS5014 Practical 1

190027392

March 2020

Contents

List of Figures

List of Tables

List of Listings

1 Introduction

2 Loading, Splitting and Cleaning

The dataset provided is a compilation of 21,263 super conductor containing 81 attributes compiled from Japan's National Institute for Materials Science.

Splitting the dataset The dataset was immediately split into training and testing data, the latter being discarded away for evaluation purposes. The training dataset was used during training and development of the models. This was implemented in the generate_data.py file which returns these two split datasets.

Pipeline A Pipeline by SKLearn was used to automate the cleaning tasks as shown in Listing 2. This would allow the training data to be cleaned up without inspecting it hence reducing the risk of Data Leakage. The Dataframe selector is a helper function designed to select only a subset of the features as to not apply this pipeline to the output feature.

```
def get_pipeline(inputs) -> Pipeline:
    return Pipeline([
        ('selector', DataFrameSelector(inputs)),
        ('std_scaler', StandardScaler()),
    ])


def apply_pipeline(data, x_col, y_col):
    y = data[y_col]
    data = DataFrame(get_pipeline(x_col).fit_transform(data),
    ↪  columns=x_col)
    data['critical_temp'] = y
    x, y = data[x_col], data[y_col]
    return data, x, y
```

Listing 1: The Pipeline for Standardising the Data

3 Visualising The Data

The describe.py script plots a number of charts attempting to display the and describe the data. Due to the high number of dimensions a scatter plot with some of the most important features found later during development is shown (Figure 3).

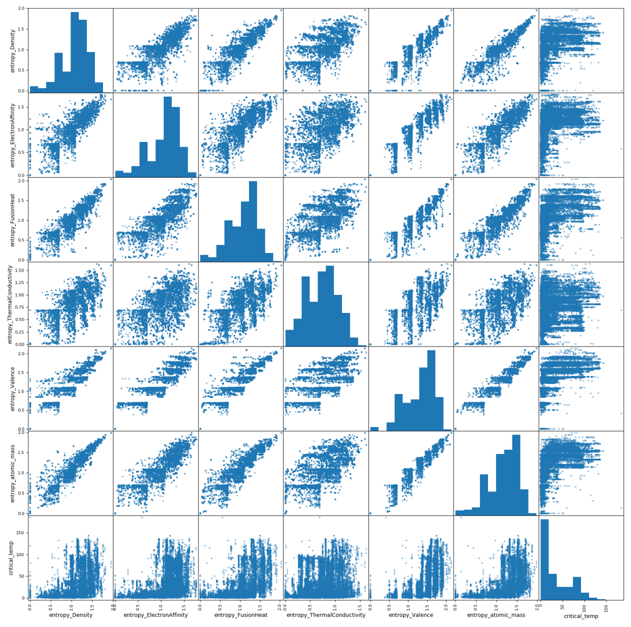

Figure 1: A scatter matrix showing the correlations between some of the important features.

Figure 3 gives a general brief overview of all the variables and their correlation. The map shows that the variables are highly correlated with one another as shown by the high concentrations of yellow in the matrix. As the data is so highly correlated, multicollinearity could be an issue. Multicollinearity is a statistical phenomenon occurring in multiple regression where a feature could be predicted linearly from the other features. This could lead to a situation where the coefficients of the model changing erratically with small changes in the data. Formally a set of variables are perfectly collinear when:

$$0 = \lambda_1 X_{1i} + \cdots + \lambda_k X_{ki} \tag{1}$$

for a regression model

$$Y_1 = \beta_0 + \beta_1 X_{1i} + \cdots + \beta_k X_{ki} + \epsilon \tag{2}$$

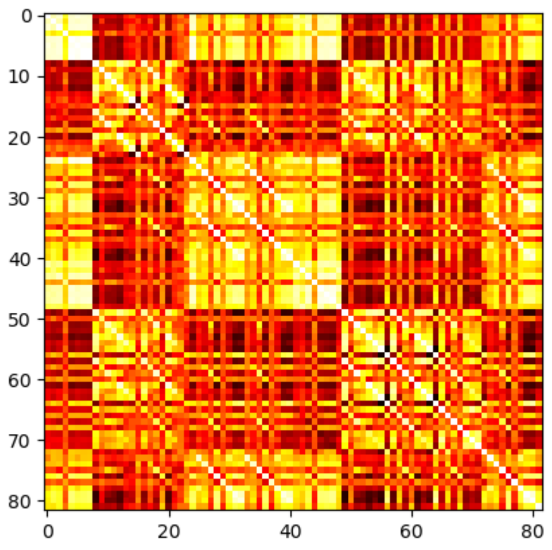

Figure 2: A heatmap of correlations. Brighter colors indicate higher correlation.

4 Feature Reduction

4.1 Variance Inflation Factor

The Variance Inflation Factor(VIF) is a measure of colinearity of the input features in multiple regression. It is defined as:

$$VIF_i = \frac{1}{1 - R_i^2} \tag{3}$$

This measure was integrated into the pipeline and the VIF for each of the columns is calculated. As a general rule columns having a score larger than 10 are removed (Kutner et al. 2005).

4.2 Principle Component Analysis

Principle component analysis is a statistic procedure that orthogonally transforms a dataset into uncorrelated variables called principle components (Pearson 1901). The first principle component x_1 covers the maximum possible variance in the data such that:

$$w_1 = \underset{||w||=1}{\arg\max} \left(\sum_i t_i^2 \right) = \underset{||w||=1}{\arg\max} \left(\sum_i (x_i \cdot w)^2 \right) \tag{4}$$

Technique	Cross Validation Score	Number of features
Benchmark	0.7342	80
VIF Elimination	0.6523	29
PCA	0.5951	16

Table 1: Scores for the feature reduction techniques.

where $w_k = w_1 \dots 1_k$ representing the vectors of the coefficients and t_i is the vector of principle component scores. The other components could be calculated subtracting the first $k-1$ principle components and finding the weight vectors that maximises the variance as before.

4.3 Selecting a Feature Reduction Technique

An experiment was devised to score these techniques on the training dataset to choose the best technique for our testing. The experiment written in feature_selection.py, spins up a cross validator, tries each of these techniques and tables up the results for comparison shown in Table 1. The dataset was standardised using the pipeline discussed in Section 2. This is important as PCA extracts the components that maximise the variance, hence having features of different scale might disproportionately affect the variance messing up the grouping. The results show that non of the techniques out performed the benchmark vanilla linear regression. None of the techniques managed to reduce the number of features to a manageable level indicating that most of the features are required to capture most of the variance in the dataset. As a result these techniques were abandoned as they performed worse than the vanilla linear regression.

5 Regression Models

In the following section a brief explanation of the models used will be given along with regularization techniques employed.

5.1 Linear Regression

Linear regression is a simple linear approach to modelling the relationship between a dependent variable and it's predictors. Formally it is described as:

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_1 x_{ip} + \epsilon_i \tag{5}$$

$$y = X\beta + \epsilon_i \qquad \text{Matrix notation} \tag{6}$$

where y is a vector of outputs, X is a matrix of row vectors x_i and $x_{i0} = 1 \ for \ i = 1, \dots, n$, β is a vector of coefficients and ϵ is a vector of error terms (Hastie, Tibshirani, and Friedman 2009). Linear regression aims to optimise β and ϵ in such a way that the cost function would be minimised. The cost function could be described as:

$$\sum_{i=1}^{M}(y_i - \hat{y_i})^2 = \sum_{i=1}^{M}\left(y_i - \sum_{j=0}^{p} w_j \cdot x_{ij}\right)^2 \tag{7}$$

where M and p

Regularisation Regularisation allows the model to generalise well over unseen samples in the real world. One problem of the standard linear regression is that the resulting model would be too complex. In this case linear regression would have 80 coefficients which could lead to an over-fitted model.

Ridge Regularisation Here the cost function described for Linear Regression is modified in such a way that a penalty is added to coefficients of large magnitude.

$$\sum_{i=1}^{M}(y_i - \hat{y_i})^2 = \sum_{i=1}^{M}\left(y_i - \sum_{j=0}^{p} w_j \cdot x_{ij}\right)^2 + \lambda \sum_{j=0}^{p} w_j^2 \tag{8}$$

λ refers to the penalty term. Setting this to 0 would turn this cost function into the one used by linear regression minimising this property.

Lasso Regularisation Like Ridge Regularisation, Lasso builds up on the Linear Regression's cost function as follows:

$$\sum_{i=1}^{M}(y_i - \hat{y_i})^2 = \sum_{i=1}^{M}\left(y_i - \sum_{j=0}^{p} w_j \cdot x_{ij}\right)^2 + \lambda \sum_{j=0}^{p} |w_j| \tag{9}$$

It takes account the magnitudes rather than the coefficients hence leading to zero coefficients and a simpler model.

5.2 Random Forest Regression

Random Forests is a machine learning technique built on Decision Trees. Decision Trees are sensitive on the trained data hence if the training data changes slightly the resulting model could be drastically different. They also tend to stick to local optima (no back tracing after splitting the tree) and risk over-fitting. Random Forests employ a bagging technique (random sampling with replacement) by deploying multiple decision trees in parallel and returns the mean prediction of the individual trees. Each tree samples randomly a sample on every split reducing the chances of over-fitting.

Model	Mean Square Error	R2
Linear regression	17.38	0.73
Ridge Regression	17.9	0.72
Random Forest	11.96	0.88

Table 2: Scores for the trained models

5.3 Metrics

Root Mean Square Error(RMSE) The RMSE error for prediction $\hat{y}_t$ and T samples is defined as:

$$RMSE = \sqrt{\frac{\sum_{t=1}^{T}(\hat{y}_t - y_t)^2}{T}} \tag{10}$$

RMSE is frequently used as the error is on the same scale of the output (in our case Kelvin). Errors are squared before taking the root hence high errors are penalised.

R^2 the R^2 error or coefficient of determination compares the resultant model with a constant baseline. The error can range from $-\infty to 1$ where negative values indicate that the model performs worse than the baseline.

$$R^2 = 1 - \frac{MSE(model)}{MSE(baseline)} \tag{11}$$

6 Results and Discussion

The evaluation script is located in the evaluate_models.py file. The script trains a Linear Regression, Ridge regression and Random forest using a K-Fold cross validator on the training dataset and then scores these models using the RMSE and R2 metrics. The script also produces learning graphs(Figure) and residual plots of the evaluated model and saves them in the fig directory. The Lasso regression was not included as this model could not converge. This could be due to a small tolerance value for the variance of the dataset. This phenomenon was not investigated further due to time constraints. Table 2 shows the scores achieved by the models whilst Figure 6 shows the residual plots for each of the model.

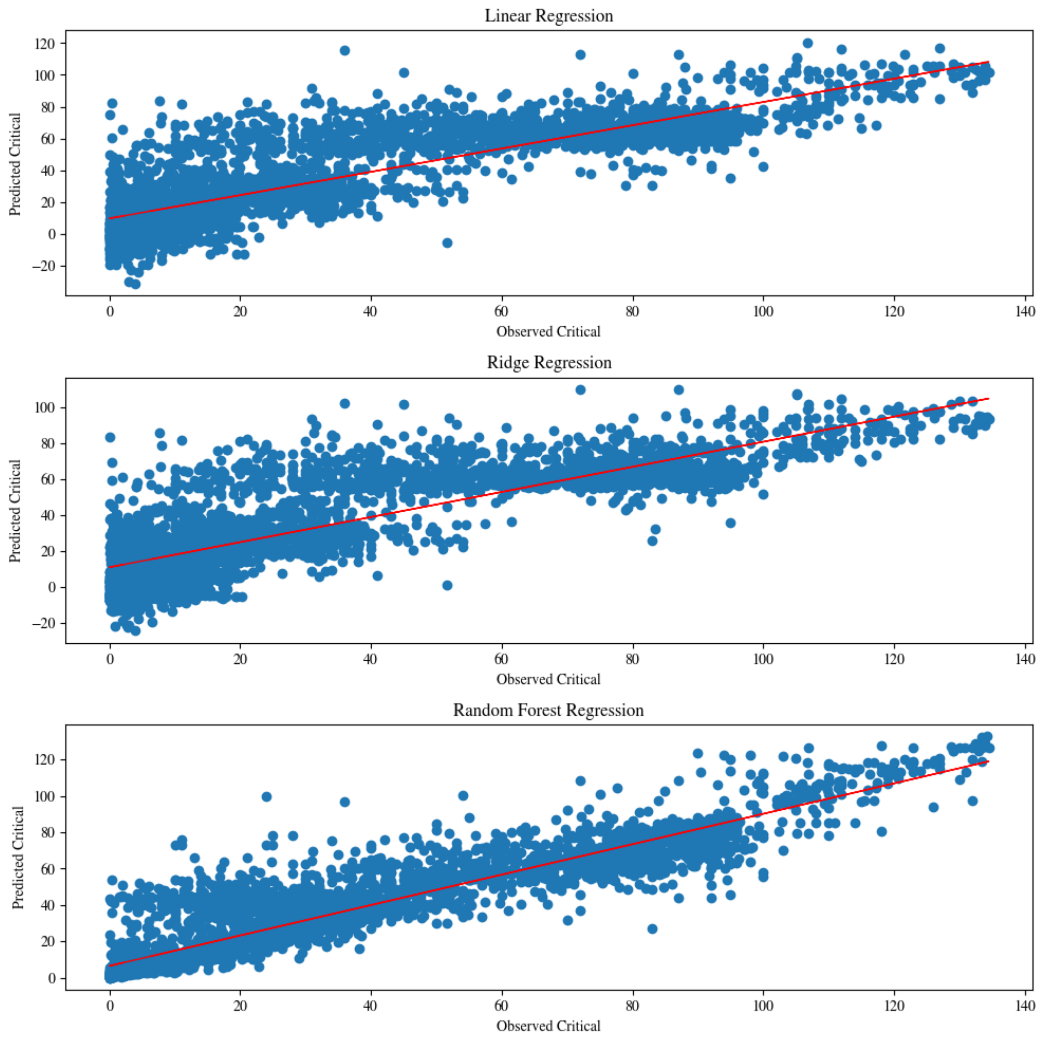

Figure 3: A matrix of residual plots of the different models.

These results coincide with the results obtained in the original paper where the RMSE obtained was 17.6 K. and a R^2 of 0.74 (Hamidieh 2018). The Ridge model is also close to the score obtained by the original paper. The XGBoost Model described in the paper also out-performs the Random Forest Regressor as it obtains a RSME of 9.5 K.

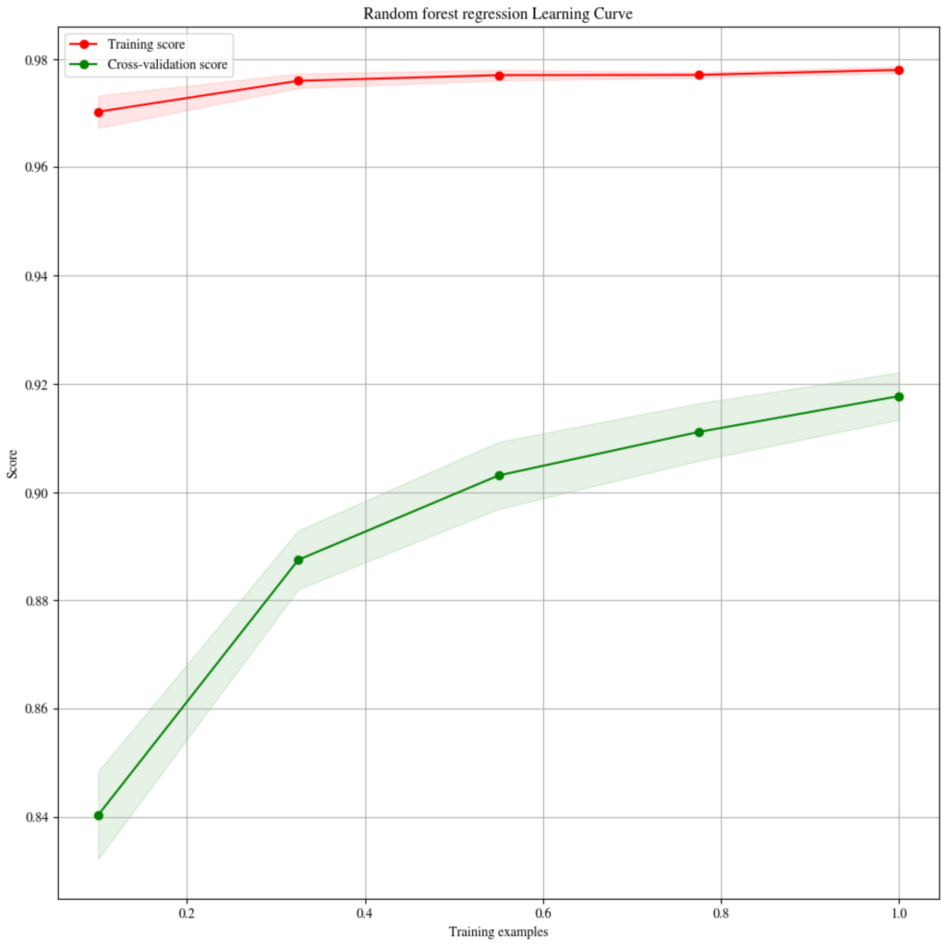

Figure 4: A plot showing the cross validation scores along with training scores for the random forest.

7 Running Instructions

The project is structured as follows:

- utils/ - containing utilities such as plotting helper functions;
- figs/ - the directory where the figures get saved;
- data/ - a directory containing the datasets (train, test and original);
- describe.py - a script that performs descriptive analytics on the dataset;
- evaluate_models.py - evaluates the trained models;

- feature_selection.py - A script that performs feature selection analysis

Listing 7 shows how to run the scripts.

```
venv/bin/python <file_name>
```

Listing 2: Running the scripts.

Dependencies

- Scikit Learn (Pedregosa et al. 2011)
- Numpy (Virtanen et al. 2020)
- Matplotlib (Hunter 2007)
- Statsmodels (Used for calculating the variance inflation factor) (Seabold and Perktold 2010)

References

Hamidieh, Kam (2018). "A data-driven statistical model for predicting the critical temperature of a superconductor". In: *Computational Materials Science* 154, pp. 346–354.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2009). *The elements of statistical learning: data mining, inference, and prediction.* Springer Science & Business Media.

Hunter, John D (2007). "Matplotlib: A 2D graphics environment". In: *Computing in science & engineering* 9.3, pp. 90–95.

Kutner, Michael H et al. (2005). *Applied linear statistical models.* Vol. 5. McGraw-Hill Irwin New York.

Pearson, Karl (1901). "LIII. On lines and planes of closest fit to systems of points in space". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11, pp. 559–572.

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Seabold, Skipper and Josef Perktold (2010). "statsmodels: Econometric and statistical modeling with python". In: *9th Python in Science Conference.*

Virtanen, Pauli et al. (2020). "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17, pp. 261–272. DOI: `https://doi.org/10.1038/s41592-019-0686-2`.