

Model Optimization and Tuning Phase Template

Date	10 July 2024
Team ID	739842
Project Title	Trip-Based Modelling of Fuel Consumption in Modern Fleet Vehicles Using Machine Learning
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	-----
Lasso Regression	No Hyperparameters used	-----
SVM	No Hyperparameters used	-----
Decision Tree	No Hyperparameters used	-----
Random Forest	No Hyperparameters used	-----

Model	Accuracy	Metrics
Linear Regression	<pre>print(linReg.coef_,linReg.intercept_)</pre> <pre>[0.00523674 -0.02371772 -0.14711979 -0.03724498 0.41546804 0.61676684 -0.06407861] 9.389308142257129</pre> <pre>accuracy = linReg.score(x_test,y_test)</pre> <pre>print(accuracy)</pre> <pre>0.1134733714697449</pre>	<pre>[58] print("Prediction Evaluation using Linear Regression") print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred)) print('Mean Squared Error:', mean_squared_error(y_test, y_pred)) print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_pred))) print('R-squared:', r2_score(y_test, y_pred))</pre> <pre>Prediction Evaluation using Linear Regression Mean Absolute Error: 0.6635761182069623 Mean Squared Error: 0.742453260904708 Root Mean Squared Error: 0.8616572757800562 R-squared: 0.1134733714697449</pre>
Lasso Regression	<pre>accuracy = lassoReg.score(x_test,y_test)</pre> <pre>print(accuracy)</pre> <pre>0.1456141532515728</pre>	<pre>y_pred = lassoReg.predict(x_test) print("Prediction Evaluation using Lasso Regression") print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred)) print('Mean Squared Error:', mean_squared_error(y_test, y_pred)) print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_pred))) print('R-squared:', r2_score(y_test, y_pred))</pre> <pre>Prediction Evaluation using Lasso Regression Mean Absolute Error: 0.6296444264267669 Mean Squared Error: 0.7155358198781405 Root Mean Squared Error: 0.8458935038633058 R-squared: 0.1456141532515728</pre>
SVM	<pre>[43] #SVM MODEL</pre> <pre>[44] svr = SVR().fit(x,y)</pre> <pre>[45]</pre> <pre>y_pred = svr.predict(x_test)</pre> <pre>[46] accuracy = svr.score(x_test,y_test)</pre> <pre>print(accuracy)</pre> <pre>0.4176454053391483</pre>	<pre>0] y_pred = svr.predict(x_test) print("Prediction Evaluation using svr Regression") print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred)) print('Mean Squared Error:', mean_squared_error(y_test, y_pred)) print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_pred))) print('R-squared:', r2_score(y_test, y_pred))</pre> <pre>Prediction Evaluation using svr Regression Mean Absolute Error: 0.49633196595520446 Mean Squared Error: 0.48771357102448615 Root Mean Squared Error: 0.698364926828722 R-squared: 0.4176454053391483</pre>
Decision Tree	<pre>[48] dt = DecisionTreeRegressor(random_state = 0) dt.fit(x,y)</pre> <pre>[49] y_pred = dt.predict(x_test)</pre> <pre>print(y_pred)</pre> <pre>[5.6 5.1 5.55 4.6 4.3 7.1 5. 4.8 5.1 4.7 4.3 4.6 5.4 6.2 7.4 4.7 5.2 4.6 4.9 5. 6.9 4.6 4.9 4.6 4.8 5.2 6.6 4.8 4.5 4.2 4.9 5.2 5.8 4.1 4.9 4.4 4.1 4.6 5.8 5.4 5.4 4.9 4.3 5.5 5.5 4.2 0. 5. 5.8 4.9 5.9 4.9 5.5 5. 4.9 5.9 5.9 5.1 5.9 4.1 0.9 4.1 4.7 5.9 5.9 5.4 9.9 4.1 4. 5.2 5. 4.7 5.7 5.9 4.9 4.7 4.7 4.8 5.7 4.6 4.9 5.8 4.2 5.1 4.6 5.2 5. 4.6 4.2 4.1 5. 4. 5.4 4.9 4. 5.6 4.6 4.4 5.59 4.7 4.5 5. 5.7 5.7 4.4 4.5 4. 5.6 7.9 4. 5.9 7.9 6.1 5.7 5.1 5.1 4.1]</pre> <pre>accuracy = dt.score(x_test,y_test)</pre> <pre>print(accuracy)</pre> <pre>0.9864521202267205</pre>	<pre>y_pred = dt.predict(x_test) print("Prediction Evaluation using decisiontree Regression") print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred)) print('Mean Squared Error:', mean_squared_error(y_test, y_pred)) print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_pred))) print('R-squared:', r2_score(y_test, y_pred))</pre> <pre>Prediction Evaluation using decisiontree Regression Mean Absolute Error: 0.016666666666666666 Mean Squared Error: 0.011346153846153837 Root Mean Squared Error: 0.10651832633943249 R-squared: 0.9864521202267205</pre>

Random Forest	<pre>4.9075 4.945 4.295]</pre> <pre>accuracy = rf.score(x_test,y_test) print(accuracy)</pre> <pre>0.9354691820163654</pre>	<pre>y_pred = rf.predict(x_test) print("Prediction Evaluation using Random Regression") print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred)) print('Mean Squared Error:', mean_squared_error(y_test, y_pred)) print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_pred))) print('R-squared:', r2_score(y_test, y_pred))</pre> <pre>Prediction Evaluation using Random Regression Mean Absolute Error: 0.1625574074074077 Mean Squared Error: 0.05404362903371328 Root Mean Squared Error: 0.23247285655257321 R-squared: 0.9354691820163654</pre>
---------------	---	--

Performance Metrics Comparison Report (2 Marks):

Final Model Selection Justification (2 Marks):

Final Model Selection	Reasoning
Decision Tree	<p>Decision Tree model was selected for its superior performance, exhibiting high accuracy than any other models .</p> <p>We chose the decision tree 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>