Assignment 2 – Regression using Scikit-learn

Student Name: David Bohan **Student ID**: 20436904 **Programme**: 4BCT

Algorithm 1 – Decision Tree Regression

The Decision Tree Regression algorithm is a commonly used Machine Learning model, capable of accepting both numerical and categorical independent variables, which can be used to predict continuous dependent target values. The model maps input data into a tree-shaped structure, consisting of a root node, branches, internal nodes, and leaf nodes. It is non-parametric in nature, meaning it makes no assumptions about the relationships or structure of the underlying data. However, Decision trees do have drawbacks, including the tendency to overfit to noise in the data, high variance leading to instability with small data variations, and low bias in complex trees.

Detailed Description - See Figures [1], [2].

A decision tree works by recursively splitting featured data based on logical conditions. This divides or classifies the entered data into smaller sub-sets based on the features of the data. The splitting process will continue to occur until the data meets a stopping condition, ie. it has hit the maximum depth of the tree (arriving at a leaf/terminal node) or if the node is pure, meaning it contains data from only one specific class instance. When a new data item is introduced, it will traverse the tree.

Regression Trees are more complex than classification trees. Lets say we have a scatter plot which represents two independent variables X1 and X2. The dependent variable is called Y (third dimension). Lets focus on X1 and X2 to see how our decision tree will be created.

We need to compute the optimal splits for our decision tree. In Decision Tree Regression, we need to calculate which split is decreasing the impurity of the child nodes the most. For that, we need to compute variance reduction. This is a major difference from classification Decision Trees, where impurity is measured by entropy or gini index. A higher value of variance means a higher impurity. To compute the variance reduction, we just subtract the combined variance of the child nodes from the parent node. The weights are the relative size of the child with respect to the parent. Variance reduction typically gets smaller as we move down the Decision Tree. The machine learning model will calculate the variance reduction for every possible split and selects the best one, which happens recursively unless we have reached our desired depth.

To predict the continuous-valued output of Y for a new observation that gets added to our dataset, we take the average value of Y of the data points in the terminal leafs, and that will be the value that will assigned to any new value that falls in this terminal leaf. This is known as the mean value. It is very straightforward. Even though this is a regression, we end up dividing the feature space into several distinct regions, which is very different to other regression techniques.

Why I choose this algorithm.

Decision Trees are a powerful tool for regression. They allow for the simplification of complex relationships between independent variables and dependent target values in a way that is easy to read and comprehend by humans. The tree structure is intuitive and allows for visualization. Its non-parametric nature allows for prediction of numerical data. Finally, by controlling the depth of the tree we reduce the model's sensitivity to outliers.

Hyperparameter Details for Tuning.

Max Depth: This refers to the maximum distance between a root node and a leaf node. It is important to tune this hyperparameter due to the tendency for Decision Trees to over-fit data.

Min Samples Leaf: This hyperparameter defines the minimum number of samples required to be at a leaf node. Helps in model smoothing and prevents prevents overly complex models that can cause overfitting.

Algorithm 2 – Support Vector Regression

Support Vector Machines (SVMs) are a popular type of machine learning algorithm, which are predominately used in data classification, but can also be extended to regression purposes. Support Vector Regression models are used when approximating the relationship between input variables and a continuous-valued output, with a minimal prediction error. This involves turning the input variables into a high-dimensional feature space and finding a hyperplane the optimises both margin (between the hyperplane and the nearest data points) and minimises prediction error. They can be used to solve non-linear problems by utilising the Kernel Trick, allowing SVMs to fit a variety of data distribution such as radial, polynomial and many others. They are not easily interpretable due to the complexity of the underlying algorithm.

Detailed Description - See Figure [3].

Support Vector Regression is a generalisation of the SVM classification problem. Therefore, to fully understand how SVR works we need to look at the underlying principles of the SVM. The objective in SVM is to find the best line that separates two sets of data points. This optimal line, known as a hyperplane, maximises the distance (or margin) between the closest points (support vectors) of each classification. Say D+ represents the shortest distance to the closest male data point, while D- is the shortest distance to the nearest female data point. The sum of D+ and D- gives us the distance margin between the two support vectors. By maximising this distance margin, the optimal hyperplane is identified, and new data points can be easily classified in a linearly-separable case.

Just like our SVM, SVR also uses a support vectors and an optimal hyperplane. Instead of maximising the margin between different classes, SVR aims to fit this hyperplane in a manner that minimises the prediction error for a continuous target variable. In SVR, we have what is called the E-Insensitive region. This tube shaped area has a width of epsilon, measured vertically (not perpendicularly) to the tube. Any data point in our set that falls inside of this tube will be disregarded. We can imagine this to mean that this tube is a margin of error which we are allowing our model to have. This is key behind SVR, it gives a little bit of movement or a little bit of buffer to our model.

We do however have points that are outside the Epsilon insensitive tube and we do care about their error. This error is measured as the distance between the data point and the tube itself (note, not the regression line). These distances are called C* if the point is below the tube or C if the point is above the tube. These values are called slack variables. We want the sum of these distances to be minimal. Effectively the points outside of our tube are dictating what the tube will look like, how the tube will be positioned and where our decision boundaries exist.

Why I choose this algorithm.

VRs are versatile, provide fast predictions and are powerful enough to work well in high- dimensional space (ie. With many features). With the right hyperparameter tuning, SVMs have excellent generalisation capabilities for prediction. SVRs are generally less sensitive to outliers due to minimisation of error in the model through the use of the E-Insensitive region.

Hyperparameter Details for Tuning.

C: Controls the trade-off between allowing training points to deviate from the predicted function and forcing the function to be as flat as possible. For a complex data set, there is a number of different valid decision boundaries we could use.

Gamma: Defines how far the influence of a single training examples reaches. It is used in tuning of non-linear kernels. If Gamma has a low value, that means that every point has a far reach. If gamma has a high value, that means that every point has a close reach. High value of gamma, the decision boundary is going to be dependent on the points closest to decision boundary, effectively ignoring points further away from boundary. Low value gamma, even far values are taken into account when deciding where to draw decision boundary.

Evaluation Defintions

R^2 – Domain Independent – R2 is used as a measure of how much the regession model enhances the overall prediction comapred to just applying the mean value. R2 is measured in the range 0 to 1, where 0 is no improvement over the mean and 1 is perfect accruacy. R2 increases with more model predictions, which can give a false sense of improvement.

Root Mean Square Error – Domain Specific – The RMSE is calculated as the square root of average sqaure differnce between predicted value and observed (residual) values. It provides a direct measure of the model's accuracy in the same unit type as the data. A low RMSE suggests a better model fit, indicating that the model's predictions are close to the actual data.

Regression Appraoch

In the following sections, we will examine process of applying Decision Tree Regression and Support Vector Regression to a dataset called steel. The dependent variable that we are trying to predict is tensile_strength.

Data Preprocessing and Visualisation – See Figure [4]

The code begins by loading in the dataset using the pandas library and perfomring exploratoy analysis. It then calls "df.dropna()" to remove any null values from the dataset. Null values can distort data relationships and lead to inaccurate results. I mapped "tensile_strength" into a histogram plot to evaluate its distribution. The data was then separated into independent variables (X), ie. features and dependent variable (y), ie. target.

Training and Evaluation Details

Both the Decision Tree Regressor and Support Vector Regressor are originally run using their default values; ie. DecisionTreeRegressor(), SVR().

The code makes use of a Pipeline to help in the preprocessing and model execution. It is used to ensure that each fold in the cross-validation process is subject to the same transformation. MinMaxScaler scaling is applied to ensure that the feature values are within a fixed range and contribute equally to the analysis.

We are using KFold for cross-validation, which divides the dataset into 10 different "folds" (n_splits=10). Each fold contains approximately the same number of samples of each target class as the complete set. The use of shuffle randomises the data before splitting occurs, which can be important for removing bias in the model evaluation. A fixed seeding of 42 was used to reproduce the results.

The cross_val_score scikit learn function is used to evaluate the pipeline across different regression evaluation metrics - MSE (from which RMSE can be calculated), MAE, R2. This provides a comprehensive understanding of the model performance. It uses KFold for splitting and the parameter n_jobs=-1 allows for the model to use all available processors to speed up execution.

Decision Tree Regression

Default Training [Figure 5 & 6]

- The mean MSE over all folds is -1539.085, which high indicating that the default DTR model could have substantial prediction errors.
- The absolute mean MAE is -26.2578, suggesting that on average, the model's predictions are about 26 pascals off from the true value.
- The average R2 Score across all folds is 0.8126, which is relatively high. This suggests the model explains a significant portion of the variance in the data.
- The average RMSE across all folds is 39.016, which also suggests the model has notable prediction errors.

Hyperparameter Training [Figure 7 & 8]

The hyperparamters tuned here were max_depth (12) and min_sample_leaf (3).

- The tuned model has a slightly lower mean negative MSE (-1475.771) compared to the original model (-1539.085). This is a small improvement in the accuracy of the tuned model's predictions.
- The absolute mean MAE increased slightly in the tuned model (-27.518) compared to the original (-26.258). Again, slightly better in the tuned model.
- RMSE (38.165) compared to the original model (39.016). This improvement might indicate the tuned model is better at handling outlier values.
- Slightly lower in the tuned model (0.8109) compared to the original model (0.8126). This difference is marginal, suggesting that both models have similar levels of explanatory power.

The lack of substantial improvement through tuning could imply that the selected hyperparameters were not ideal, or the model has reached its performance ceiling considering the current dataset. These modest enhancements could suggest that the Decision Tree Regression may not be the most suitable model for this dataset, or it might require further tuning.

Support Vector Regression

Default Training [Figure 9 & 10]

- The mean MSE over all folds is -6161.422, which is high. This indicates that the default SVR model could have substantial prediction errors.
- The absolute mean MAE is -60.7238, suggesting that on average, the model's predictions are about 60.72 pascals off from the true value.
- The average R2 Score across all folds is 0.2605, which is not particularly high. This suggests the SVR model explains only a relatively small portion of the variance in the data.
- The average RMSE across all folds is 77.267, which also suggests the model has notable prediction errors.

Hyperparameter Training [Figure 11 & 12]

The hyperparamters tuned here were C (100) and Gamma (1).

- The updated mean MSE for the tuned model is -1252.9846. This is a substantial improvement from the previous mean MSE of -6161.422, indicating a reduction in the average squared difference between the estimated values and the actual values.
- The new mean MAE is -26.8645, down from the previous -60.7238. This improvement suggests that on average, the model's predictions are now about 26.86 units off from the true values, which is a significant reduction in the typical prediction error.
- The R2 score has improved to an average of 0.8352 across all folds, up from the previous average of 0.2605.
- The average RMSE for the tuned model is 35.3111, a notable decrease from the earlier average of 77.267. This decrease in RMSE indicates that the residuals (prediction errors) are less spread out, meaning the data points are more concentrated around the line of best fit, indicating a better model performance.

These improved metrics demonstrate that the tuning of parameters C and Gamma has significantly enhanced the performance of the SVR model. The model is now a lot more accurate, as shown by the lower MSE and MAE, it explains a greater portion of the variance in the data (as shown by the higher R2 score), and it has reduced prediction errors (indicated by the lower RMSE).

Conclusions

This report evaluated the use of Decision Tree Regression and Support Vector Regression on a steel dataset, focusing on predicting a continuous valued output, tensile strength. The model evaluation used different performance metrics like MSE, MAE, R^2, and RMSE, resulting in key findings about each model's predictive capabilities. Hyperparameter tuning was executed to optimise model performance, revealing significant improvements - particularly in the SVR model.

Key Findings

- Default settings showed large prediction errors (MSE: -1539.085, RMSE: 39.016), with an average R2 of 0.8126.
- DTR Hyperparameter tuning (max_depth: 12, min_sample_leaf: 3) showed very small improvements in model accuracy. We should try tuning differnet hyperparamters to see if we could increase the accuracy of the model.
- Default model training performance was (MSE: -6161.422, RMSE: 77.267, R2: 0.2605). The model was trained with RBF kernel, this should be tested for performance vs. other kernels.
- Saw significant improvements through tuning (C: 100, Gamma: 1), with large reductions in MSE (-1252.9846) and RMSE (35.3111), and a substantial increase in R^2 (0.8352). This was an interesting insight to see the huge impact of hyperparamter training.

Comparative Analysis of Algorithm Performances

In their default states, the Decision Tree Regression model greatly outperformed the Support Vector Regression model. DTR registered a mean MSE of -1539.085, substantially lower than SVR's -6161.422. DTR's default RMSE of 39.016 was much lower than SVR's 77.267. DTR's default MAE (-26.2578) was considerably lower than SVR's (-60.7238), indicating better average accuracy in DTR's predictions. DTR showed a relatively high R^2 score of 0.8126, implying a significant explanation of variance, whereas SVR's R^2 score of 0.2605 suggested limited explanatory power.

After tuning hyperparameters, the performance dynamics between DTR and SVR changed noticeably. Post-tuning, SVR's MSE improved drastically from -6161.422 to -1252.9846, surpassing DTR's slightly improved MSE of -1475.771. SVR also saw a substantial decrease in RMSE to 35.3111, compared to DTR's modest improvement to 38.165. SVR's MAE improved dramatically to -26.8645 from -60.7238, indicating closer predictions to actual values, while DTR showed a slight increase in MAE. Both models exhibited a comparable level of variance explanation post-tuning, with SVR's R2 improving to 0.8352, closely matching DTR's 0.8109. This suggests that both models, after tuning, are similarly effective in explaining the variability in the data.

Recommended Hyperparameter Valued based on Results

Algorithm 1: Max_Depth: Optimal Max_Depth found to be 12.

Algorithm 1: Min_Samples_Leaf: Optimal Min_Sample_Leaf found to be 3.

Algorithm : 2 C: Optimal Min_Sample_Leaf found to be 100.

Algorithm : 2 Gamma: Optimal Min_Sample_Leaf found to be 1.

Concluding Remarks

This model training shows importance of algorithm selection and hyperparameter tuning in machine learning. While DTR initally perfoemed better, using tuning SVR demonstrated superior performance, particularly in reducing prediction errors and explaining variance. DTR might be a good for its simplicity and lower error rates, but SVR can yield better predictions especially in complex datasets. To create the optimal model for this dataset, we should train multiple ML models and then tune all relevant hyperparameters.

References

[1]. Towards Data Science. 3 techniques to avoid overfitting of decision trees. Retrieved from https://towardsdatascience.com/3-techniques-to-avoid-overfitting-of-decision-trees-1e7d3d985a09

[2]. Scikit-learn. Tree Algorithms: Minimal Cost-Complexity Pruning. Retrieved from https://scikit-learn.org/stable/modules/tree.html#minimal-cost-complexity-pruning

[3]. Towards Data Science. Learn how decision trees are grown. Retrieved from

https://towardsdatascience.com/learn-how-decision-trees-are-grown-22bc3d22fb51

[4]. ScienceDirect. Cost-complexity pruning. Retrieved from https://www.sciencedirect.com/topics/computer-science/cost-complexity

[5]. Wikipedia. Entropy (information theory). Retrieved from

https://en.wikipedia.org/wiki/Entropy_(information_theory)

[6]. YouTube. Videos Retrieved from

https://www.youtube.com/watch?v=_YPScrckx28
https://www.youtube.com/watch?v=rvVkVsG49uU
https://www.youtube.com/watch?v=TtKF996oEl8
https://www.youtube.com/watch?v=QZ0DtNFdDko
https://www.youtube.com/watch?v=oTSj0TTDSPI
https://www.youtube.com/watch?v=RmajweUFKvM
https://www.youtube.com/watch?v=_PwhiWxHK8o&t=55s

https://www.youtube.com/watch?v=_wZ1Lo7bhGg&t=15s

https://www.youtube.com/watch?v=kPw1IGUAoY8&t=7s

[12]. Towards Data Science. Decision trees explained: entropy, information gain, gini index, ccp pruning. Retrieved from https://towardsdatascience.com/decision-trees-explained-entropy-information-gain-gini-index-ccp-pruning-4d78070db36c

[13]. IEEE Xplore. Document Details Not Specified. Retrieved from

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1250975

[14]. Wikipedia. Support vector machine. Retrieved from

https://en.wikipedia.org/wiki/Support_vector_machine

[15]. Analytics Vidhya. Understanding Support Vector Machine algorithm from examples (along with code). Retrieved from https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

[16]. Stack Abuse. Understanding SVM Hyperparameters. Retrieved from

https://stackabuse.com/understanding-svm-hyperparameters/

[17]. ScienceDirect. Article Title Not Specified. Retrieved from

https://www.sciencedirect.com/science/article/abs/pii/S0925231220307153

[18]. Scikit-learn. sklearn.svm.SVC. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[19]. Wikipedia. Decision tree pruning. Retrieved from

https://en.wikipedia.org/wiki/Decision_tree_pruning

[20]. IEEE Xplore. Retrieved from https://ieeexplore.ieee.org/abstract/document/6498972

[21]. ISANA Systems. Machine Learning: Handling Dataset Having Multiple Features. Retrieved from https://www.isanasystems.com/machine-learning-handling-dataset-having-multiple-features/

[22]. NBShare. (n.d.). Decision Tree Regression With Hyper Parameter Tuning In Python. Available at: https://www.nbshare.io/notebook/312837011/Decision-Tree-Regression-With-Hyper-Parameter-Tuning-In-Python/

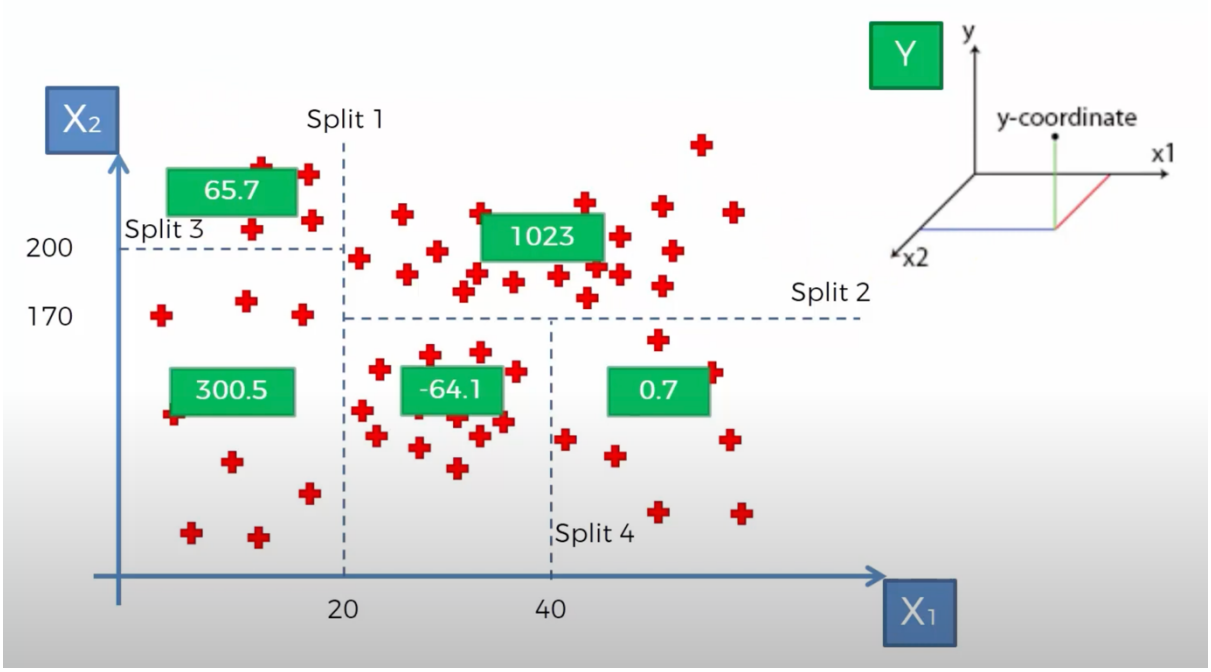

Figure 1: Visualisation of Graphed Decision Tree Regression Data Splitting

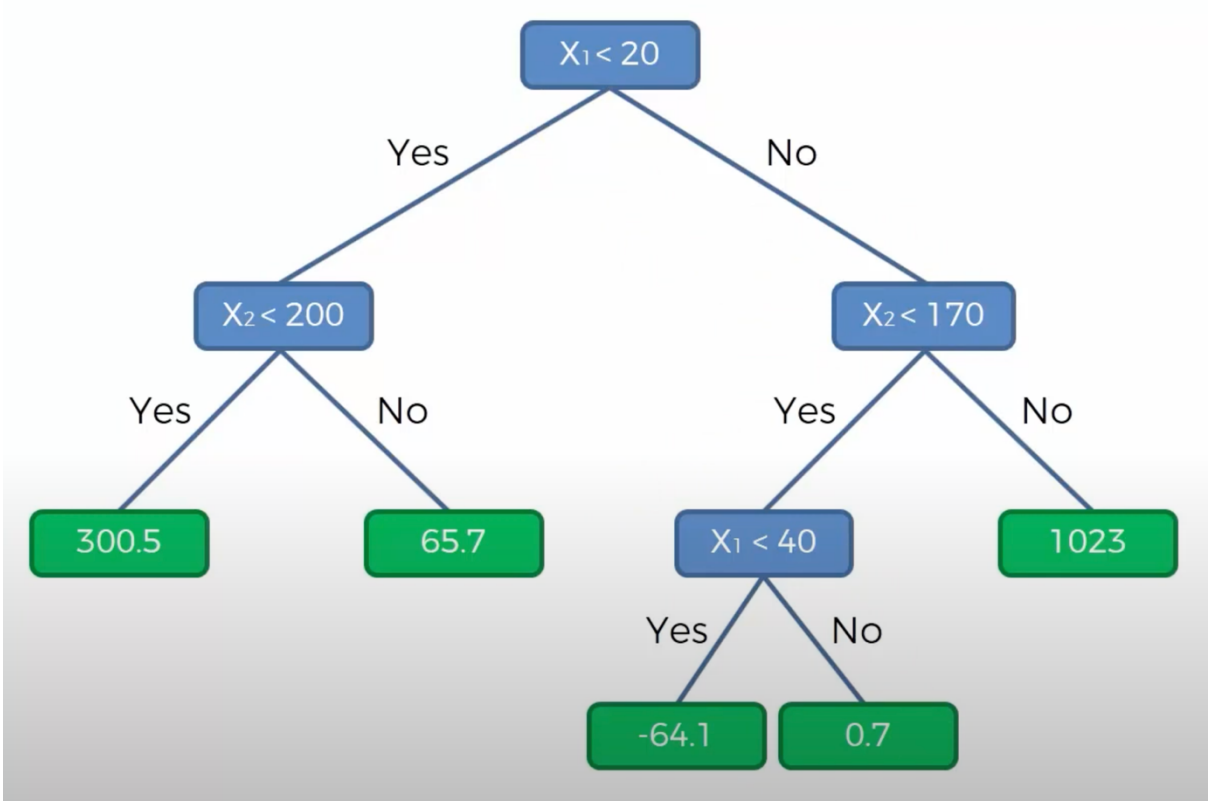

Figure 2: Visualisation of Decision Tree, used in Applying Mean Value

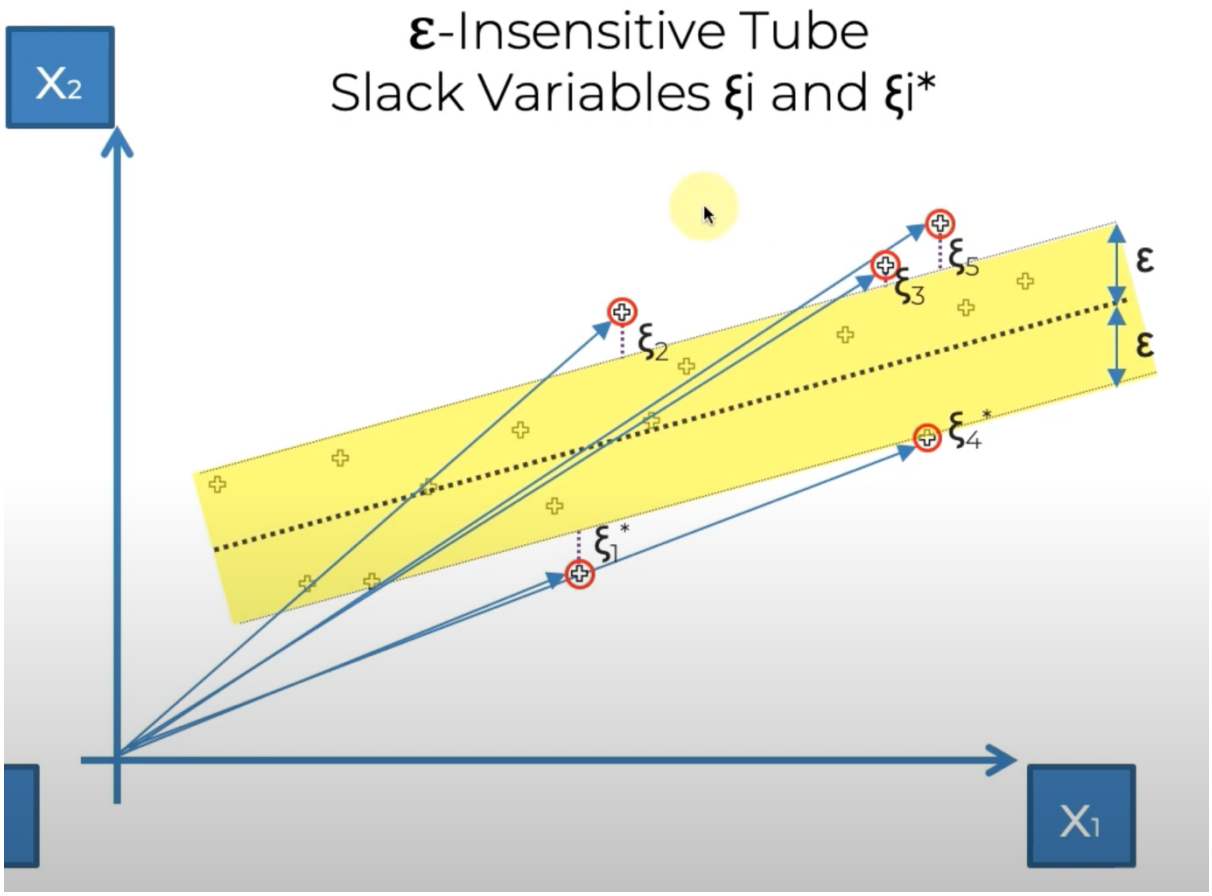

Figure 3: Visualisation of SVR showing Hyperplane, Support Vectors, Slack Variables, E-Insensitive Tube

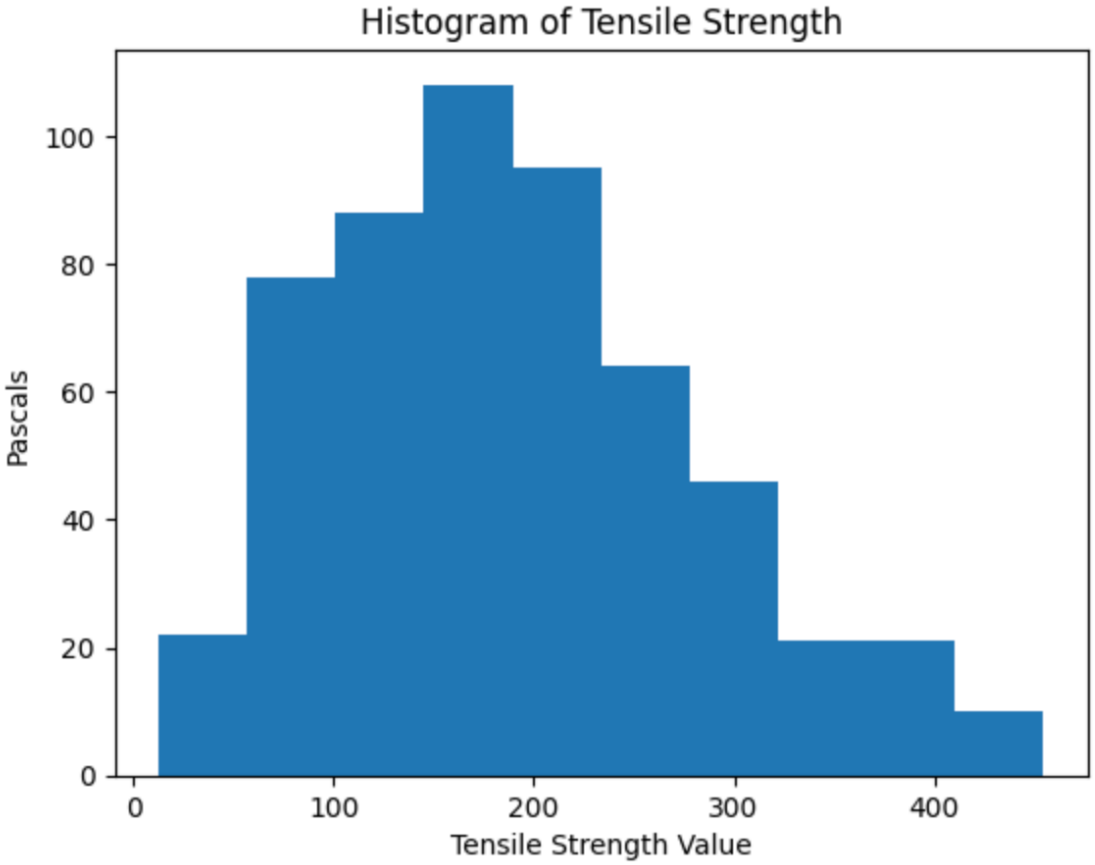

Figure 4: Plotting Target Variable to Look at its Distribtuion

```
Fold 1: Negative MSE: -1727.9716007789648, Negative MAE: -28.96014397428572, R2 Score: 0.8733928375324116, RMSE: 41.56887779070978
Fold 2: Negative MSE: -1089.9831380953367, Negative MAE: -25.545525525714286, R2 Score: 0.8252681218542662, RMSE: 33.014892671267866
Fold 3: Negative MSE: -1700.3615130589476, Negative MAE: -28.52170935589286, R2 Score: 0.7273528472022797, RMSE: 41.23544001291786
Fold 4: Negative MSE: -1330.5145812869016, Negative MAE: -20.35408995672727, R2 Score: 0.883509546526871, RMSE: 36.47621939410527
Fold 5: Negative MSE: -1936.0098418400128, Negative MAE: -28.72936175890909, R2 Score: 0.7313453438605245, RMSE: 44.00011183894892
Fold 6: Negative MSE: -1413.7592653973034, Negative MAE: -27.431265586181816, R2 Score: 0.7785896669419201, RMSE: 37.59999023134585
Fold 7: Negative MSE: -1207.1311798973813, Negative MAE: -24.14148010890909, R2 Score: 0.8263340111845466, RMSE: 34.74379340108649
Fold 8: Negative MSE: -1993.5479873351767, Negative MAE: -30.76917649472727, R2 Score: 0.7218097656261184, RMSE: 44.649165583862555
Fold 9: Negative MSE: -1827.172514914817, Negative MAE: -27.017964130545458, R2 Score: 0.8727547291444848, RMSE: 42.74543852757645
Fold 10: Negative MSE: -1164.4020243264515, Negative MAE: -21.10719472472726, R2 Score: 0.8860234356969149, RMSE: 34.12333548067146
----------
Mean Negative Mean Squared Error over all folds is: -1539.0853646931293
Absolute Mean Negative Mean Squared Error over all folds is: -26.25779116166201
Average RMSE over all folds is: 39.01572649324925
R2 Score over all folds is: 0.8126380305570338
```

Figure 5: Default Decision Tree Regression Outputs

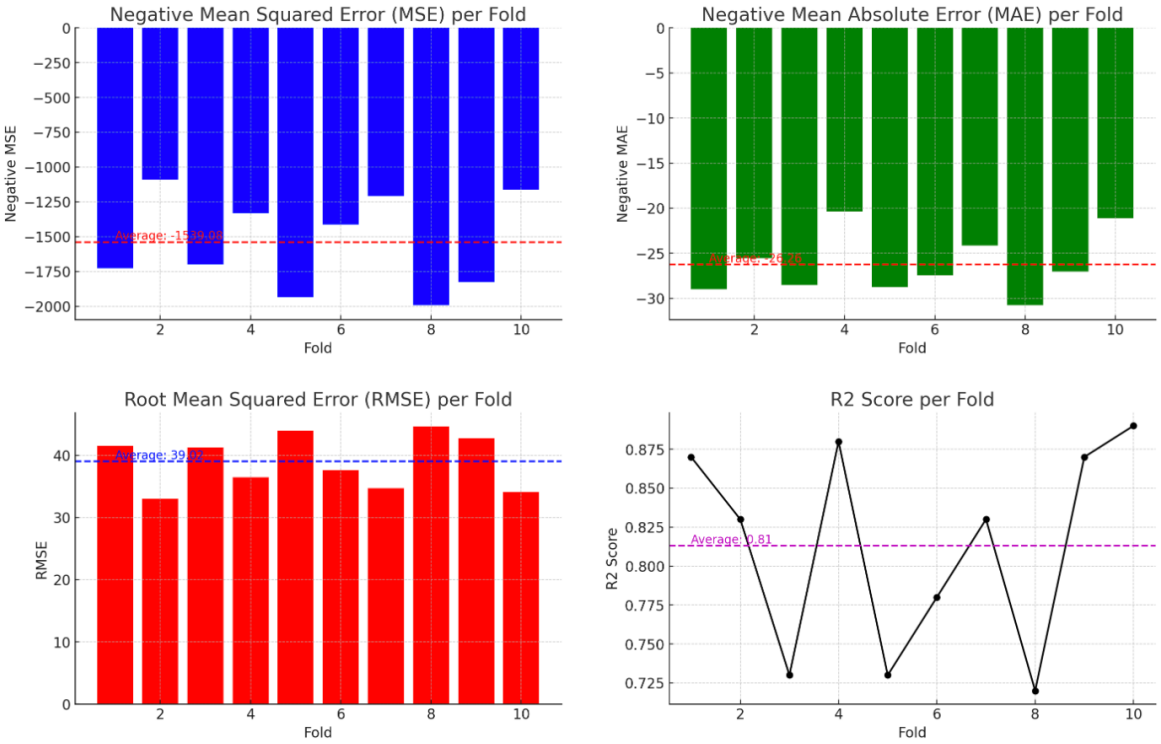

Figure 6: Graphed Default Decision Tree Regression Outputs

```
Best Parameters: {'model__max_depth': 12, 'model__min_samples_leaf': 3}
Best Score: 1425.5244760865075
Fold 1: Negative MSE: -1441.7044275207923, Negative MAE: -26.503349062267855, R2 Score: 0.8767919429955167, RMSE: 37.969783084984726
Fold 2: Negative MSE: -1525.9509952666356, Negative MAE: -29.818653697159863, R2 Score: 0.7570376755785029, RMSE: 39.06342272851466
Fold 3: Negative MSE: -1463.9282076966883, Negative MAE: -28.620807296473213, R2 Score: 0.7335251192861391, RMSE: 38.26131476696388
Fold 4: Negative MSE: -1387.7558719175136, Negative MAE: -26.88951401637374, R2 Score: 0.846235208122958, RMSE: 37.2525955057834
Fold 5: Negative MSE: -2289.0426700500307, Negative MAE: -28.948997132469692, R2 Score: 0.7543748078321029, RMSE: 47.84394078720973
Fold 6: Negative MSE: -949.6531407459776, Negative MAE: -24.135580979078796, R2 Score: 0.8285206215678352, RMSE: 30.816442701031825
Fold 7: Negative MSE: -1079.6641041832866, Negative MAE: -24.400246708004037, R2 Score: 0.8480298139947925, RMSE: 32.85824256078354
Fold 8: Negative MSE: -1697.3675509738114, Negative MAE: -28.315804365257573, R2 Score: 0.7510108324630161, RMSE: 41.199120754863344
Fold 9: Negative MSE: -1585.7182238124149, Negative MAE: -31.903356823455955, R2 Score: 0.876065391082954, RMSE: 39.82107763248522
Fold 10: Negative MSE: -1336.9289070339132, Negative MAE: -25.647995287124104, R2 Score: 0.8372899129939209, RMSE: 36.56403843989219
----------
Mean Negative Mean Squared Error for Tuned Model: -1475.7714099201062
Absolute Mean Negative Mean Squared Error for Tuned Model: -27.51843053676648
Average RMSE for Tuned Model: 38.16499789625125
R2 Score for Tuned Model: 0.810888132591774
```

Figure 7:Tuned Hyperparameter Decision Tree Regression Outputs

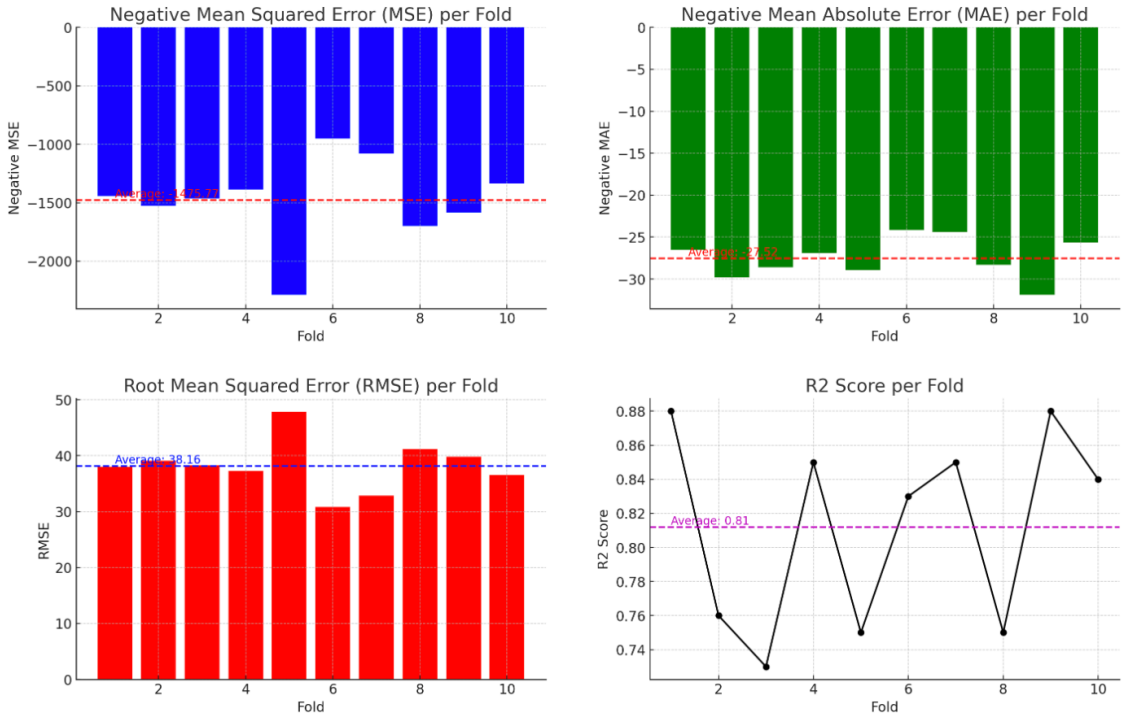

Figure 8: Graphed Tuned Hyperparameter Decision Tree Regression Outputs

```
Fold 1: Negative MSE: -9871.642326022542, Negative MAE: -79.02509060207487, R2 Score: 0.21821402629616293, RMSE: 99.35613884417279
Fold 2: Negative MSE: -5474.696835883902, Negative MAE: -57.34630352709913, R2 Score: 0.19118640767581485, RMSE: 73.99119431313366
Fold 3: Negative MSE: -3860.9366926395587, Negative MAE: -49.43112612392205, R2 Score: 0.29720416670319305, RMSE: 62.136436111508345
Fold 4: Negative MSE: -7356.599978972933, Negative MAE: -66.23050273850424, R2 Score: 0.17306548006924172, RMSE: 85.77062421932659
Fold 5: Negative MSE: -5578.641040988181, Negative MAE: -56.446266019776594, R2 Score: 0.2707312307193954, RMSE: 74.69030084949571
Fold 6: Negative MSE: -3594.8507092871073, Negative MAE: -47.835629268196264, R2 Score: 0.3502504451426599, RMSE: 59.95707388863392
Fold 7: Negative MSE: -5306.8616434485575, Negative MAE: -59.784226153529374, R2 Score: 0.25302253920088824, RMSE: 72.84820961045342
Fold 8: Negative MSE: -4271.011116915675, Negative MAE: -55.10251971833725, R2 Score: 0.3785395028415971, RMSE: 65.35297328290179
Fold 9: Negative MSE: -10531.056301693945, Negative MAE: -76.37254320134373, R2 Score: 0.20153018836678538, RMSE: 102.6209350069173
Fold 10: Negative MSE: -5767.924141239448, Negative MAE: -59.66374561769316, R2 Score: 0.2716000385566958, RMSE: 75.94685076577862
----------
Mean Negative Mean Squared Error over all folds is: -6161.422078709185
Absolute Mean Negative Mean Squared Error over all folds is: -60.72379529704767
Average RMSE over all folds is: 77.26707368923222
R2 Score over all folds is: 0.26053440255724347
```

Figure 9: Support Vector Regression Outputs

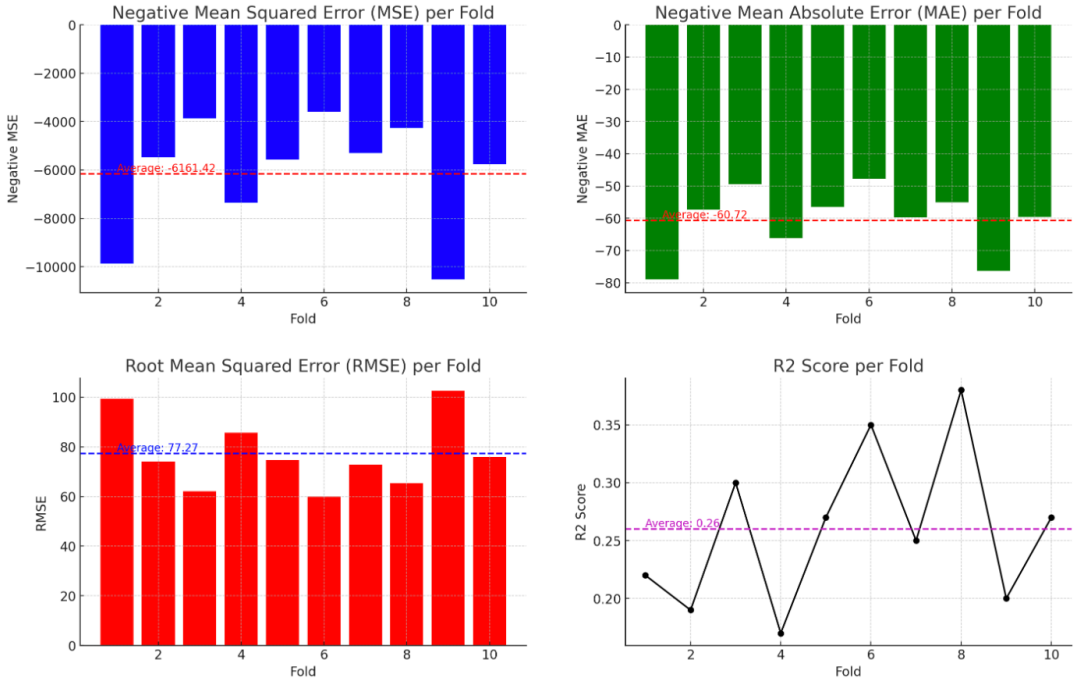

Figure 10: Graphed Support Vector Regression Outputs

```
Best Parameters: {'model__C': 100, 'model__gamma': 1}
Best Score: 1252.9846101961698
Fold 1: Negative MSE: -1399.74626368929, Negative MAE: -30.349540274342637, R2 Score: 0.8891469160291636, RMSE: 37.41318302001702
Fold 2: Negative MSE: -1276.274127073543, Negative MAE: -29.06450364942873, R2 Score: 0.8114474842254705, RMSE: 35.724979035312856
Fold 3: Negative MSE: -1473.3503275278292, Negative MAE: -29.56809018053686, R2 Score: 0.7318100363709561, RMSE: 38.384245824658706
Fold 4: Negative MSE: -1252.114396692702, Negative MAE: -27.45114649072778, R2 Score: 0.8592533751343068, RMSE: 35.3852285098274
Fold 5: Negative MSE: -1144.1410837257458, Negative MAE: -25.224750884389902, R2 Score: 0.8504319683088533, RMSE: 33.82515460017509
Fold 6: Negative MSE: -1085.4207122976002, Negative MAE: -24.03653952515082, R2 Score: 0.8038161576984761, RMSE: 32.94572373309775
Fold 7: Negative MSE: -1385.2912680836791, Negative MAE: -26.397051845216165, R2 Score: 0.8050106779818182, RMSE: 37.219501179941666
Fold 8: Negative MSE: -1260.5229547932142, Negative MAE: -26.482473555326745, R2 Score: 0.8165855342631185, RMSE: 35.50384422556541
Fold 9: Negative MSE: -1375.9118207150123, Negative MAE: -28.009831187626506, R2 Score: 0.895677696440241, RMSE: 37.09328538583516
Fold 10: Negative MSE: -877.0731473630847, Negative MAE: -22.06063654714768, R2 Score: 0.8892391732140662, RMSE: 29.615420769644395
----------
Mean Negative Mean Squared Error for Tuned Model: -1252.9846101961698
Absolute Mean Negative Mean Squared Error for Tuned Model: -26.86445641398938
Average RMSE for Tuned Model: 35.31105662840754
R2 Score for Tuned Model: 0.835241901966647
```

Figure 11: Tuned Support Vector Regression Outputs

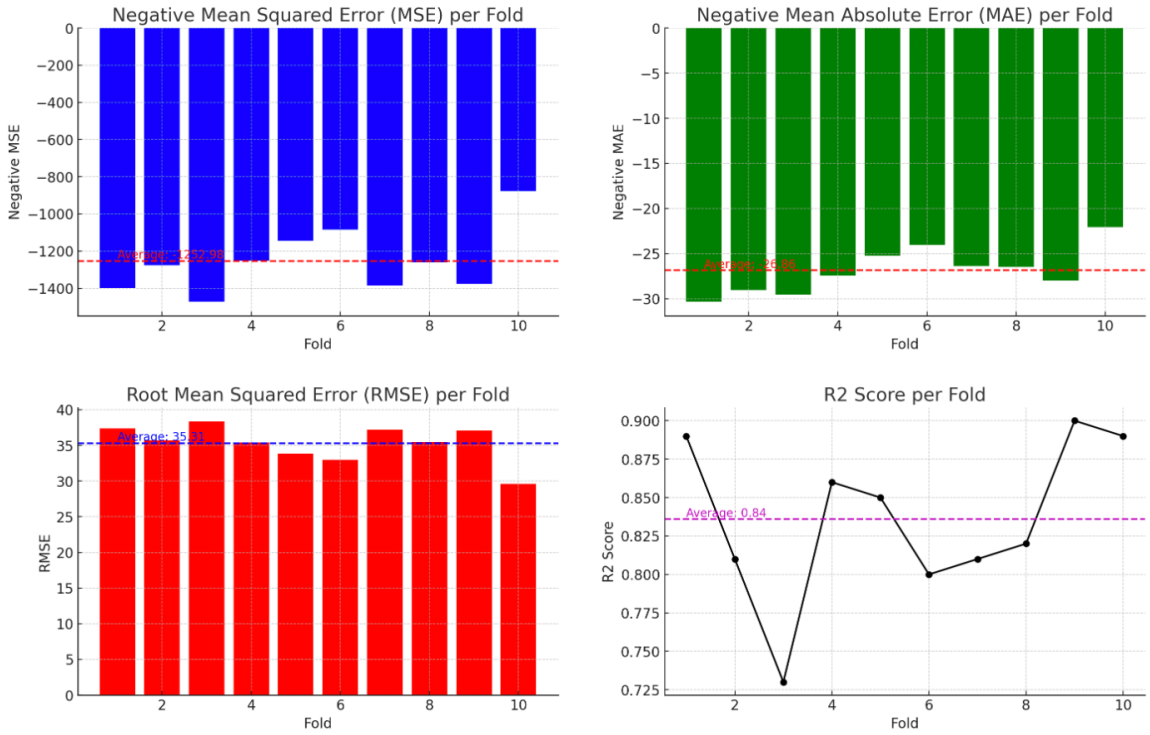

Figure 12: Graphed Tuned Support Vector Regression Outputs