

## Model Optimization and Tuning Phase Template

Date	10 July 2024
Team ID	739916
Project Title	Predicting the Compressive Strength of Concrete
Maximum Marks	10 Marks

### Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

### Hyperparameter Tuning Documentation (6 Marks):

Model	Tuned Hyperparameters	Optimal Values
Linear Regression	No Hyperparameters used	-----
Gradient Boosting Regressor	No Hyperparameters used	-----

Model	Accuracy	Metrics
Linear Regression	<pre>lr = LinearRegression() lr.fit(x_train, y_train)</pre> <div> <div>LinearRegression</div> <div>LinearRegression()</div> </div> <pre>lr.score(x_train, y_train) lr.score(x_test, y_test)</pre> <p>0.7459334448168664</p>	<pre>ypred=lr.predict(x_test)</pre> <div> <div>from sklearn.linear_model import LinearRegression</div> <div>lr=LinearRegression()</div> <div>lr.fit(x_train, y_train)</div> <div>mse=mean_squared_error(ypred, y_test)</div> <div>print("mean squared error:", mse)</div> <div>r2_lr=r2_score(ypred, y_test)</div> <div>print("r2 score", r2_lr)</div> </div> <pre>mean squared error: 95.92548435337708 r2 score 0.42383938888034913</pre>

<p><b>Gradient Boosting Regressor</b></p>	<pre>from sklearn.ensemble import GradientBoostingRegressor  from sklearn.model_selection import train_test_split from sklearn.ensemble import GradientBoostingRegressor cols = df.columns.drop('Concrete compressive strength') x = df[cols] y = df['concrete compressive strength']  x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)  # Initialize the GradientBoostingRegressor gbr_model = GradientBoostingRegressor() # Use a different variable name gbr_model.fit(x_train, y_train)  gbr_model.score(x_train, y_train) gbr_model.score(x_test, y_test)</pre>	<pre>print("MAE", mean_absolute_error(y_test, ypred)) print("MSE", mean_squared_error(y_test, ypred)) print("R2", r2_score(y_test, ypred)) print("RMSE", np.sqrt(mean_squared_error(y_test, ypred)))  MAE 7.745392872421345 MSE 95.97548435337708 R2 0.8820699199800247 RMSE 9.796707832398447</pre>
---	---	--

### Performance Metrics Comparison Report (2 Marks):

### Final Model Selection Justification (2 Marks):

Final Model Selection	Reasoning
Gradient Boosting Regressor	<p>The Gradient Boosting model was selected for its superior performance, exhibiting high accuracy than linear regression.</p> <p>We chose the Gradient Boosting Regressor because it gives very accurate predictions, can handle complex patterns in data, and avoids overfitting. It works well with different types of data and allows us to see which features are most important. This makes it a reliable and effective model for our task</p>