Journal of Systemics, Cybernetics and Informatics (2025) 23(5), 41-52
https://doi.org/10.54808/JSCI.23.05.41

Predicting strength of high-performance concrete using gradient boosting machine learning: A comparative analysis between manual and Grid Search Cross-validation hyperparameter tuning

Ryan TYLER
Department of Civil and Environmental Engineering and Building Science
University of South Africa
Pretoria, South Africa

Masengo ILUNGA
Department of Civil and Environmental Engineering and Building Science
University of South Africa
Pretoria, South Africa

Bolanle IKOTUN
Department of Civil and Environmental Engineering and Building Science
University of South Africa
Pretoria, South Africa

Omphemetse ZIMBILI
Department of Civil and Environmental Engineering and Building Science
University of South Africa
Pretoria, South Africa

ABSTRACT

This study evaluates the effectiveness of a gradient boosting regression model in forecasting concrete strength by comparing three hyperparameter configurations: default settings, manual tuning, and automated Grid Search CV. A publicly available dataset of 1030 concrete mixes, featuring cement, slag, fly ash, water, superplasticiser, coarse and fine aggregates, and concrete age, was divided into an 80-20 train-test split. Model performance was assessed using mean absolute error (MAE), mean squared error (MSE), root mean squared error (RMSE), and the coefficient of determination (R^2). The default model achieved $R^2 = 0.89$, while targeted adjustments to the parameters, such as the number of estimators, raised R^2 to 0.93. Manual fine-tuning of all hyperparameters simultaneously produced the best results of $R^2 = 0.94$, marginally outperforming Grid Search CV 3- and 5-fold of $R^2 = 0.93$. The number of estimators was identified as the most influential parameter. Although exhaustive grid search offers systematic optimisation with high runtimes, manual fine-tuning can yield superior accuracy within a constrained parameter space.

Keywords: high-performance concrete, compressive strength, gradient boosting, hyperparameter tuning, error evaluation

1. INTRODUCTION

Concrete has been a cornerstone of the building industry because of its durability, adaptability, and versatility. As urbanisation accelerates globally, the demand for more sustainable and efficient construction materials has grown. High-performance concrete has emerged as a critical material capable of delivering high strength and durability while optimising resource utilisation. Predicting the strength of high-performance concrete is crucial to construction. However, traditional empirical methods for predicting concrete strength often rely on fixed formulas or trial-and-error experimentation. This can lead to suboptimal mix designs, increased costs, and material waste, especially when dealing with complex concrete formulations. Machine learning, through gradient boosting, offers a modern, accurate, and resource-efficient alternative by learning patterns directly from data rather than relying on rigid heuristics [1]. This study aims to implement and assess a machine learning model, i.e. gradient boosting, for predicting the strength of high-performance concrete by comparing hyperparameter tuning approaches. The model will be performed without any hyperparameter tuning, then manual and automated hyperparameter tuning will be carried out [2].

2. CONCRETE STRENGTH OVERVIEW

Background

Renowned for its versatility, durability, and adaptability, concrete is a widely used building material. Cement, water, admixtures, and fine and coarse aggregates primarily make up its composition. The interaction between these components creates a hard, rock-like material through the process of hydration, where the cement chemically reacts with water. Concrete's characteristics depend on its mix design, including the type and ratio of materials used. Key properties include compressive strength, workability, durability, and thermal resistance [3].

Strength Determination

Determining the strength of concrete is a fundamental aspect of construction and engineering research. Its strength, particularly compressive strength, reflects its ability to withstand loads without failing and directly influences its suitability for various applications. Factors that affect the concrete's strength are the curing conditions, water-cement ratio, mix design, admixtures used, and the age of the concrete. Concrete's strength can be separated into two types, namely, compressive and tensile strength. Tests are done on samples to determine the strength [4].

The most frequently conducted test is the compressive strength test. Cylindrical or, most often, cubic samples are cured and subjected to axial compression using a universal testing machine or hydraulic press [4-5]. The maximum load the sample

withstands before failure is captured, where the formula for the compressive strength is:

$$f_{cc} = \frac{F}{A_c} \quad (1)$$

where:
$f_{cc} = strength, compressive\ (MPa)$
$F = failure\ load\ (N)$
$A_c = Area, cross\ sectional\ (mm^2)$

3. MACHINE LEARNING

Overview
A subset of artificial intelligence called machine learning allows models to learn from given datasets and make decisions with little human assistance. The three learning types of machine learning models are reinforcement learning, unsupervised learning, and supervised learning [6]. Regression and classification problems are two more subcategories of supervised learning. Although classification problems are widely studied, this study strictly focused on regression models.

In supervised learning, models learn from labelled datasets, where each input is associated with a known output, called the target variable. This enables the model to learn the relationship between inputs and outputs, so it can later predict or classify new, unseen examples. If the target variable consists of discrete categories, the problem is referred to as classification. If the target variable is numerical, the task is a regression problem [6]. The nature of the target variable determines whether a classification or regression approach is appropriate.

In unsupervised learning, algorithms identify patterns and relationships in data without labelled outputs. These methods are commonly used for clustering, such as grouping similar behaviours and principal component analysis for dimensionality reduction [6].

Reinforcement learning, however, involves an agent interacting with an environment to maximise cumulative rewards through trial and error. This technique is widely applied in robotics, autonomous systems, and gaming, with foundational concepts such as Q-learning and policy optimisation [7]. Reinforcement learning does not depend on predefined datasets but rather learns optimal actions over time through iterative feedback mechanisms.

Features are the independent variables that provide information to a model, while labels are the dependent variables that the model aims to predict [8]. The goal of a machine learning model is to identify a function that converts inputs to outputs. Various techniques, such as dummy regression and gradient boosting, refine the function to improve predictive effectiveness [9]. In regression, this function generally takes the form:

$$y = f(x) + \varepsilon \quad (2)$$

where:
$y = label$
$x = features$
$f(x) = function\ of\ feature\ to\ label$
$\varepsilon = noise$

Training, Validating, and Testing
To evaluate model performance, datasets are separated into training, validation, and test sets [10]. Quite often, training and testing may be enough, depending on the size of the dataset. In this case, testing coincides with validation. The training set is primarily used to teach the model patterns within the data, and it is common practice to allocate 60 to 80% of the dataset for this purpose, enabling the model to learn effectively without overfitting to the data [11]. Overfitting, or high variability, results from models that are unable to generalise new data as a model learns patterns unique to training data. In contrast, underfitting indicates that the model is too simplistic and does not capture meaningful patterns, leading to high error rates. Regularisation techniques, such as penalty functions, mitigate these issues [8].

By using hyperparameter cross-validation techniques such as k-fold, where k-fold splits data into k parts to train and test the model k times for reliable tuning, researchers can evaluate model performance across different subsets of data, ensuring that the model's parameters are optimised for generalisation. Validation also acts as a checkpoint to detect overfitting during the training phase [10-11].

The testing set is employed for the final model evaluation and is kept entirely separate from the training and validation data. It guarantees that the algorithm's performance on truly unseen data is evaluated, providing an accurate measure of its generalisation ability. Split ratios are employed depending on the dataset size and specific use cases. However, careful consideration is required to strike a balance between training and testing data availability [10],[12].

Error Evaluation
Error evaluators, including mean squared error (MSE), mean absolute error (MAE), root mean squared error (RMSE), and coefficient of determination (R^2), are used to quantify accuracy for regression tasks [13]:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - \hat{x}| \quad (3)$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x})^2 \quad (4)$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x})^2} \quad (5)$$

$$R^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2 - \sum_{i=1}^{n}(x_i - \hat{x})^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \quad (6)$$

where:
$i = index$
$n = number\ of\ operations$
$x_i = actual\ value$
$\bar{x} = mean\ value$
$\hat{x} = predicted\ value$

Gradient boosting algorithm
Boosting is a machine learning technique that combines weak learners to create a stronger overall predictor. Gradient boosting applies this by sequentially integrating several of these weak learners, where each new model focuses on correcting the errors made by the previous ones. Furthermore, each subsequent model undergoes training to remedy the previous model's residual errors to increase predicted accuracy. For problems regarding

regression, gradient boosting works effectively because of this iterative process [14].

Gradient boosting algorithms, by iteratively adding models that predict the loss function's negative gradient, minimise the loss function. The negative gradient represents the direction of the errors made by the current model, and training the next model to predict this gradient helps reduce those errors. A learning rate is applied to each model to control its influence on the final prediction, striking a balance between convergence speed and the risk of overfitting [15].

Hyperparameter tuning

Hyperparameters are the settings defined before training of a machine learning model has begun; they control how the learning process unfolds. Hyperparameter tuning is essential for optimising regression models, as it directly influences predictive accuracy and generalisation. Many techniques offer systematic approaches to examining various hyperparameter configurations, while more advanced approaches improve efficiency by guiding the search process based on probabilistic models or evolutionary strategies. Gradient-based optimisation refines hyperparameters dynamically using gradient descent, ensuring optimal learning rates and regularisation strengths. Additionally, adaptive tuning methods adjust hyperparameters in real-time based on model performance, enhancing robustness and convergence [16]. There are several types of hyperparameter tuning techniques:

- Grid Search CV [17]
- Randomised Search CV [18]
- Optuna [19]
- Bayesian Optimisation [20]

However, the focus of this study is on Grid Search CV.

Manual hyperparameter tuning involves adjusting values by means of trial and error for parameters such as learning rate, number of estimators, or max depth by hand. It is a simple iterative experiment, but it can be time-consuming depending on experience [21]. This is often the first step in the hyperparameter process, as no extra tooling is required and it offers a direct insight into the model behaviour, but it is highly subjective and becomes impractical as the number of parameters grows [22].

Grid Search CV automates hyperparameter tuning by exhaustively evaluating all combinations of user-defined parameter values using k-fold cross-validation. It returns the best configuration based on a chosen error evaluation metric [21]. This method is advantageous as it is systematic and exhaustive, reduces overfitting, and is simple to implement, but as the hyperparameter range becomes wider, the computational cost grows exponentially, and this makes it harder as you cannot give a variety of values [22].

4. DATA AVAILABILITY AND METHODS

Data used

The study employed a public dataset of input data on *Kaggle* from Choundary [16]. The dataset provided the features, which included cement, slag, fly ash, water, superplasticiser, coarse aggregate, fine aggregate content, and the age of the concrete, and the label, which was the actual concrete compressive strength (MPa), determined from the laboratory, with 1030 mix designs of the features and labels.

Methods

A gradient boosting model using training, validation, and test datasets, employing split ratios such as 60-40, 70-30, and 80-20, was developed [10], [12]. The last split was adopted in this study. Error metrics, as explained earlier, were used to assess the machine learning models' performance.

During validation, hyperparameter tuning approaches were used to enhance the models' performance, optimising for accuracy and efficiency. Without any, manual and automated Grid Search CV hyperparameter tuning was performed. Results were compared between manual and automated approaches to identify the superior approach.

The research employed machine learning libraries, such as Scikit learn from Python, and Google Colab was used as a cloud-based computing environment. These tools were freely available online and ensured the replicability of the research. Figure 1 below shows the flow diagram of the methodological approach:

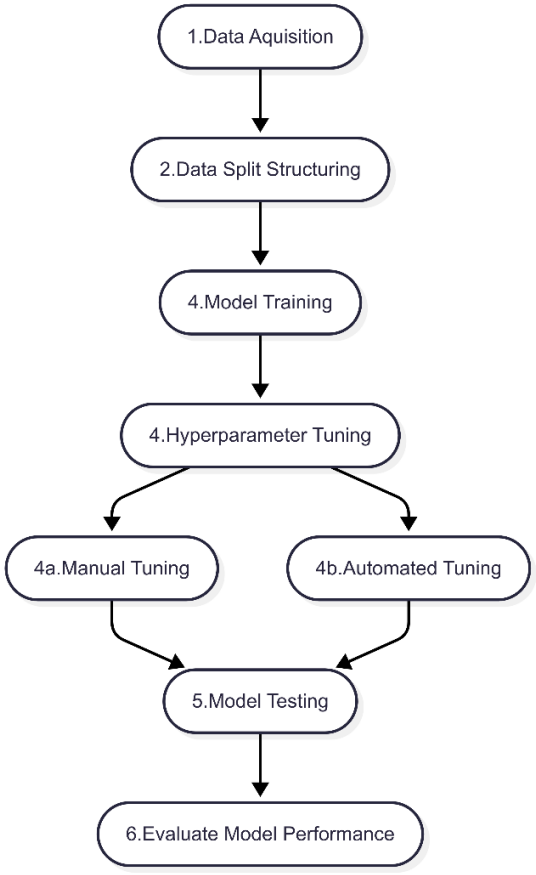

Figure 1: Flow of Research

1. Data acquisition: Acquire and review the above-mentioned dataset.
2. Dataset split structuring: Configure a ratio split for training and testing of the dataset.
3. Model Training: Use the dataset to train the gradient boosting model.
4. Tune models' hyperparameters: After training, apply hyperparameter techniques.

a) Manual Tuning: Use trial and error to manually determine the best hyperparameters.
b) Automated Tuning: Use Grid Search CV to determine the best hyperparameters.

5. Model Testing: Use the rest of the dataset to test the model.
6. Model performance assessment: Use error metrics to compute the performance of the models.

5. RESULTS AND DISCUSSION

For this gradient boosting model, a random state of 42 is used throughout to ensure reproducibility, and an 80-20 split was used in three different cases, namely:

- No hyperparameter tuning
- Manual hyperparameter tuning
- Grid Search CV automated hyperparameter tuning

Hyperparameter settings

The default settings were applied for no hyperparameter tuning, as shown in Table 1 below.

Table 1: No hyperparameter settings

Hyperparameter	Value
n_estimators	100
learning_rate	0.1
max_depth	3
min_samples_split	2
min_samples_leaf	1
max_features	None
Subsample	1
random_state	42

For manual tuning, each parameter was tuned to give the best result while keeping the other parameters at the default setting. The best parameter was kept for each adjustment and is shown in Table 2 below. However, this did not actually give the best result; the hyperparameters needed to be fine-tuned simultaneously to achieve better results and can be found in Table 3 below.

Table 2: Manual hyperparameter settings

Hyperparameter	Value
n_estimators	660
learning_rate	0.42
max_depth	6
min_samples_split	4
min_samples_leaf	5
max_features	5
subsample	0.8
random_state	42

Table 3: Manual hyperparameter settings (fine-tuned)

Hyperparameter	Value
n_estimators	830
learning_rate	0.14
max_depth	3
min_samples_split	2
min_samples_leaf	5
max_features	6
subsample	0.8
random_state	42

For automated hyperparameter tuning, Grid Search CV requires a set of values for each parameter; the results from the manual tuning have been included in the set of values and are shown in Table 4 below. Table 5 and 6 shows the best parameters from the result of Grid Search CV, where a cross-validation of 3- and 5-fold was applied.

Table 4: Automated hyperparameter tuning values

Hyperparameter	Values
n_estimators	50, 500, 830
learning_rate	0.05, 0.1, 0.14, 0.5, 1
max_depth	3, 5, 7
min_samples_split	2, 3, 4, 5, 6
min_samples_leaf	1, 2, 3, 4, 5
max_features	1, 3, 5, 6
subsample	0.1, 0.5, 0.8
random_state	42

Table 5: Automated hyperparameter settings (3-fold)

Hyperparameter	Value
n_estimators	830
learning_rate	0.14
max_depth	3
min_samples_split	6
min_samples_leaf	2
max_features	3
subsample	0.8
random_state	42

Table 6: Automated hyperparameter settings (5-fold)

Hyperparameter	Value
n_estimators	830
learning_rate	0.05
max_depth	7
min_samples_split	2
min_samples_leaf	5
max_features	1
subsample	0.5
random_state	42

Error Evaluation

From Table 7 below is a summary of all error metric results from training and testing, using the hyperparameter settings as described in Table 1 to Table 6 is displayed. From this, the n_estimators parameter has the most effect on the dataset. Comparing R^2 for the different cases, the manual and automated hyperparameter tuning scored the same value, which was higher than the default setting. Even if R^2 is identical, comparing the MAE, MSE, and RMSE shows you which model is more precise and how tightly your predictions cluster around the true values. For instance, comparing Grid Search CV 3- to 5-fold, 3-fold shows a lower MAE and MSE but a higher RMSE; this indicates that the predictions are accurate on average, but with a few big outliers driving up the RMSE.

Table 7: Summary of error metrics

Adjusted Parameter	MAE	MSE	RMSE	R^2
No Hyperparameter Tuning				
default	3.98	30.10	5.49	0.89
Manual Hyperparameter Tuning				
n_estimators	2.84	19.60	4.43	0.93
learning_rate	3.13	21.63	4.65	0.92
max_depth	3.03	22.44	4.74	0.92
min_samples_split	3.97	30.03	5.48	0.90
min_samples_leaf	3.88	29.33	5.42	0.90
max_features	3.96	29.41	5.42	0.90
subsample	3.64	27.12	5.21	0.91
all optimised	3.19	33.48	5.79	0.88
fine tuned	2.58	16.14	4.02	0.94
Automated Hyperparameter Tuning				
gridsearchcv 3-fold	2.80	19.99	4.47	0.93
gridsearchcv 5-fold	2.71	20.33	4.51	0.93

Actual vs Predicted Comparison

From Figure 2 to Figure 13 below, the model's performance when predicting compressive strength in each case is presented. These figures show a high prediction capability, with an R^2 ranging between 88 and 94%. Figure 3 to Figure 9 below illustrates the results of adjusting one parameter at a time, highlighting the varying effects of each parameter. Whereas Figure 10 utilises all of the best parameters at once, but shows a significantly lower R^2 than its individual counterparts and Figure 11 fine-tunes these parameters to ensure that they work better with one another. Figure 12 and Figure 13 utilises different hyperparameter settings but results in similar R^2 scores.

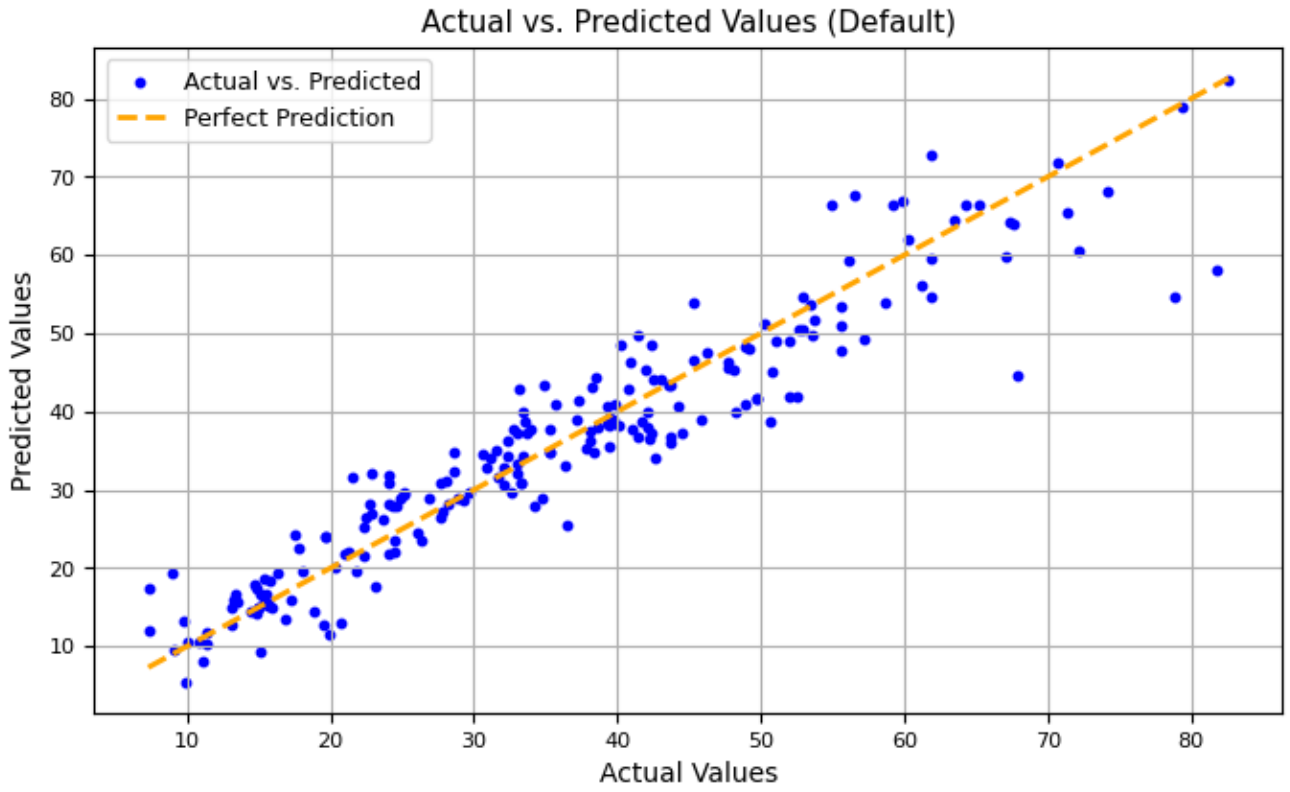

Figure 2: Default settings (see Table 1, R^2 =0.89)

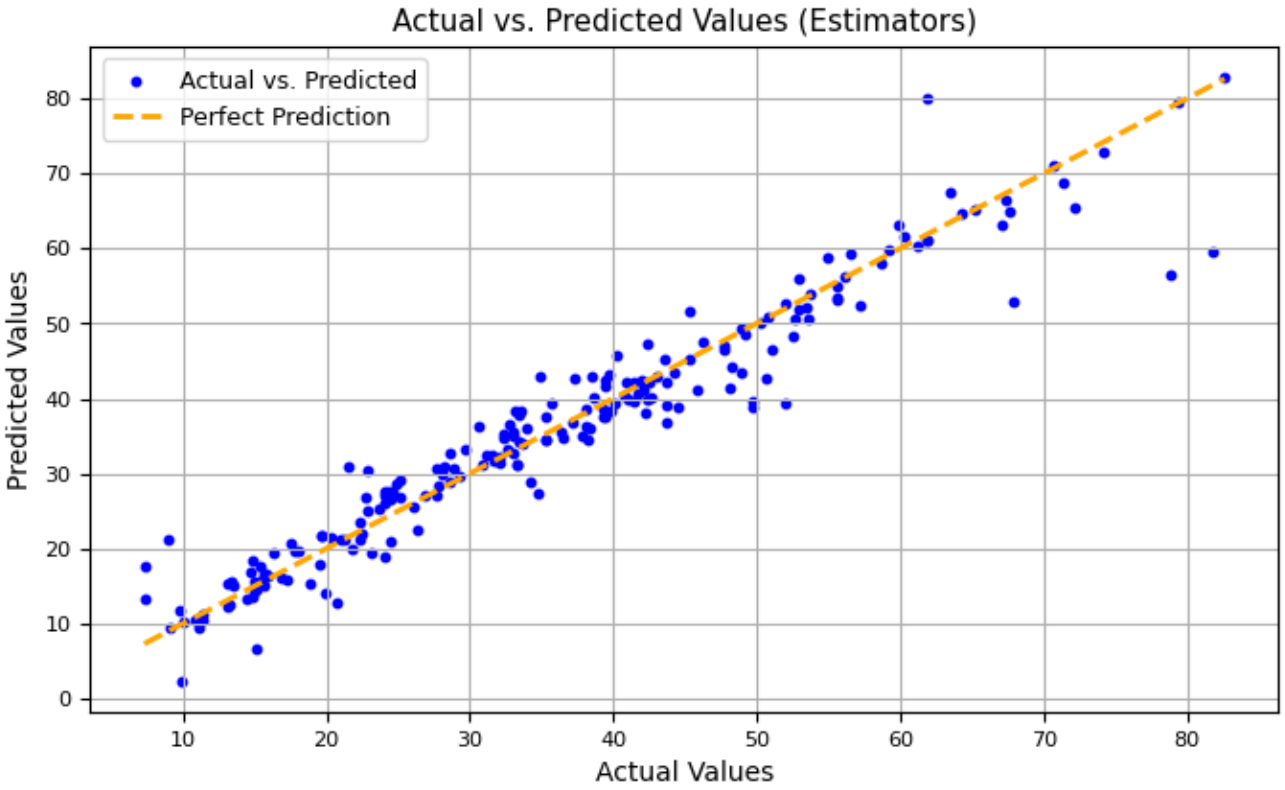

Figure 3: Optimised Estimators (see Table 2, R^2=0.93)

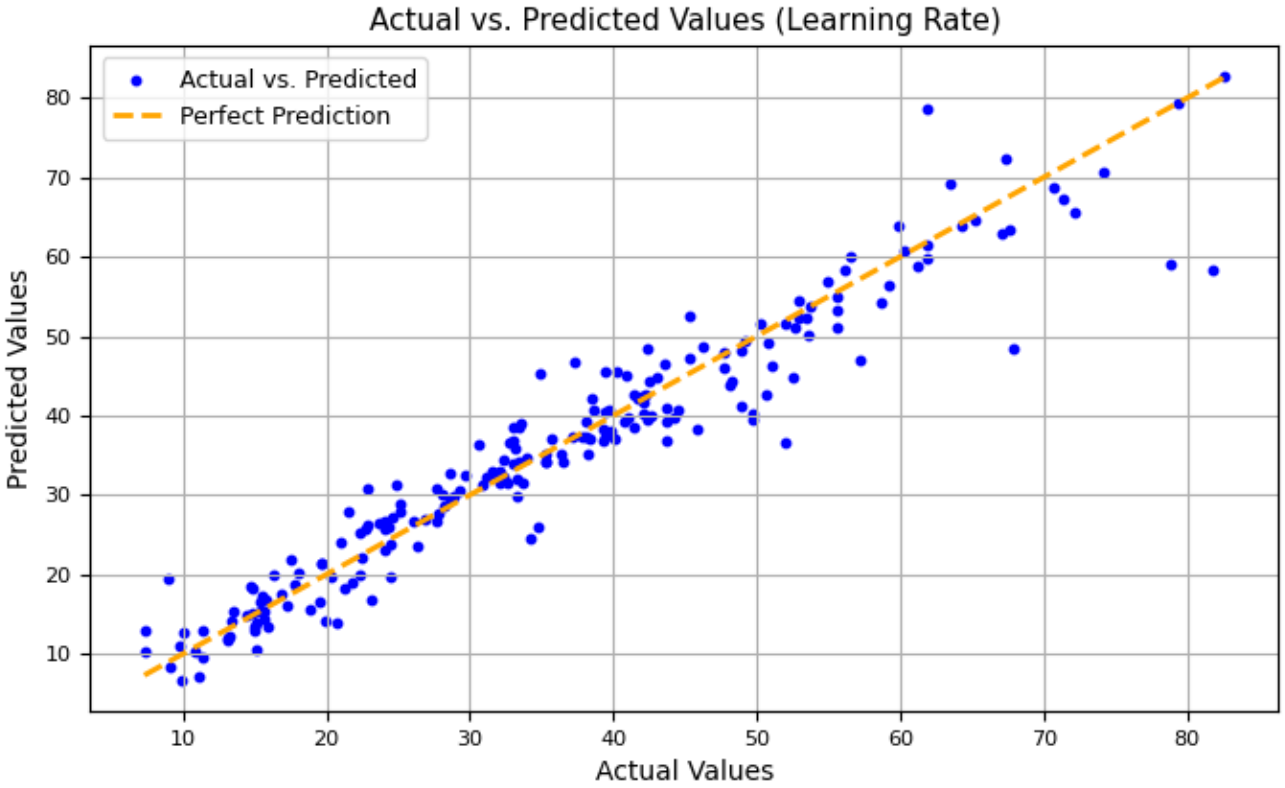

Figure 4: Optimised Learning Rate (see Table 2, R^2=0.92)

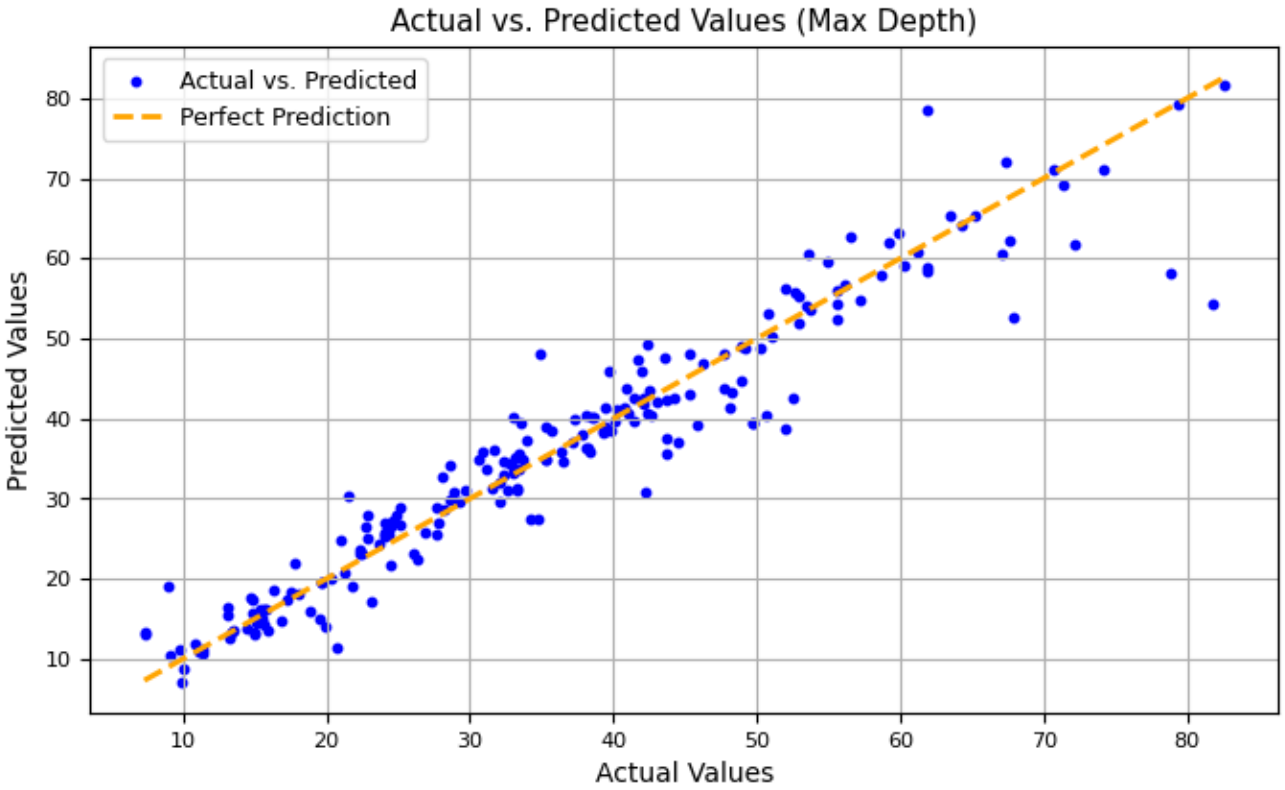

Figure 5: Optimised Max Depth (see Table 2, R^2 =0.92)

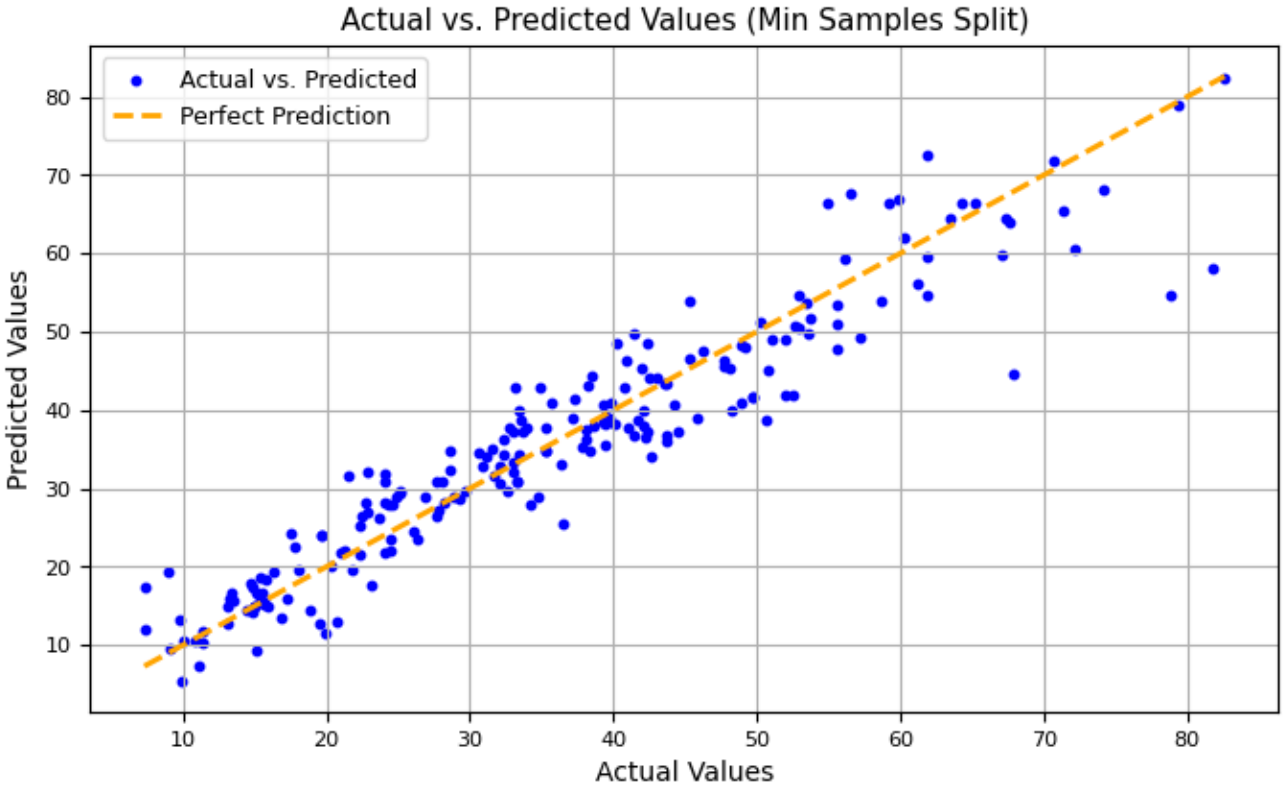

Figure 6: Optimised Min Samples Split (see Table 2, R^2 =0.90)

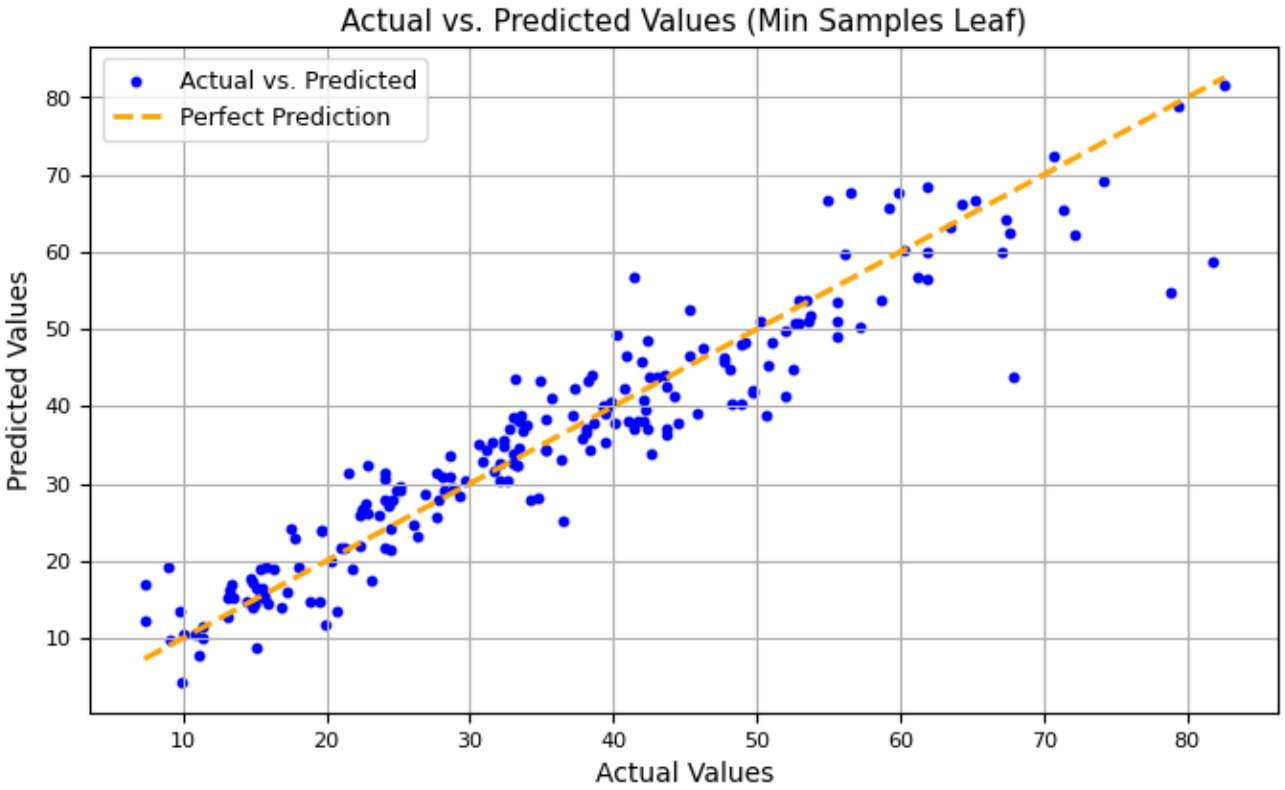

Figure 7: Optimised Min Samples Leaf (see Table 2, R^2=0.90)

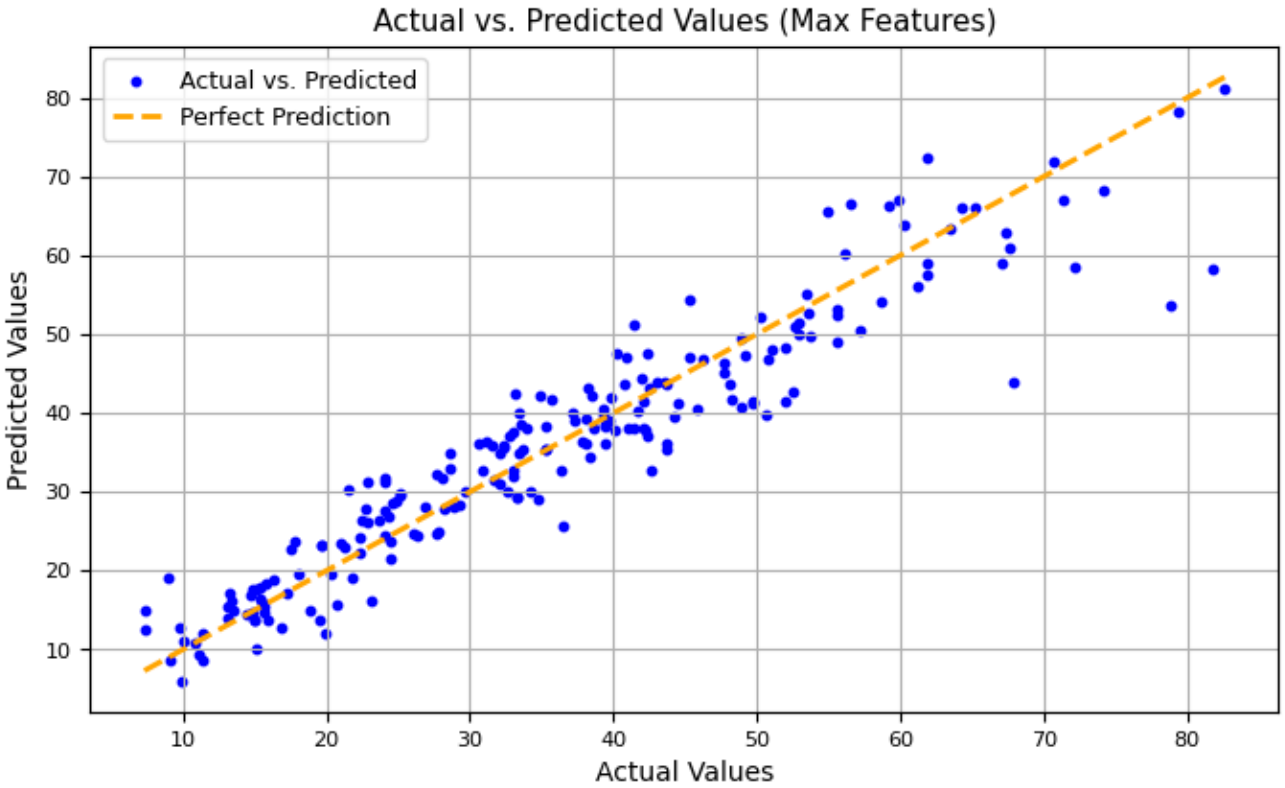

Figure 8: Optimised Max Features (see Table 2, R^2=0.90)

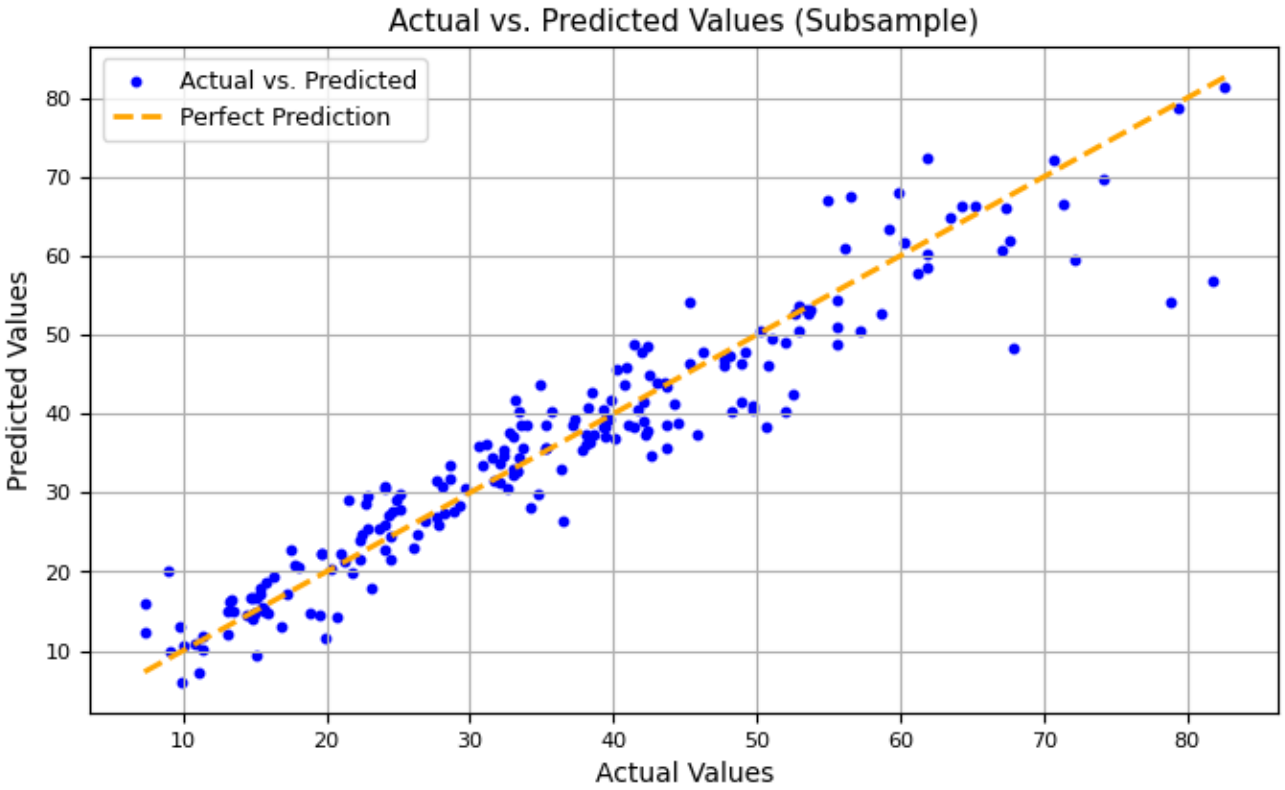

Figure 9: Optimised Subsample (see Table 2, R^2=0.91)

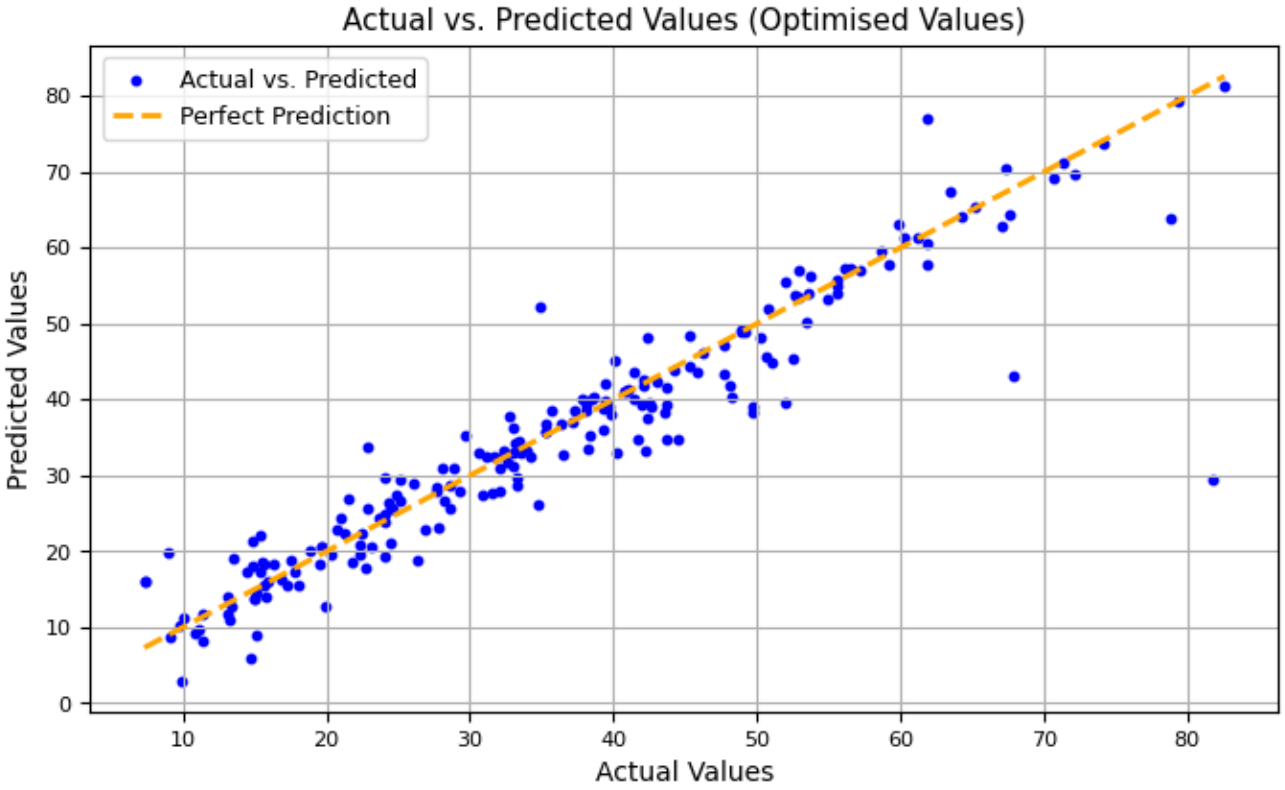

Figure 10: All Optimised Values (see Table 2, R^2=0.88)

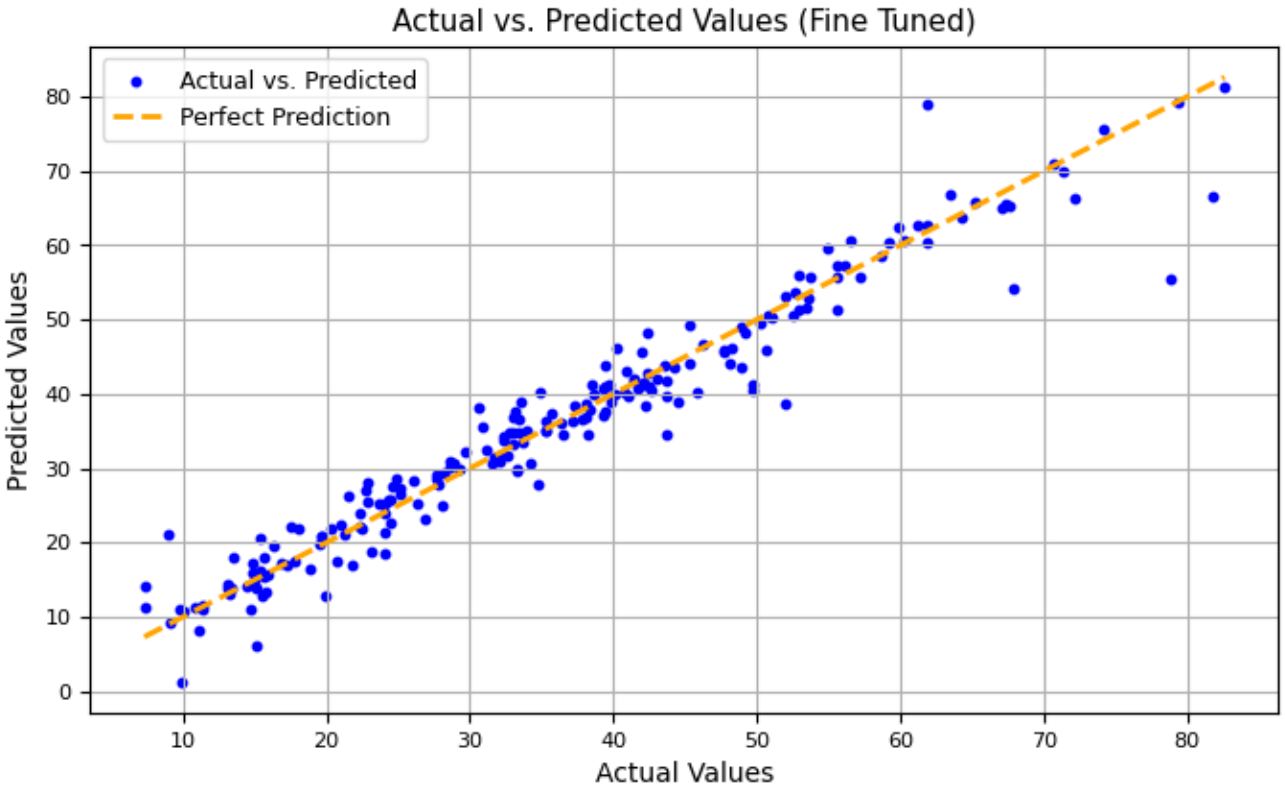

Figure 11: Fined Tuned Values (see Table 3, R^2=0.94)

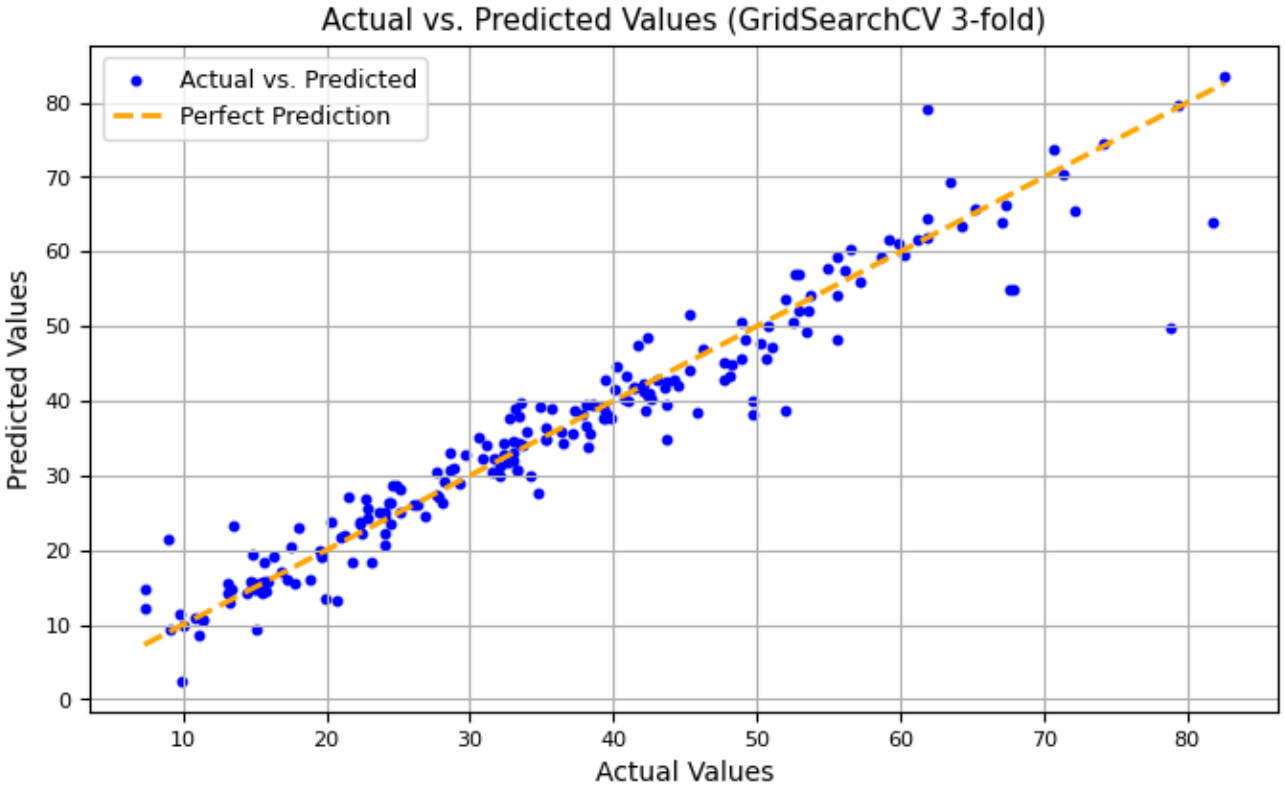

Figure 12: Grid Search CV 3-fold (see Table 5, R^2=0.93)

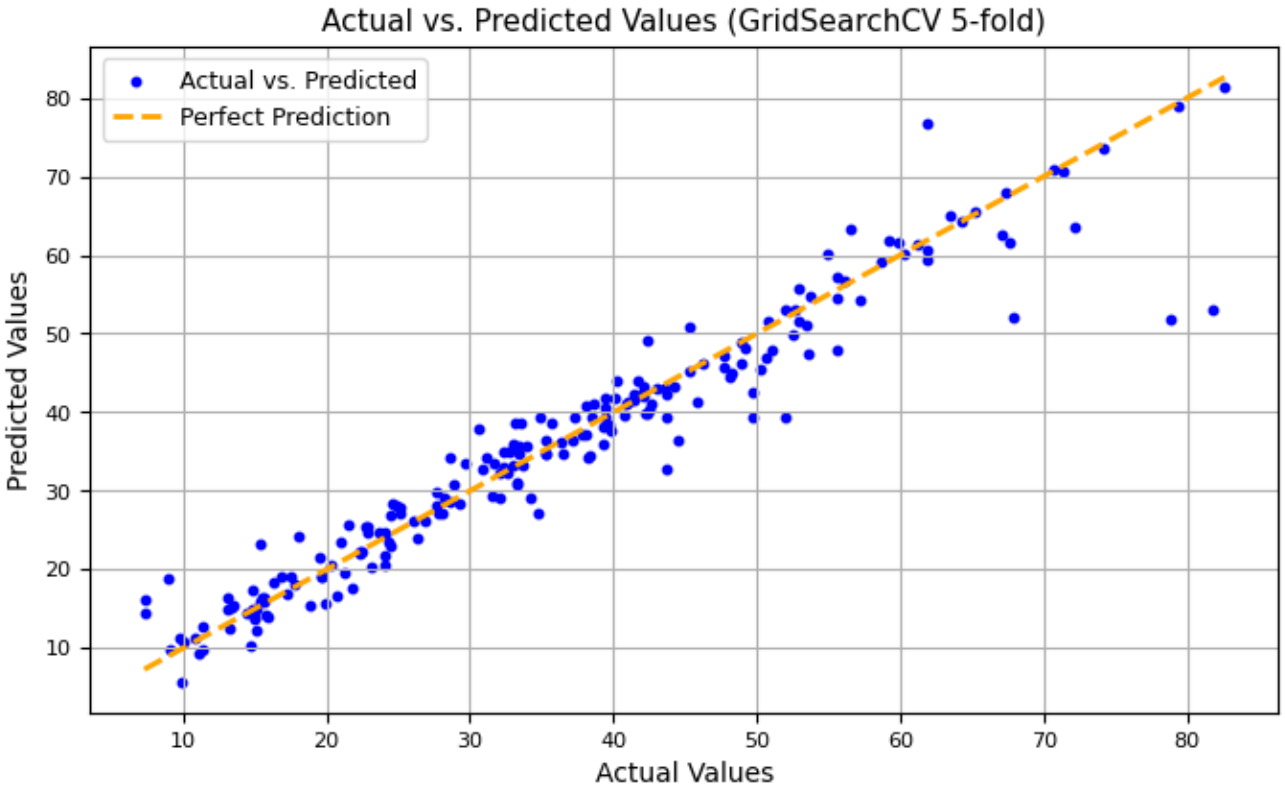

Figure 13: Grid Search CV 5-fold (see Table 6, R^2=0.93)

Runtime Evaluation

In addition to evaluating error, it is important to consider the runtime performance of manual versus automated approaches. Where manual tuning does not take much runtime, it does, however, involve trial and error strategies to determine the best hyperparameters. On the other hand, Grid Search CV utilises significantly more runtime and computational effort.

6. CONCLUSION

This study demonstrated that gradient boosting models improve the accuracy of predictions when predicting the compressive strength of high-performance concrete when hyperparameters are properly optimised. Using an 80–20 train-test split, when using default settings, the model achieved R^2 = 0.89, while targeted adjustments produced progressively better results. Manual fine-tuning of all parameters, simultaneously, yielded the best overall performance. Automated Grid Search CV 3-fold and 5-fold also substantially improved accuracy compared to default settings, but 5-fold had slightly inferior results when compared to 3-fold. Hyperparameter optimisation evidently reduces prediction error and boosts the coefficient of determination. The number of estimators had the largest individual impact on predictability, matching the gains seen under Grid Search CV. Manual tuning could edge out exhaustive grid search in a constrained parameter space due to the high runtimes, but required substantial trial and error effort. Further research suggests comparing alternative tuning approaches such as Randomised Search CV, Bayesian optimisation, and Optuna against manual tuning.

7. REFERENCES

[1] Lan, W. (2024). Accurate Compressive Strength Prediction Using Machine Learning Algorithms and Optimization Techniques. *Journal of Engineering and Applied Science*, 71(1). doi:https://doi.org/10.1186/s44147-023-00326-1.

[2] Smarzewski, P. and Stolarski, A. (2022). Properties and Performance of Concrete Materials and Structures. *Crystals*, [online] 12(9), p.1193. doi:https://doi.org/10.3390/cryst12091193.

[3] SABS (2006). *SANS 5863:2006. Concrete tests - compressive strength of hardened concrete*. 2.1 ed. Pretoria: SABS Standards Division.

[4] Owens, G. and Fulton, F. (2009). *Fulton's Concrete Technology*. 9th ed. Midrand: Cement & Concrete Institute.

[5] Sarker, I.H. (2021). Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science*, 2(3), pp.1–21. doi:https://doi.org/10.1007/s42979-021-00592-x

[6] Sutton, R. and Barto, A. (1998). *Introduction to Reinforcement Learning*. Cambridge, Mass: Mit Press.

[7] Goodfellow, I., Bengio, Y. and Courville, A. (2016). *Deep Learning*. Cambridge, Massachusetts: The MIT Press.

[8] Endut, N., W. Hamzah, W.M.A.F., Ismail, I., Kamir Yusof, M., Abu Baker, Y. and Yusoff, H. (2022). A Systematic Literature Review on Multi-Label Classification Based on Machine Learning Algorithms. *TEM Journal*, 11(2), pp.658–666. doi:https://doi.org/10.18421/tem112-20.

[9] Bhagat, M. and Bakariya, B. (2025). A Comprehensive Review of Cross-Validation Techniques in Machine Learning. *International Journal on Science and Technology*, 16(1). doi:https://doi.org/10.71097/ijsat.v16.i1.1305.

[10] Xu, Y. and Goodacre, R. (2018). On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the

Generalization Performance of Supervised Learning. *Journal of Analysis and Testing*, 2(3), pp.249–262. doi:https://doi.org/10.1007/s41664-018-0068-2

[11] Tan, J., Yang, J., Wu, S., Chen, G. and Zhao, J. (2021). *A Critical Look at the Current train/test Split in Machine Learning*. [online] arXiv.org. doi:https://doi.org/10.48550/arXiv.2106.04525.

[12] Jamal, A.S. and Ahmed, A.N. (2025). Estimating Compressive Strength of high-performance Concrete Using Different Machine Learning Approaches. *Alexandria Engineering Journal*, 114(1), pp.256–265. doi:https://doi.org/10.1016/j.aej.2024.11.084.

[13] Friedman, J.H. (2001). Greedy Function approximation: a Gradient Boosting Machine. *The Annals of Statistics*, 29(5), pp.1189–1232. doi:https://doi.org/10.1214/aos/1013203451.

[14] Chen, T. and Guestrin, C. (2016). XGBoost: a Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, 1(1), pp.785–794. doi:https://doi.org/10.1145/2939672.2939785.

[15] Buddiga, S. (2021). Optimizing Machine Learning Models: A Comprehensive Overview of Hyperparameter Tuning Techniques. *The Journal of Scientific and Engineering Research*, 8(2), pp.269–273. doi:https://doi.org/10.5281/zenodo.11216339.

[16] Choudhary, P. (2020). *Predicting the Strength of Concrete*. [online] Kaggle.com. Available at: https://www.kaggle.com/code/pritech/predicting-the-strength-of-concrete/input [Accessed 22 Apr. 2025].

[17] Scikit-learn (2019a). *sklearn.model_selection.GridSearchCV — scikit-learn 0.22 Documentation*. [online] Scikit-learn.org. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html [Accessed 1 Jul. 2025].

[18] Scikit-learn (2019b). *sklearn.model_selection.RandomizedSearchCV — scikit-learn 0.21.3 documentation*. [online] Scikit-learn.org. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html.

[19] Akiba, T., Sano, S., Yanase, T., Ohta, T. and Koyama, M. (2019). Optuna. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1(1). doi:https://doi.org/10.1145/3292500.3330701.

[20] Scikit-optimize (2017). *skopt.BayesSearchCV — scikit-optimize 0.8.1 Documentation*. [online] scikit-optimize.github.io. Available at: https://scikit-optimize.github.io/stable/modules/generated/skopt.BayesSearchCV.html.

[21] Prasanna, C. (2024). *Hyperparameter Tuning Explained — Manual tuning, GridSearchCV and RandomizedSearchCV*. [online] Medium. Available at: https://medium.com/@chanakapinfo/hyperparameter-tuning-explained-manual-tuning-gridsearchcv-and-randomizedsearchcv-bf0308162ebb [Accessed 1 Jul. 2025].

[22] ML Journey (2025). *Hyperparameter Tuning Methods: Comprehensive Comparison - ML Journey*. [online] ML Journey. Available at: https://mljourney.com/hyperparameter-tuning-methods-comprehensive-comparison/ [Accessed 1 Jul. 2025].