Assignment 2 – Regression using Scikit-learn

Student Name: Joey Galvin **Student ID**: 22348861 **Programme**: 4BCT

Github Link: https://github.com/Joey-2134/MachineLearningAssignment2

Algorithm 1 – Random Forest Regressor

Random Forest algorithms are essentially an average of multiple decision trees with some other minor differences. Firstly when making a decision tree, we first need to bootstrap the data. This means we take a random sub sample of the entire dataset while allowing rows which have already been selected to be used again. The next step is to select our features for the root node. For random forest tree's, we randomly select a subset of the features to use. We then build our decision tree as usual with these features. When we want to regress/classify an new sample, we run it through every tree and get the average result/majority class. In this case we are doing regression so our leaf nodes will be numerical instead of a class.

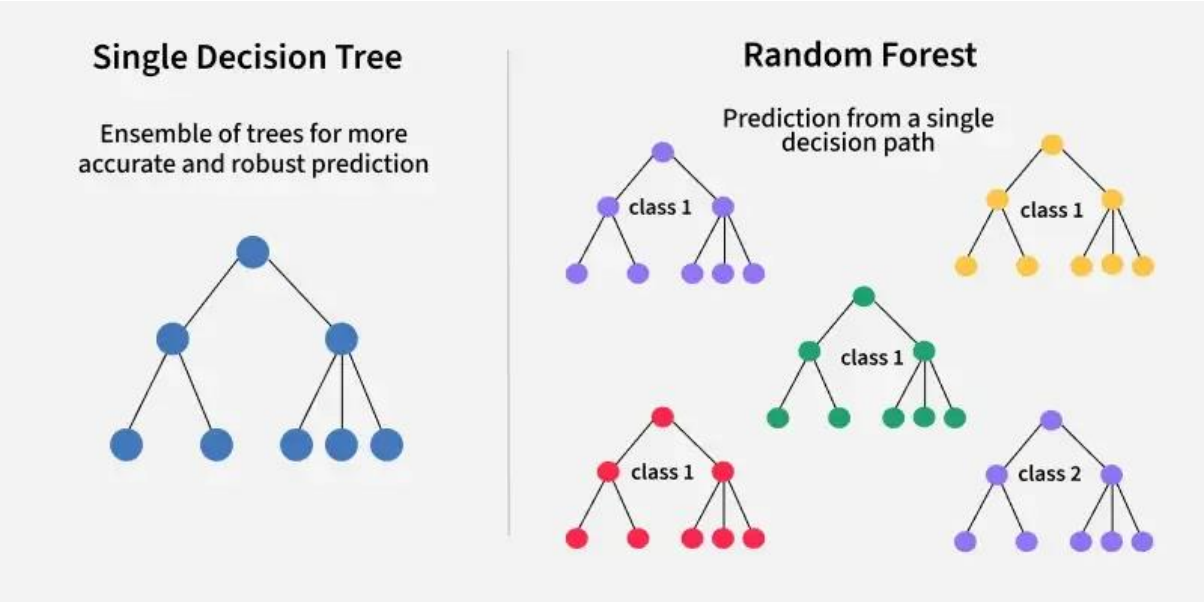


Figure 1: Diagram showing multiple decision trees

Why I choose this algorithm.

I chose this particular algorithm is mostly because I wanted to learn about decision trees and random forest algorithms since I hadn't yet. The algorithm should work perfect for this assignment and has the appropriate number of hyper paramaters to tune.

Hyperparameter Details for Tuning.

n_estimators: This is simply the number of trees to construct to make up the forest. If calculation performance is not a factor, the more the merrier.

max_depth: This is the maximum amount of layers a tree can be comprised of. Too many layers can lead to overfitting and too little can lead to underfitting.

Algorithm 2 – Gradient Boosting Regressor

Gradient Boosting Regressor is a similiar but much more complex algorithm than random forest. It is similiar in the sene they both make use of decision trees. However in random forest, we make a forest of trees using a degree of randomization as discussed above while gradient boosting uses the mistakes of previous trees to learn how to improve the next. At the end of random forest we take the average result of the trees while with Gradient boosting, certain trees have more say than others which is factored in. It is named gradient boosting due to how it reduces error each time similiar to gradient descent optimisation.

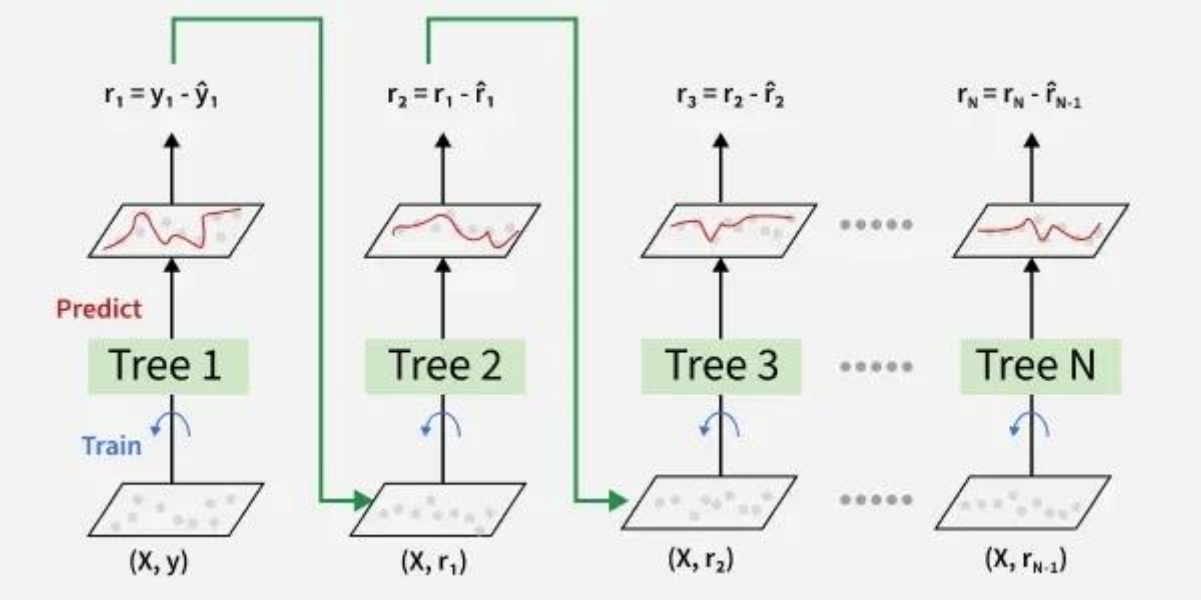


Figure 2: Graphic illustration of how the algorithm works

Why I choose this algorithm.

This is an algorithm I was planning on using as part of my final year project regardless so I was interested in learning more about it. The algorithm is clearly a more robust algorithm than random forest and hopefully should get better results than with random forest.

Hyperparameter Details for Tuning.

n_estimators: This is simply the number of trees to construct to make up the forest. If calculation performance is not a factor, the more the merrier.

max_depth: This is the maximum amount of layers a tree can be comprised of. Gradient Boosting typically has much shallower trees so might use a different parameter if I need to.

Algorithm 1 - Random Forest Regressor - Model Training and Evaluation

In this assignment, the metrics I will be focusing on are R^2 and MAE. R2 measures how much the model describes the target feature. For example, a R2 value of 0.98 means our model features explains 98% of the variance in the predictions. So the missing 2% is unnaccounted for. This could be due to missing features, bad data, noise etc.

MAE (Mean Absolute Error) to put very simply is on average how much is our prediction off by. The unit will be the same as we are trying to predict. In this case its the tensile strength of steel. Typically we expect to see values between 100 and 200. It is calculated by summing up the differences between the predicted value and actual values and then dividing by the amount of predictions made.

Training and Evaluation Details

I will be using the same graphs and data display for showing my results for both algorithms so they can be easily compared. 10 Fold cross validation is used for a more accurate model test results by computing the average across 10 different test/train splits.

Below is the graph of my first run results using default hyper parameters for the Random Forest Regressor.

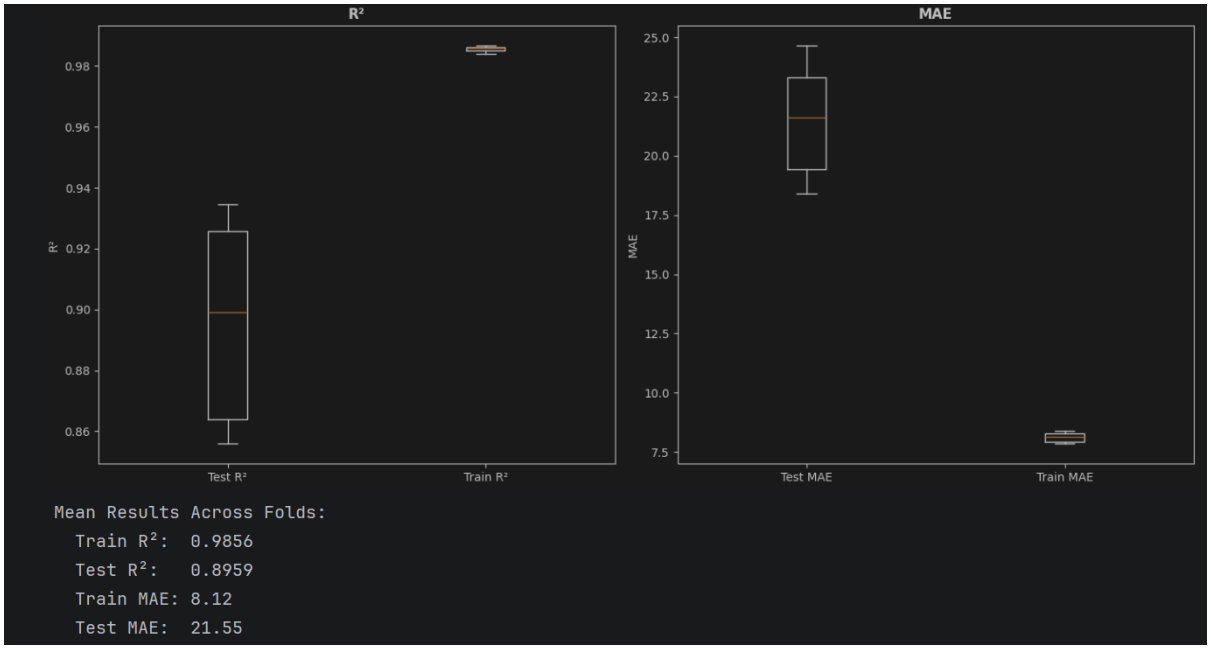


Figure 3: First run results

R2:

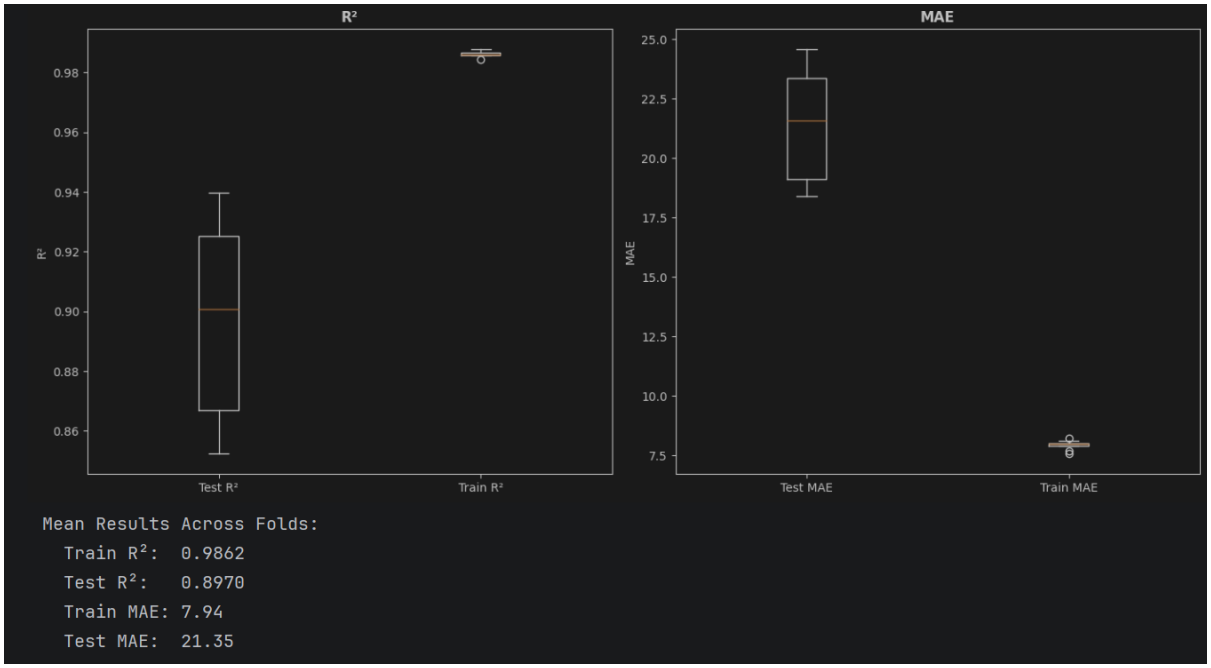
The results here show slight overfitting, on the training data the model shows an R2 value of .98, this means that the model has very little error when compared to the test datasets. Which had an average error of .89 which while is still impressive, the gap between the the the test and training datasets signifies slight overfitting.

MAE:

The mean absolute error results show the same pattern as R2, however this represents why R2 can be more easily interpretable result as no prior knowledge of the domain is needed. For example on the train set there is a mean MAE of 8.12, without knowing the average tensile strength of steel I have no idea if this result is good. However by quickly scanning some entries in the dataset I think its a decent result. The Test MAE was significantly higher at 21.55 again showing an overfitted model.

I will now implement GridSearchCV to find the optimal hyperparameters to perfectly fit the model.

After testing gridsearchcv, I found that the parameters that optimise R2 for RandomForestRegressor was 'max_depth': None and 'n_estimators': 300. However, its effect on R2 seemed very minimal.



The effect of gridsearch was an increase in R2 from .896 on the test set to .897 so effectively no difference at all.

For full coverage sake, I ran it again with a wider gridsearch search including more parameters.

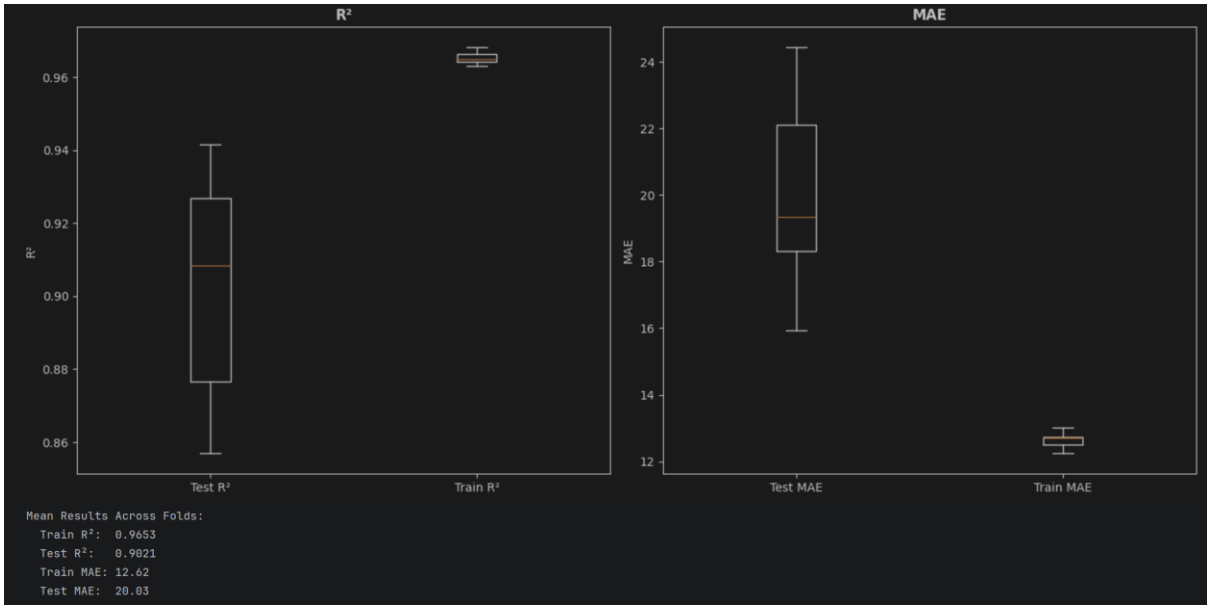
```
param_grid = {
  'n_estimators': [50, 100, 200, 300],
  'max_depth': [5, 10, 15, 20, None],
  'min_samples_split': [10, 20, 30],
  'min_samples_leaf': [5, 10, 15],
  'max_features': ['sqrt', 'log2', 1]
}
```

```
Mean Results Across Folds:
  Train R²: 0.9119
  Test R²: 0.8427
  Train MAE: 20.89
  Test MAE: 27.79
```

This resulted in a slightly worse Test R2 score which confused me on how thats even possible but regardless I think this proves these parameters are not the issue causing overfitting and is likely the lack of data or the model class.

Algorithm 2 - Gradient Boosting Regressor - Model Training and Evaluation

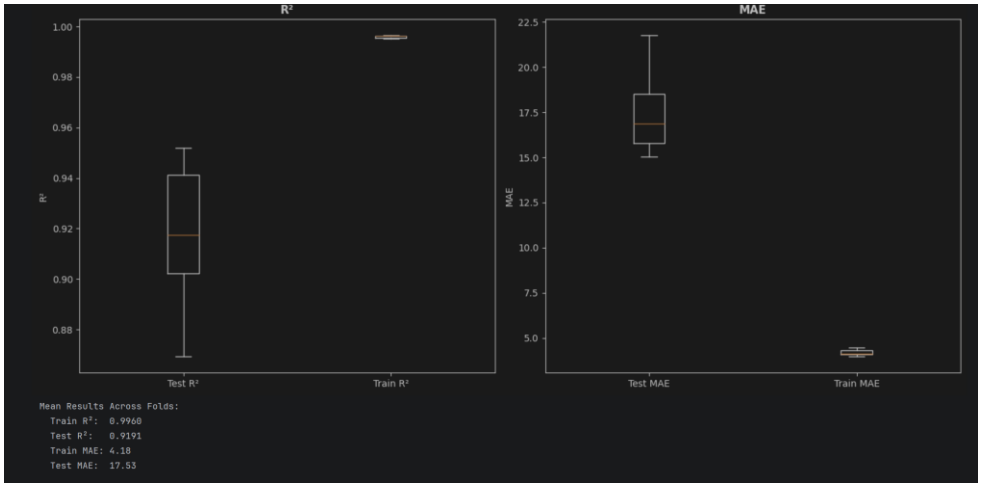
Same as Random forest, here is my first run with default parameters.



Once again, clear overfitting is evident with much better performance on the training set than the test sets. However the error results were nonetheless impressive and outperforming Random Forest even with optiimised parameters.

Now I will run gridsearch and find the optimal parameters and show the results.

Gridsearch found the optimal parameters to be {'max_depth': 3, 'n_estimators': 500}.



Similiar to Random Forest, adjusted the hyperparameters did decrease the error by the model but again quite minimally. From an R2 value of .90 up to approx .91 and reducing the MAE from approx 20 to 17.

Conclusions

In conclusion, the difference between Random Forest and Gradient Boosting is clear, the latter being the obvious and unsurprising winner by having much lower error. Adjusting the hyperparameters didn't have a huge effect on the performance but definetely a consistent effect. The limited amount

of features and small dataset are likely what is limiting this model from being able to correctly predict the tensile strength of steel with little mistake

References

1. Figure 1: https://www.geeksforgeeks.org/machine-learning/random-forest-algorithm-in-machine-learning/
2. Figure 2: https://www.geeksforgeeks.org/machine-learning/ml-gradient-boosting/
3. Random Forest Video: https://www.youtube.com/watch?v=J4Wdy0Wc_xQ
4. Gradient Boosting Video: https://www.youtube.com/watch?v=3CC4N4z3GJc
5. AdaBoost Video: https://www.youtube.com/watch?v=LsK-xG1cLYA

Report Update Log

Date	Update Description	Time Spent
10/11/25	Learned about decision trees and random forest, setup repo and started notebook	3 hours
11/11/25	Finished up notebook for random forest and wrote up about it	3 hours
12/11/25	Learned about adaboost and gradient boost, made notebook and finished report	5 hour