CS5014 Practical 1

160001826

4th March 2020

1 Introduction

This project involves the training and evaluation of a regression model to predict the critical temperature of superconductors. Superconductor data from the Superconducting Material Database is used in supervised learning to predict critical temperature based on a superconductor's chemical formula.

In this report, the decisions at each stage of the process are described and justified, by demonstrating the effect they have on the out-of-sample error of the model.

While [1] finds success using a gradient boosted model for predictions with this dataset, for this practical, I focus on a linear regression model. I chose to do this because it performs fairly well on this data, and by selecting just one optimisation strategy, I aim for a deeper understanding of how all the other options - introducing polynomial terms, different subsets of the data, regularisation approaches, and different evaluative metrics - can affect the performance of the model. This constitutes an extension as it extends the basic model in a meaningful way.

2 Loading and cleaning the data

The data is loaded using `pandas` from the two input files: `train.csv`, which contains numerical data describing the properties of each superconductor, including critical temperature, and `unique_m.csv`, which describes the quantity of each element in the superconductor's chemical formula.

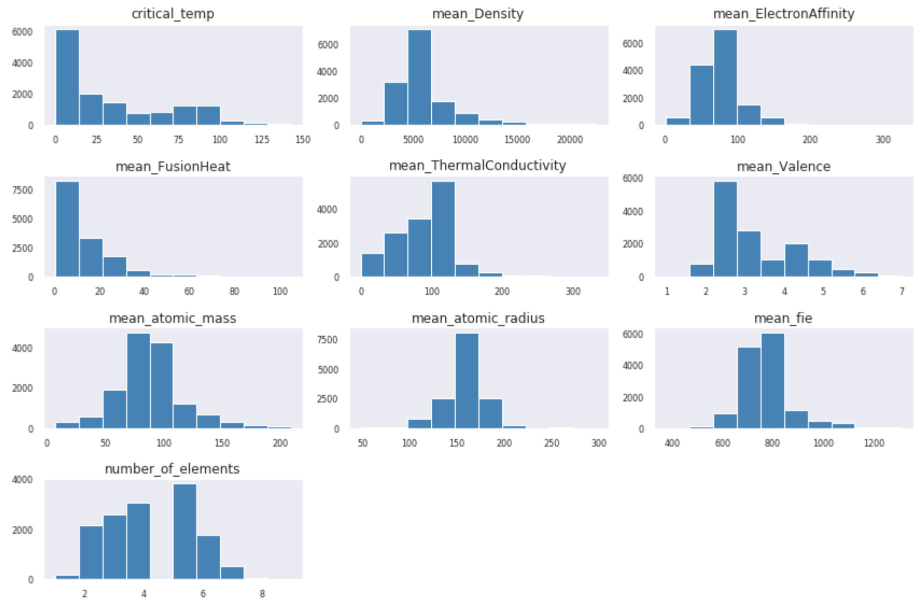
Before any analysis was performed, the data was split into train and test portions, with 0.33% of the data being reserved for testing to match that of [1], allowing comparability with [1]'s results.

The data was inspected for defects, such as missing, absurd or inconsistent values, but was found to be very clean already - this is the dataset cleaned for [1]'s analysis, so it was ready to be used for the same purposes.

3 Analysing and visualising the data

3.1 Univariate analysis

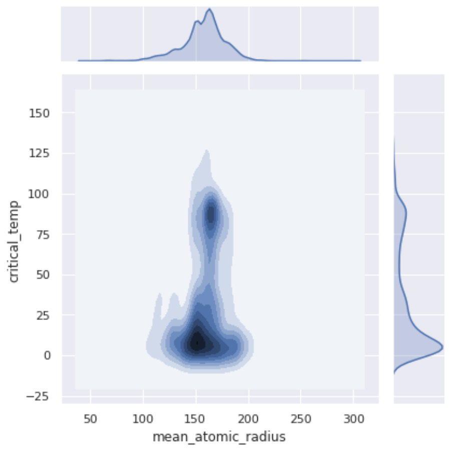
Decisions involved in the selection of a good regression model, features, and evaluative methods are highly data-dependent, and thus good understanding of the dataset is essential. I first inspected the inputs univariately, using histograms. There are a total of 167 inputs, so a selection are displayed in the figure below.



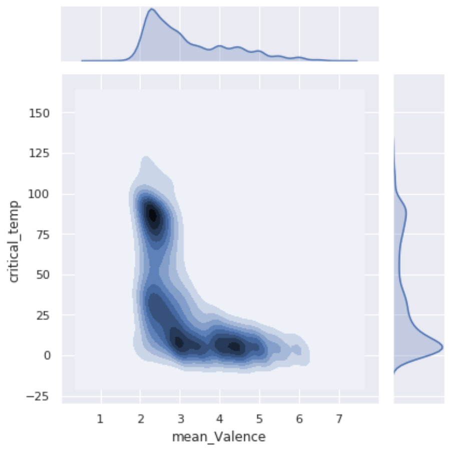
The inputs seem to follow a variety of distributions. Values in the dataset for critical temperature are most commonly sub-25K.

3.2 Joint analysis

Next, I plotted features against the critical temperature to visualise each relationship. Darker areas indicate a higher density of data points.



This figure shows that the range of mean atomic radius is greater among superconductors with a lower critical temperature. This may partly be explained by the fact that most superconductors in the dataset have low critical temperatures, as noted above.

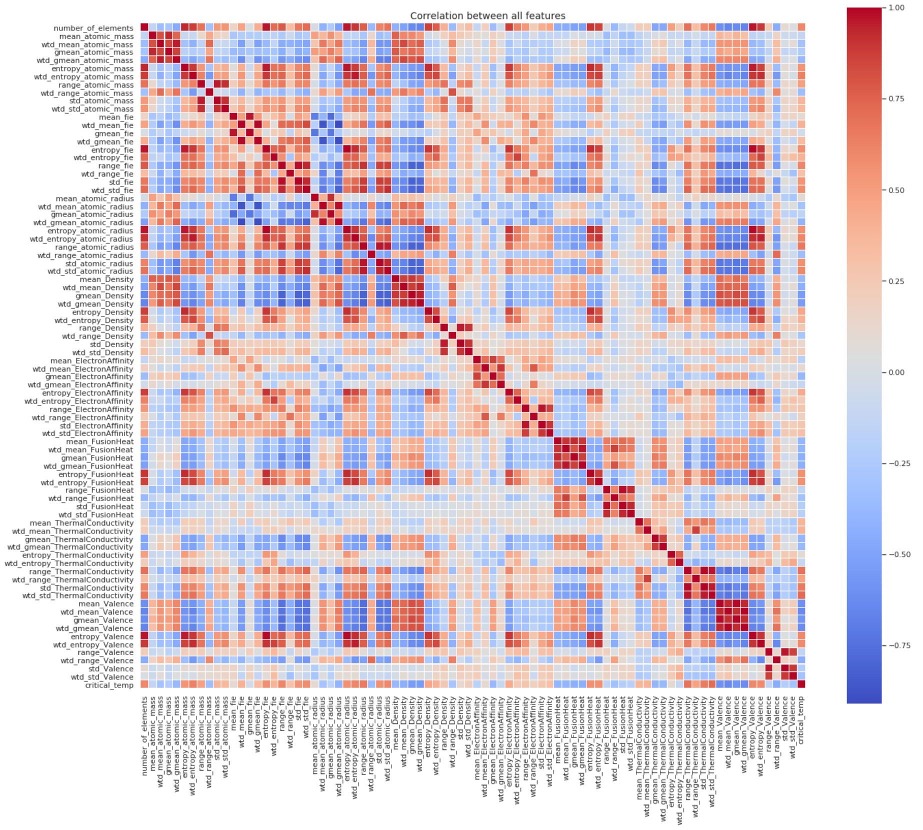


This figure suggests that the relationship between critical temperature and mean valence of a superconductor may be non-linear. This will be returned to in the discussion below of introducing polynomial terms into the input matrix (section 5.2.3).

3.3 Correlation

Mean absolute correlation between `train.csv` inputs and critical temperature is 0.39, a high fairly degree of correlation, suggesting that these will be fairly useful features for predicting critical temperature. Mean absolute correlation between elements inputs and critical temp is 0.078, so only a small improvement should be expected with the addition of the elements inputs as features.

The individual correlation of each input (excluding the elements inputs) with each other input, and the critical temperature, is shown in the heatmap below. Bright reds indicate high positive correlation, bright blues indicate high negative correlation, and greys indicate no correlation.



The diagonal line of 1.0 correlations indicates that each input is perfectly correlated with itself. Patterns of blocks of similar correlation emerge because different aggregations (such as mean or range) of a given measure (such as atomic radius or density) are adjacent. For example we see that, as expected, mean atomic mass is highly correlated with geometric mean atomic mass.

This visualisation shows some interesting properties - for example, there is a strong negative correlation between mean atomic radius and mean FIE (First Ionization Energy).

4 Model evaluation

Throughout the rest of the report, the effect of applying various techniques, such as regularisation or using different subsets of data, will be compared with not applying the techniques, to show the effect on performance. Performance is evaluated by comparing RMSE (root mean squared error), R^2 and median absolute error. RMSE and R^2 are useful general performance measures, and also provide direct comparability with the results reported in [1]. I include median absolute error as it gives an impression of how good the typical prediction is without being heavily weighted by small numbers of extremely incorrect predictions.

I use 5-fold cross validation to avoid scoring these measures on data used for training, while also not peeking at the test data by evaluating performance using it then using that information to adapt my training strategy.

5 Preparing the inputs and selecting features

5.1 Preparing the inputs

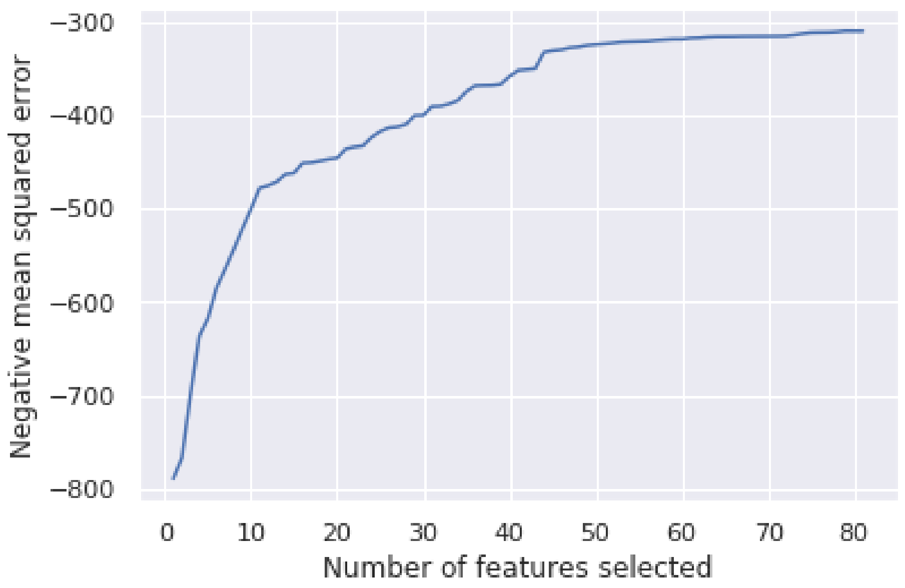
To prepare the inputs, I experimented with standardisation - transforming the data to centre it, by removing the mean value of each feature, then dividing each feature by its standard deviation. I expected this to have no effect on the error rate of a linear regression model, because unlike other estimators, it is not an assumption of linear regression that the data is distributed normally with a mean of zero and unit variance. This was confirmed: the root mean squared error (RMSE), R^2 and median absolute error were all identical for linear regressions on non-standardised and standardised data:

	RMSE	R^2	Median absolute error
Not standardised	17.61 ($\pm$0.40)	0.74 ($\pm$0.02)	10.19 ($\pm$0.29)
Standardised	17.61 ($\pm$0.40)	0.74 ($\pm$0.02)	10.19 ($\pm$0.29)

5.2 Feature selection

5.2.1 `train.csv` data

First, I used Recursive Feature Elimination to determine which features from `train.csv` do not improve the performance of the model. The optimal number of features was determined to be 79, a decrease of 2 from the full feature set.



The two dropped features were `wtd_mean_Density` and `wtd_range_Density`. This result is confirmed in two ways. Firstly, by inspecting the coefficients assigned to these two features in a linear regression trained on the full feature set. Amongst the five models trained in a 5-fold cross-validation run for standard linear regression with all `train.csv` features, the mean coefficient for `wtd_mean_Density` is 0.0003, and the mean coefficient for `wtd_range_Density` is -0.0002. These values are very close to zero, indicating that they play virtually no role in the prediction process. The second way the non-importance of these features is confirmed is by comparing the performance of a linear regression with all features in `train.csv` versus with all features except for these two. The results are almost identical, demonstrating that eliminating those features does not worsen performance:

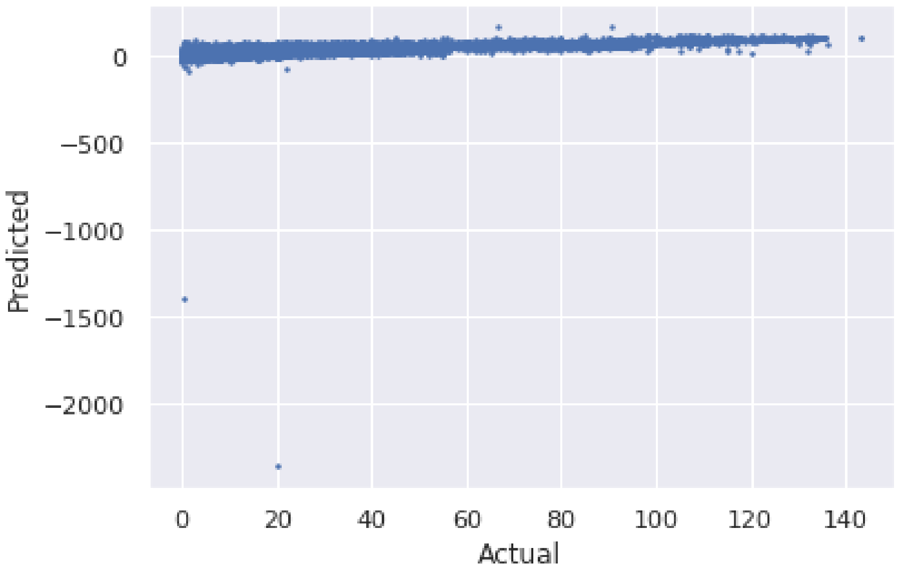
	RMSE	R^2	Median absolute error
Full feature set	17.61 ($\pm$0.40)	0.74 ($\pm$0.02)	10.19 ($\pm$0.29)
Reduced feature set	17.61 ($\pm$0.40)	0.74 ($\pm$0.02)	10.17 ($\pm$0.29)

5.2.2 Element data

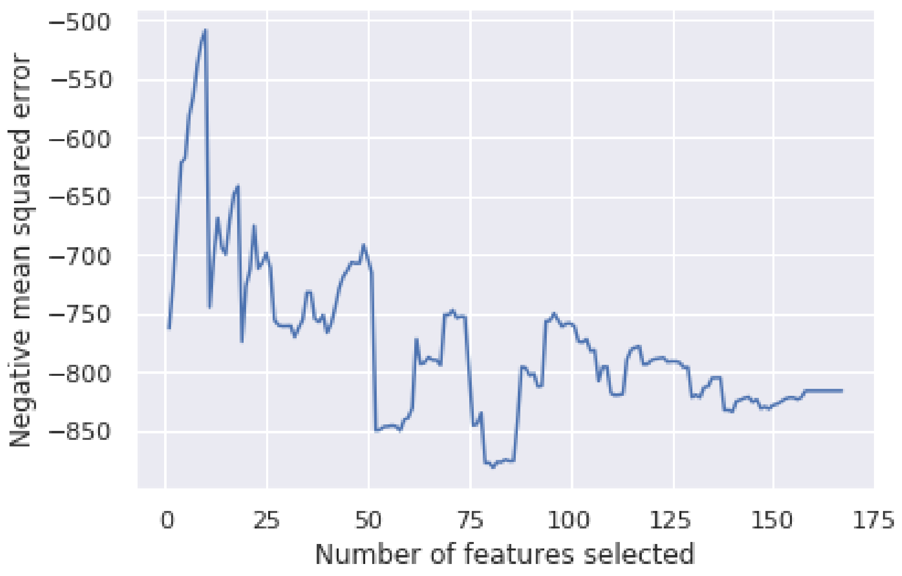
In [1], the set of features used is just those in `train.csv`, which I will refer to as the "general features". [1] does not use the elements data - data from `unique_m.csv` describing the number of elements present in the chemical formaulae, instead just using the aggregations that make up the general features. An obvious question, then, is how performance compares between using the general features, versus using all input data - the general features *and* the element data.

	RMSE	R^2	Median absolute error
General features	17.61 ($\pm$0.40)	0.74 ($\pm$0.02)	10.19 ($\pm$0.29)
General and element data	25.78 ($\pm$25.59)	0.31 ($\pm$1.32)	9.32 ($\pm$0.25)

The above table shows that adding all of the element data makes mean performance substantially worse, and with much higher variation between folds in the 5-fold cross validation. Meanwhile median performance is slightly improved. This seems to indicate that the model is sensitive to local trends in some parts of the data, which make performance much worse when key data points are not present in the training set. The effect of these disparities is shown in the graph below - note that there are a small number of extremely poor predictions:



I hypothesised that these outlier poor predictions could be eliminated by removing the element features that give rise to them. My initial attempt at removing these features was to apply the same `sklearn` recursive feature elimination (RFE) algorithm applied above. However, the results were very poor - the optimal number of features was found to be just 10:



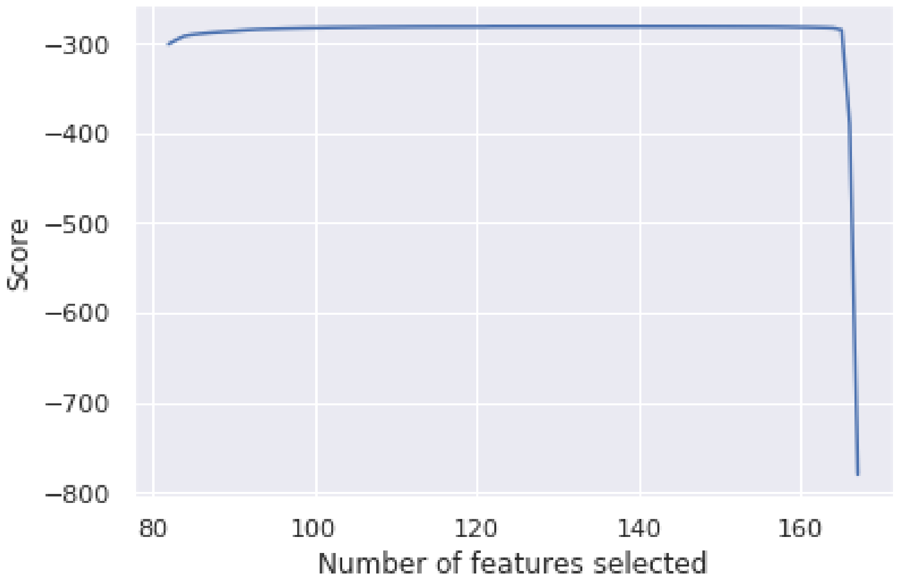
This is clearly not the actual optimal number of features, because running a linear regression with just those 10 features performs considerably worse than a linear regression with all `train.csv` features:

	RMSE	R^2	Median absolute error
All `train.csv` features	17.61 (±0.40)	0.74 (±0.02)	10.19 (±0.29)
10 selected features	22.79 (±0.33)	0.56 (±0.01)	15.24 (±0.53)

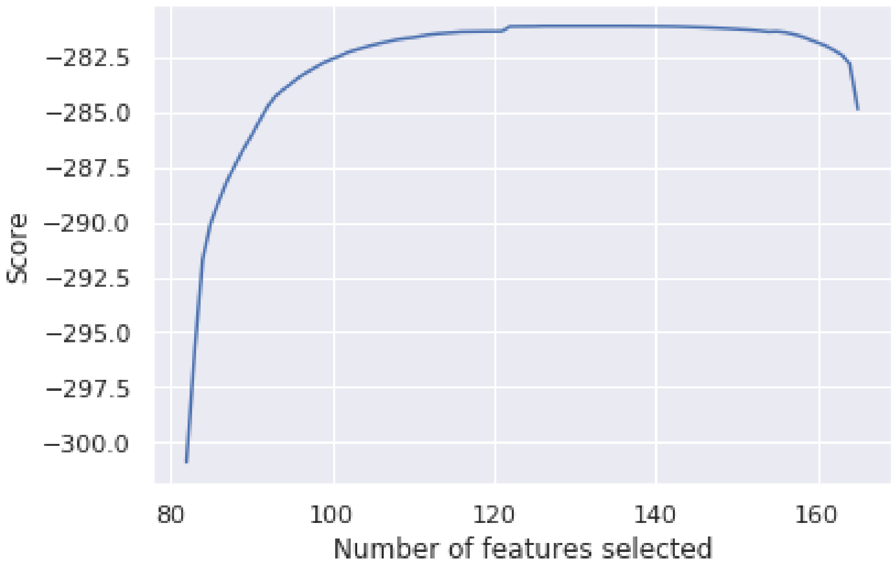
While it is unclear why the `sklearn` RFE algorithm performed poorly here, my own solution was more successful. I implemented an ”iterative feature addition” algorithm, which executed the following process:

1. Begin with the `train.csv` features
2. For each element column in `unique_m.csv` not yet in the feature set, try adding it to the feature set, and evaluate RMSE with cross validation
3. Set the new feature set to be the current feature set plus the element with which RMSE is best, and update the record of the best RMSE of all time
4. Repeat from (2) until all elements have been added
5. Return the best performing feature set

This process is more careful, as it begins with the solidly performing `train.csv` feature set, and slowly adds additional features, tracking the change in performance at each step. The optimal number of features is 130, with a sharp falloff as the last two features are added.



The above graph is reproduced below for greater interpretability - the below figure does not include the final two element columns added, which were responsible for the extreme decrease in score when the final two element columns were added (”Cd” and ”Pr”).



The resulting feature set, composed of 130 inputs (of all 81 `train.csv` features and 49 of the `unique_m.csv` inputs) performs substantially better than the original 81 `train.csv` features alone:

	RMSE	R^2	Median absolute error
81 `train.csv` features	17.61 (±0.40)	0.74 (±0.02)	10.19 (±0.29)
130 optimal features	16.76 (±0.42)	0.76 (±0.02)	9.36 (±0.29)

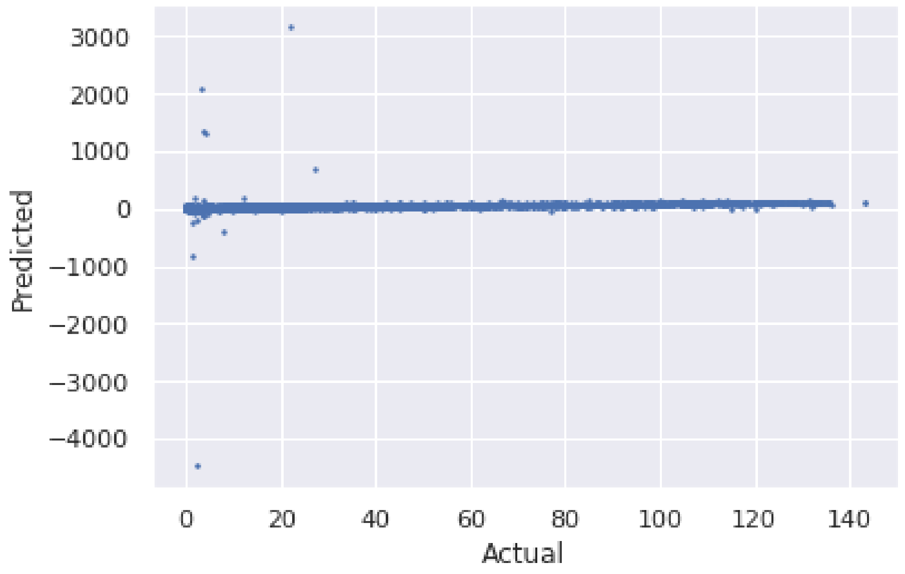
5.2.3 Polynomial terms

As discussed above in section 3.2 where mean valence was plotted against critical temperature, some of the features look like they would be best modelled with non-linear terms. To test this, I introduced polynomial features for every feature. The results for the `train.csv` features, with the addition of squared terms for each, show a strong improvement over `train.csv` features alone, and adding terms of degree 3 produces even better performance:

	Degree	RMSE	R^2	Median absolute error
`train.csv`	1	17.61 (±0.40)	0.74 (±0.02)	10.19 (±0.29)
`train.csv`	2	16.33 (±0.48)	0.77 (±0.02)	9.23 (±0.54)
`train.csv`	3	15.84 (±0.66)	0.79 (±0.03)	8.86 (±0.40)
Expanded	1	16.76 (±0.42)	0.76 (±0.02)	9.36 (±0.29)
Expanded	2	16.33 (±0.95)	0.77 (±0.04)	8.53 (±0.37)
Expanded	3	47.78 (±53.84)	-1.58 (±4.77)	8.30 (±0.44)

The bottom three "Expanded" rows in the above table show the effect of introducing polynomial terms for each feature in the optimal set of 130 features found in Section 5.2.2 using iterative feature addition. Interestingly, a modest improvement is found with the introduction of squared terms, bringing the performance to a level similar to that of `train.csv` features with squared terms. One possible explanation for this is that the additional information added by introducing the elements data actually improves performance because in important places it varies with the `train.csv` features in a non-linear fashion, which would explain why the introduction of directly non-linear terms does not greatly improve performance.

Introducing terms of up to degree 3 to the expanded 130 feature set produces a sharp downturn in performance, with a negative R^2 indicating it is worse than predicting the expected value for every input. The R^2 and root mean squared error seem to be very heavily impacted by a small number of very poor predictions - interestingly, the median absolute error is better than the other feature sets. These extremely erroneous predictions can be seen on following figure:



As a result, this regression model could prove most useful if the use case is that a human scientist would run it on an individual chemical formula and have a chance to visually inspect the result, and discard it if the predicted Kelvin value for critical temperature was extremely high or extremely low (for example, if it was the > 2000 K value predicted instead of 27 K on the graph above). Alternatively, if the results are generated at scale programatically, there could be manual checks for patently ridiculous Kelvin values that would flag these results for manual inspection.

However, if the model was used in a setting without human oversight or manual intervention, or one where very incorrect predictions would be a big issue, then the more reliable (lower RMSE) `train.csv` with polynomial terms up to degree 3 feature set would be preferable.

6 Selecting and training a regression model

The selected model was a linear regression model with polynomial features, as described in the section above. I focus on a linear regression to more deeply examine the effect of the various tecniques described above - eliminating features, using different subsets of data, introducing polynomial features, and standardisation.

I experimented with `sklearn`'s `Ridge` algorithm for regularisation (using cross validation for parameter optimisation), but performance was similar to without regularisation for the `train.csv` feature set, and extremely poor for the feature set with polynomial terms up to degree 3:

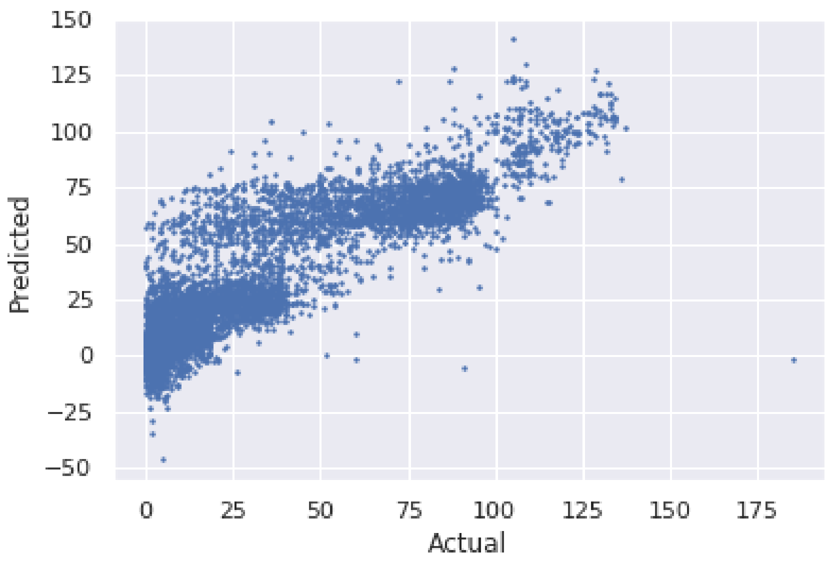
	Degree	RMSE	R^2	Median absolute error
No regularisation	1	17.61 ($\pm$0.40)	0.74 ($\pm$0.02)	10.19 ($\pm$0.29)
`RidgeCV`	1	17.61 ($\pm$0.41)	0.74 ($\pm$0.02)	10.19 ($\pm$0.29)
No regularisation	3	15.84 ($\pm$0.66)	0.79 ($\pm$0.03)	8.86 ($\pm$0.40)
`RidgeCV`	1	3.7×10^{17}	-2.0×10^{28}	1.3×10^{16}

NB: the standard deviation is not listed for the bottom row for ease of typesetting, but suffice to say they are extremely large values.

Due to this poor performance of `Ridge` regularisation, I selected a non-regularised linear regression model.

7 Final results

The final model selected is a linear regression model with squared and cubed polynomial terms introduced, with the feature set of `train.csv`. Actual vs predicted values in the test set are plotted below.



The final evaluated performance with the test set is shown below, presented alongside the cross-validated performance in the training set for comparison.

	RMSE	R^2	Median absolute error
Test set	15.73	0.73	8.76
CV training set	15.84 ($\pm$0.66)	0.79 ($\pm$0.03)	8.86 ($\pm$0.40)
[1]'s multiple regression	17.6	0.74	-
[1]'s XGBoost	9.5	0.92	-

The results are similar, with a minor improvement in RMSE and a minor decrease in R^2 performance.

The bottom two rows of the above table compares this result to performance achieved by the author of [1] (median absolute error is not reported). My linear regression model outperforms their simple multiple regression (intended as a benchmark performance model) by incorporating polynomial terms to better fit the non-linear relationship between some features and the critical temperature.

As expected, [1]'s more advanced gradient boosted XGBoost model significantly outperforms both, using an ensemble of trees to account for points which are hard to predict - which neatly solves the issues I uncovered using different feature sets for linear regression, such as the expanded feature set including element data and polynomial terms, in section 5.2.3.

A next step for research on this problem would involve applying the findings in this report to more advanced machine learning methods, such as XGBoost, examining how regularisation, standardisation, different feature sets, polynomial terms, and different loss functions can be employed to improve the prediction of superconductor critical temperatures from chemical formulae.

8 Works cited

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