A. Introduction

Superconducting materials have widespread applications with the potential to revolutionise the energy industry with frictionless superconducting wires and to deliver electricity with no energy loss. However, superconductors are only able to conduct current with zero resistance at their critical temperature (T_c). Thus this study aims to predict the superconducting T_c of superconductors, building on top of previous work done by (Hamidieh, 2018; Li et al., 2020; Owolabi et al., 2014, 2015; Stanev et al., 2018).

The superconductor data comes from the UCI Machine Learning Repository at https://archive.ics.uci.edu/ml/datasets/Superconductivty+Data. All features were used to make predictions and explanations for this are later mentioned in the report. This study investigates a novel approach towards predicting the T^2 of superconductors by fine-tuning several regressors to end up with a final stacked regressor. The stacked regressor was able to make predictions on the test set with an RMSE of 10.81 and R^2 of 0.9.

B. Method

B.1 Data Preparation

Loading and cleaning the dataset

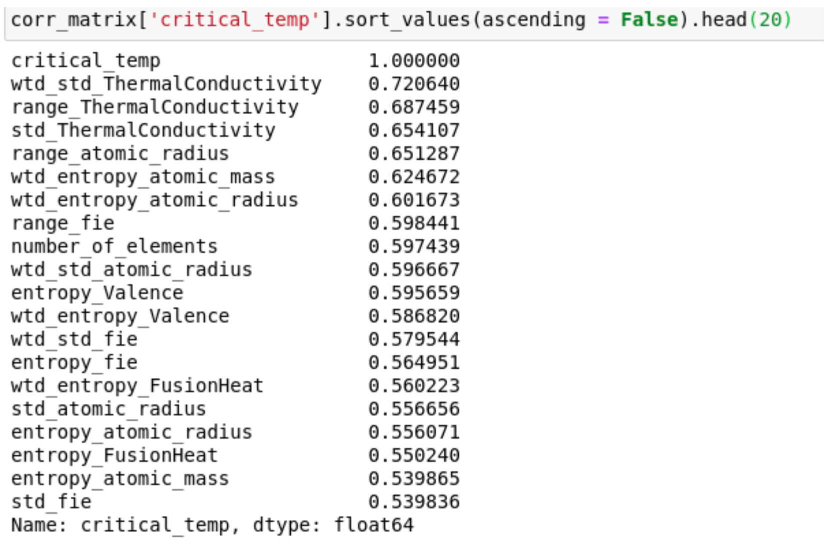
The dataset had 21263 entries with a total of 82 columns. Most of the data types in each column belonged to the floating-point data type, while "range_valence", "range_atomic_radius" and "number_of_elements" columns were integers. I converted the "range_valence" and "range_atomic_radius" columns to floating points for consistency and left the "number of elements" column as is.

There were no null values; however, the data had 66 duplicate rows. Duplicate values tend to be an extreme case of non-random sampling and could bias the fitted the model. Including them could increase the chances of the model overfitting this subset of points. Moreover, duplicate entries could appear both in the test and training set after a randomised test-train split, thus hampering the integrity of the findings. Based on the above reasons, the duplicates were removed, creating a cleaned data set with 21132 entries.

Train/Test Split

After cleaning the data, the data was then split between the test and training set. Given that the cleaned data set had over 20000 entries, splitting the data with an 80/20 split was seen as a good starting point, as 4000 examples can represent most of the variance in the data. The dataset was split using sklearn's train_test split option; hence data was split in a randomised fashion without the need for stratification as the output had no class labels.

B.2 Analysing And Visualising The Data



```
corr_matrix['critical_temp'].sort_values(ascending = False).head(20)
```

```
critical_temp                  1.000000
wtd_std_ThermalConductivity    0.720640
range_ThermalConductivity      0.687459
std_ThermalConductivity        0.654107
range_atomic_radius            0.651287
wtd_entropy_atomic_mass        0.624672
wtd_entropy_atomic_radius      0.601673
range_fie                      0.598441
number_of_elements             0.597439
wtd_std_atomic_radius          0.596667
entropy_Valence                0.595659
wtd_entropy_Valence            0.586820
wtd_std_fie                    0.579544
entropy_fie                    0.564951
wtd_entropy_FusionHeat         0.560223
std_atomic_radius              0.556656
entropy_atomic_radius          0.556071
entropy_FusionHeat             0.550240
entropy_atomic_mass            0.539865
std_fie                        0.539836
Name: critical_temp, dtype: float64
```

Figure 1: *Top 20 features that correlate with T_c according to the standard correlation coefficient*

Given that there are a large number of features, to visualise the data, a subset of features was chosen based on the standard correlation coefficient (also called the Pearson's R) between the T_c and every other feature **(see Figure 1)**. Interestingly, thermal conductivity, atomic radius, valence, electron affinity and atomic mass were noted as the most important features for predicting a materials T_c by the (Hamidieh, 2018) study. All except, electron affinity appear in the top 20, suggesting that electron affinity may not be amongst the top features that correlate with the critical temperature.

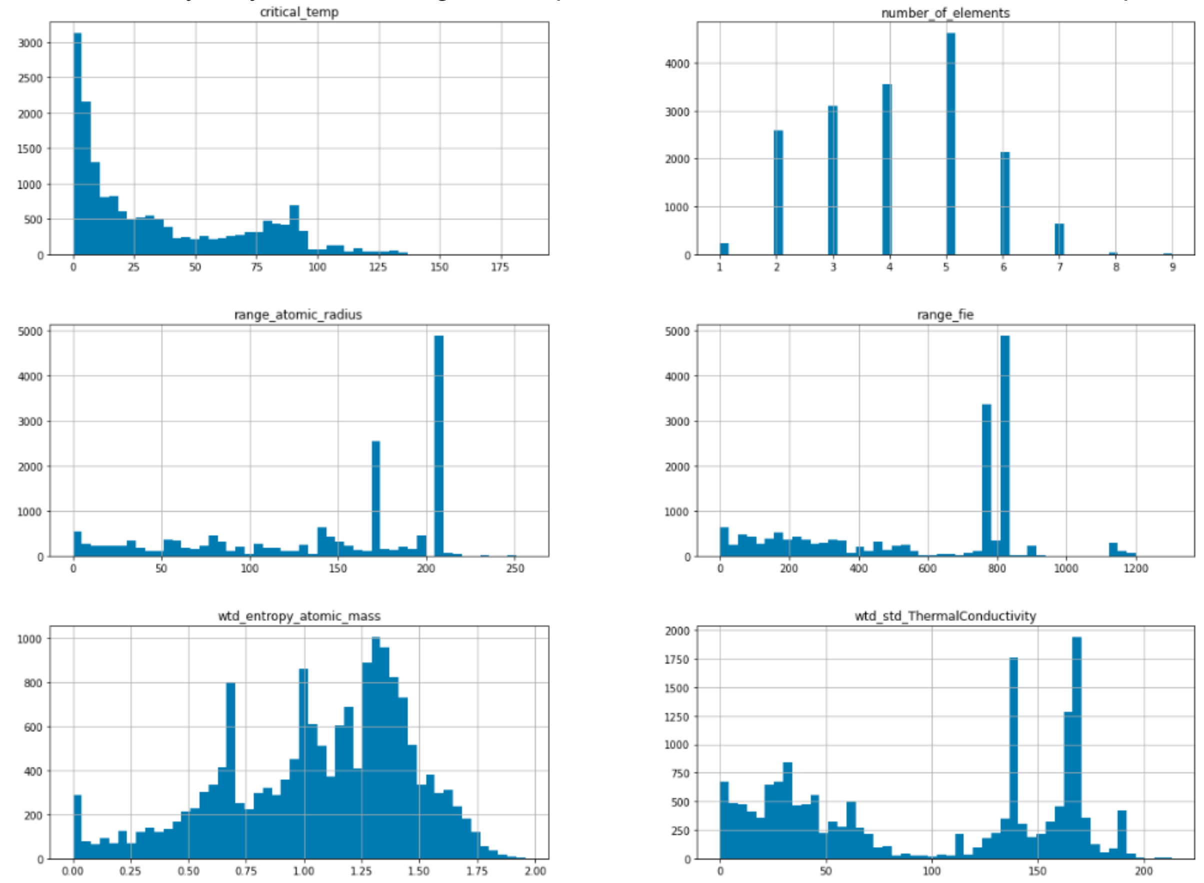


Figure 2: *Frequency distribution of T_c and 5 top 10 highly correlating features.*

Next, five distinct features out of the top 10 were chosen to understand their frequency distribution in the training set **(see Figure 2)**. It is evident that the majority of the entries in the training set have a T_c between near absolute zero (0 degrees kelvin) and liquid nitrogen temperatures (77k) which corresponds to temperature range needed for superconductivity (Hamidieh, 2018). Moreover, only the 'wtd_entropy_atomic_mass' and 'number_of_elements' attributes appear to have bell-shaped/Gaussian distributions which are usually assumed by parametric Least Squares based regression models. Hence, the data was standardised for Ridge Regression because if a feature has a variance that is orders of magnitude larger than others, it might dominate the objective function and make the regressor unable to learn from other features correctly as expected.

B.3 Preparing The Inputs And Choosing Suitable features

Each of the features except the number of elements **(see Table 1)** is a composite of a property of an element used to predict the critical temperature and a feature extracted from the chemical's formula. In addition to 'number of elements', the dataset includes the product of 8 element properties and ten features – 8 x 10 = 80 features.

Variable	Units	Description
Atomic Mass	Atomic mass units (AMU)	Total proton and neutron rest masses
First Ionization Energy	Kilo-Joules per mole (kJ/mol)	Energy required to remove a valence electron
Atomic Radius	Picometer (pm)	Calculated atomic radius
Density	Kilograms per meters cubed (kg/m^3)	Density at standard temperature and pressure
Electron Affinity	Kilo-Joules per mole (kJ/mol)	Energy required to add an electron to a neutral atom
Fusion Heat	Kilo-Joules per mole (kJ/mol)	Energy to change from solid to liquid without temperature change
Thermal Conductivity	Watts per meter-Kelvin (W/(m K))	Thermal conductivity coefficient κ
Valence	No units	Typical number of chemical bonds formed by the element

Table 1: *Table showing the properties of an element which are used for creating features to predict $T_{c.}$ Image is taken from:*

After understanding how the features in the dataset were created, I further investigated the correlation between the features to visually understand their relationship with T_c and each other. This was done using a confusion matrix, where I took the top correlating features that represented a distinct property of an element used to predict $T_{c.}$ **(see Figure 3).**

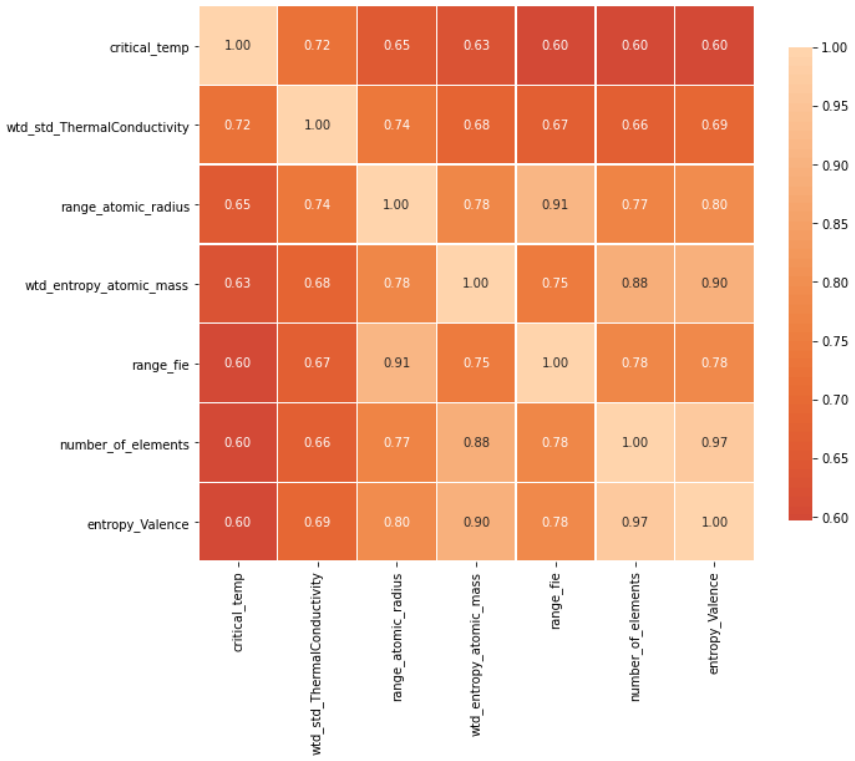


Figure 3: *Correlation matrix of T_c and top correlating features.*

The confusion matrix reveals a few things. All the top features have a highly positive correlation with one another (>0.65) suggesting that if these subsets of features were chosen, the performance of the model would be impacted by a problem called multicollinearity. Multicollinearity happens when one predictor variable in a multiple regression model can be linearly predicted from the others with a high degree of accuracy (Alin, 2010). This can lead to skewed or misleading results. This is because, with strong correlations, it becomes difficult for the model to estimate the relationship between each independent variable and the dependent variable independently because the independent variables tend to change in unison. Interestingly, decision trees and boosted trees algorithms are immune to multicollinearity by nature**.** When they decide to split, the tree will choose only one of the perfectly correlated features. However, other algorithms, like Logistic Regression or Linear Regression, are not immune to this.

Thus to avoid the issue of multicollinearity, all features were initially chosen for training all the models.

B.4. Selecting and training a regression model

a) Initial model selection

Linear Regression (Multiple Regression)

After investigating the correlation between the features and $T_{c,}$ I hypothesised that there might be a linear relationship between the features and $T_{c.}$, hence a Linear Regression model was chosen to make predictions.

Random Forests

A Random Forest predictor was also chosen in order to have a predictor that could avoid the potential problem of multicollinearity if a smaller subset of features were chosen. Moreover, a Random Forest is an example of Ensemble learning, where a group of Decision Trees are trained on a random subset of the training set (Farnham et al., 2019). The Random Forest algorithm introduces extra randomness when growing trees by searching for the best feature among a random subset of features as opposed to searching for the very best feature when splitting a node (Ho, 1995). This results in a greater tree diversity, which trades a higher bias for a lower variance, generally yielding an overall better model.

b) Training the models

After training both the models, the multiple regression model had an RMSE of 17.69 and an R^2 score of 0.73. The random forest performed slightly worse with an RMSE of 19.808 and an R^2 score of 0.668. **Figure 4** indicates that the multiple regression model under predicts T_c for high-temperature superconductors since many predicted points are below the line for the high-temperature superconductors. The model over-predicts for low-temperature superconductors.

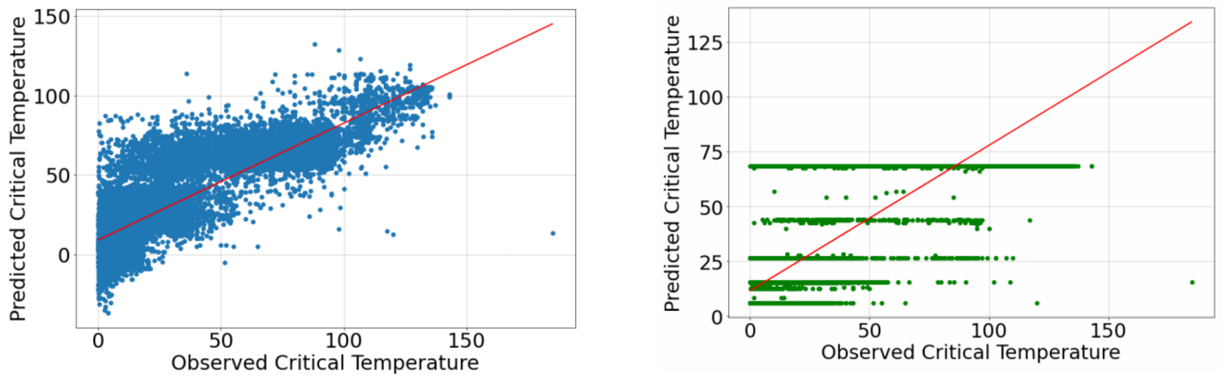


Figure 4: *These plots show the observed superconducting $T_{c.}$ Vs the predicting superconducting* T_c based on the multiple regression model (blue) and the random forest model (green).

Conversely, the Random Forest under predicts for high temperatures but similarly over predicts for low temperatures.

c) *K*-fold Cross-validation

To better evaluate the performance of both models, K-fold cross-validation was implemented. This involved randomly splitting the training set into ten distinct subsets called folds and then training and evaluating the model's performance ten times, picking a different fold for evaluation every time and training on the other nine folds. After cross-validation, the multiple regression model had a mean RMSE of 19.73 with a standard deviation of 0.6, whilst the random forest still performed worse with a mean of RMSE of 19.85. Interestingly the mean score on the training set for both models is slightly lower than on the validation sets, hinting that the models may be overfitting the training set. Solutions to addressing overfitting are simplifying the model, regularising the model, or working on a larger training set

d) Fine-tuning the models

This section focuses on a few fine-tuning strategies that were implemented to help improve the models' performance.

Grid search on Random Forest
When the Random Forest model was trained initially, its max depth was set at two whilst the rest of the parameters were set at Sklearn's default values. To improve the performance of the Random Forest model, I implemented a grid search with fivefold cross-validation to test different combinations of values for the parameters max features, number of estimators and max depth. The grid search was limited to these features due to time constraints.

The max depth represents the depth of each tree in the Random Forest. The deeper the tree, the more splits it has to capture more information about the data (Segal, 2004). The number of estimator's parameter represents the number of trees in the forest. Usually, the higher the number of trees, the better the model can learn from the data (Probst & Boulesteix, 2017). However, adding a lot of trees can be computationally expensive; hence performing a parameter search is useful for finding the optimal number. Lastly, the max features parameter looks for the number of features to consider when looking for the best split (Shreyas et al., 2016). I fitted each model with max depth values ranging from 2 – 15, max features ranging from 2-8 and number of estimators ranging from 3 – 30. As a result, 48 Random forest model combinations were generated.

The results of the grid search were enlightening. Indeed, there was a strong positive correlation between max depth and RMSE as the top 10 models all had max depth values of 15, confirming that deeper trees do help the model learn better. Similarly, with number of estimators, the majority of the top 10 models had either 10 or 30 estimators. Less so for max features as models that only considered two features when looking for the best split still performed well.

Importantly, the top-performing model had a mean RMSE on the validation set of 10.03 and a mean RMSE on the training set of 6.58, thus performing far better than the previously trained model. However, the top model had the highest difference in RMSE between the validation and training sets, suggesting that the model is greatly overfitting; therefore, it was not chosen. I decided to pick a model out of the 48 that had a much lower difference between the mean RMSE on the validation and training set but still performed strongly. The model chosen had a max depth of 10, max features of 4 and max number of estimators of 10 with a validation set RMSE of 12.50 and training set RMSE of 11.2.

Regularisation on Linear Regression

Regularisation is a good way to reduce overfitting of the model as the fewer degrees of freedom the model has, the harder it will be to overfit the data (Farnham et al., 2019)**.** For regularising the Linear Model, Ridge Regression and Lasso Regression were investigated.

Ridge Regression is a regularised version of Linear Regression where a regularisation term equal is added to the cost function **(see Figure 5 below)**. This forces the learning algorithm to not only fit the data but also keep the model weights as small as possible (Farnham et al., 2019)**.** The hyperparameter α controls how much the model is regularised. If α = 0 then Ridge Regression is just Linear Regression. If α is very large, then all weights end up very close to zero, and the result is a flat line going through the data's mean. As a result, I implemented grid search with fivefold cross-validation to tune the α hyperparameter with values ranging between 0.2 – 10. Before doing so, the data were scaled using Sklearn's StandardScaler as Ridge Regression is sensitive to the scale of the input features (Farnham et al., 2019). The top-performing model had a mean RMSE of 17.87 on the validation set and a mean RMSE of 17.76 on the training set.

$$J(\boldsymbol{\theta}) = \mathrm{MSE}(\boldsymbol{\theta}) + \alpha\frac{1}{2}\Sigma_{i=1}^{n}\theta_i^2$$

Figure 5: *Ridge Regression cost function*

Lasso (Least Absolute Shrinkage and Selection Operator) regression is a type of linear regression that uses shrinkage, where data values are shrunk towards a central point like the mean (Farnham et al., 2019). Lasso regression performs L1 regularisation, which adds a penalty equal to the absolute value of the magnitude of coefficients. Larger penalties result in coefficient values closer to zero, which is the ideal for producing simpler models. On the other hand, L2 regularisation (e.g. Ridge regression) doesn't result in the elimination of coefficients (Farnham et al., 2019). Grid search with fivefold cross-validation was implemented on Lasso's α hyperparameter with α values ranging between 0.2 – 10. The top-performing model had an RMSE of 18.4 on the validation set and 18.4 on the training set.

$$J(\boldsymbol{\theta}) = \mathrm{MSE}(\boldsymbol{\theta}) + \alpha\Sigma_{i=1}^{n}\left|\theta_i\right|$$

Figure 6: *Lasso Regression cost function*

Interestingly, the multiple linear regression model outperformed both Lasso and Ridge Regression on the validation set, suggesting that the linear model may not be overfitting.

Stacking

The last fine-tuning method that was implemented was stacking, which involved blending the performance of the previously trained regressors. I decided to remove Lasso Regression from the stack as it performed with worst, leaving Linear Regression, Ridge Regression and Random Forests in the stack. My strategy was to fit the regressors individually on the training data while a final regressor was trained using the stacked predictions of these base regressors. I chose this strategy because it can be tedious to find the model which will best perform on a given dataset. Stacking provides an alternative by combining the outputs of several learners, without the need to choose a model specifically. Interestingly, the stacking regressor outperformed the Random Forest regressor slightly with an RMSE of 11.79 but had an equal R^2. As we can see **(see Figure 6)**, the stacked regressor combined the strengths of the different regressors; however, training the stacked regressor can be computationally expensive. Nonetheless, given that the stacked regressor outperformed the rest, it will be chosen as the final model to predict on the test set.

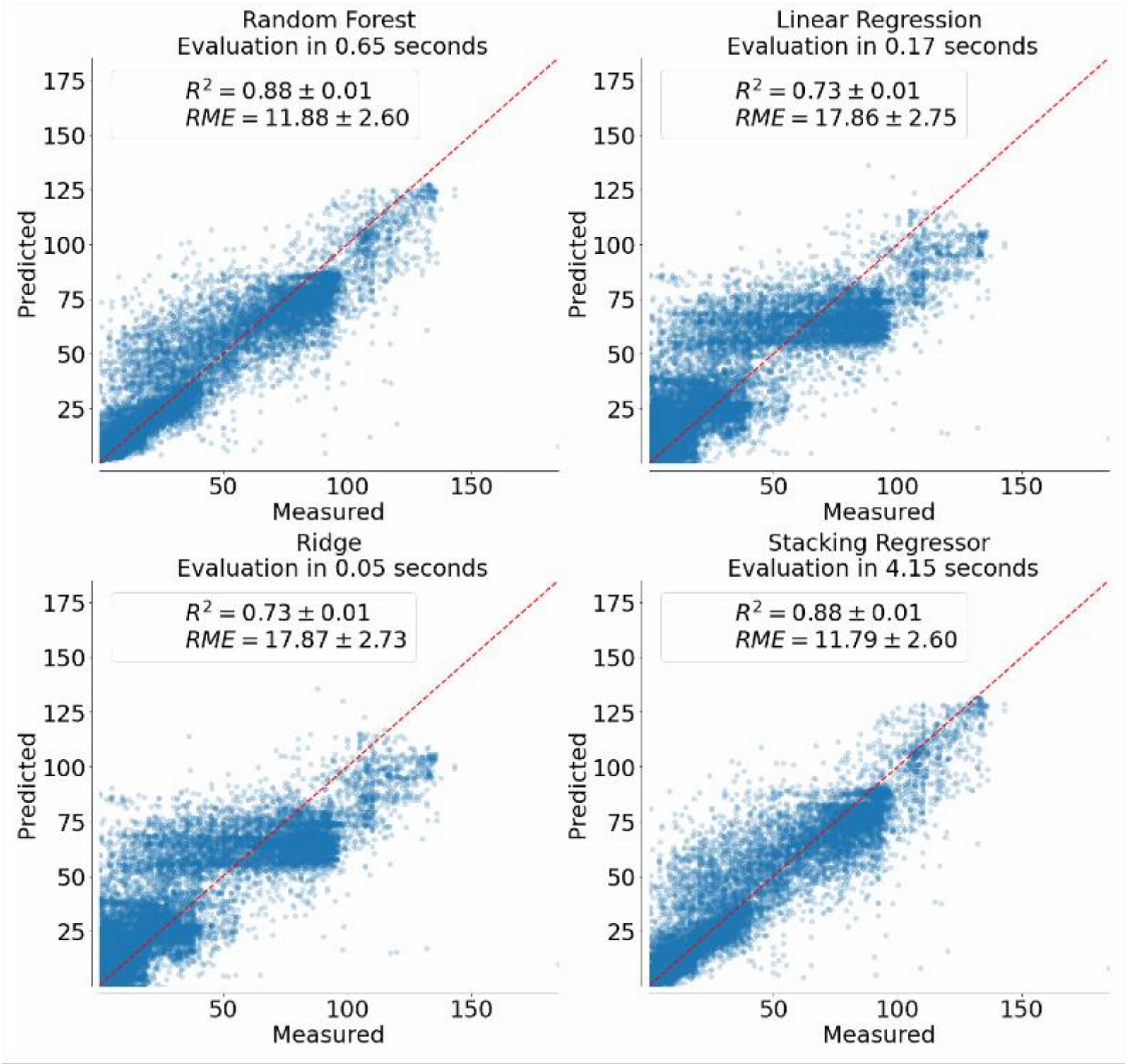


Figure 6: *Plot showing the performance the single regressors vs the stacked regressor*

C Evaluation on the test set

After tuning all the models on the training and validation sets, the stacked regressor was found to be the top-performing model. It was evaluated on the test set, and it achieved an RMSE of 10.81 and R^2 of 0.9. These results show that despite the fine-tuning to perform on the validation dataset, the model was still able to generalise well on unseen data. Moreover, from **Figure 7** it appears that the model under predicts for superconductors with both low and high $T_{c,}$ which is the opposite of what was seen in the Linear Regression model.

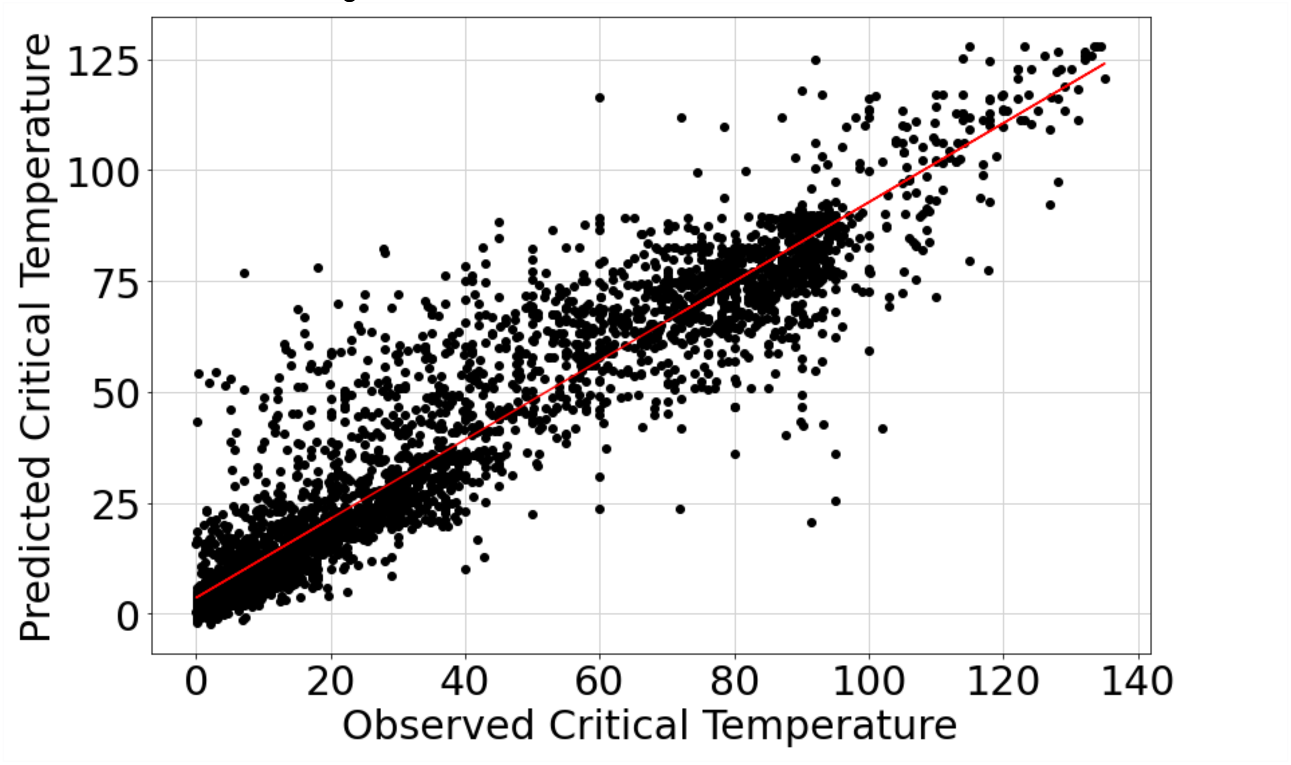


Figure 7: *These plots show the observed superconducting $T_{c.}$ Vs the predicting superconducting T_c based on the stacked regressor model. The model achieved an RMSE of 10.81 and R^2 of 0.9.*

D Conclusion

This is study has shown that a stacked approach using only the superconductor's chemical formula can predict T_c at a decent level. It was evident that the Random Forest regressor was the strongest contributor to the stacked approach; hence, future research would involve extended hyperparameter tuning that searches for the optimal max depth as increases in max depth were shown to improve RMSE. Lastly, the results from this study outperform a previous study done by (Stanev et al., 2018) who also used Random Forests to predict T_c and achieved an R^2 of 0.88, thus suggesting that stacking Random Forests with other linear models is an effective approach to predicting $T_{c.}$

References:

1. Alin, A. (2010). Multicollinearity. *Wiley Interdisciplinary Reviews: Computational Statistics*, *2*(3), 370–374.
2. Farnham, B., Tokyo, S., Boston, B., Sebastopol, F., & Beijing, T. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems SECOND EDITION*.

3. Hamidieh, K. (2018). A data-driven statistical model for predicting the critical temperature of a superconductor. *Computational Materials Science*, *154*, 346–354. https://doi.org/10.1016/j.commatsci.2018.07.052
4. Ho, T. K. (1995). Random decision forests. *Proceedings of 3rd International Conference on Document Analysis and Recognition*, *1*, 278–282.
5. Li, S., Dan, Y., Li, X., Hu, T., Dong, R., Cao, Z., & Hu, J. (2020). Critical Temperature Prediction of Superconductors Based on Atomic Vectors and Deep Learning. *Symmetry*, *12*(2), 262.
6. Owolabi, T. O., Akande, K. O., & Olatunji, S. O. (2014). Prediction of superconducting transition temperatures for Fe-based superconductors using support vector machine. *Adv. Phys. Theor. Appl*, *35*, 12–26.
7. Owolabi, T. O., Akande, K. O., & Olatunji, S. O. (2015). Estimation of superconducting transition temperature T C for superconductors of the doped MgB 2 system from the crystal lattice parameters using support vector regression. *Journal of Superconductivity and Novel Magnetism*, *28*(1), 75–81.
8. Probst, P., & Boulesteix, A.-L. (2017). To tune or not to tune the number of trees in random forest. *The Journal of Machine Learning Research*, *18*(1), 6673–6690.
9. Segal, M. R. (2004). *Machine learning benchmarks and random forest regression*.
10. Shreyas, R., Akshata, D. M., Mahanand, B. S., Shagun, B., & Abhishek, C. M. (2016). Predicting popularity of online articles using random forest regression. *2016 Second International Conference on Cognitive Computing and Information Processing (CCIP)*, 1–5.
11. Stanev, V., Oses, C., Kusne, A. G., Rodriguez, E., Paglione, J., Curtarolo, S., & Takeuchi, I. (2018). Machine learning modeling of superconducting critical temperature. *Npj Computational Materials*, *4*(1). https://doi.org/10.1038/s41524-018-0085-8