

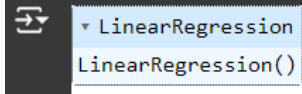
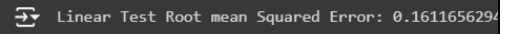
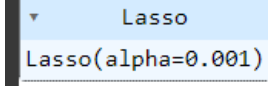
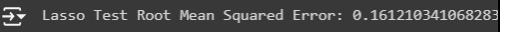
Model Optimization and Tuning Phase Template

Date	4 July 2024
Team ID	SWTID1720090524
Project Title	Garment Worker Productivity Prediction
Maximum Marks	10 Marks

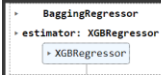
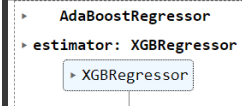
Model Optimization and Tuning Phase

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 Model	<pre>[83] lin_reg = LinearRegression() lin_reg.fit(x_train,y_train)</pre> 	<pre>[47] # testing score mse_test = mean_squared_error(y_test, y_pred_test) rmse_test = np.sqrt(mse_test) print('Linear Test Root mean Squared Error:',rmse_test)</pre> 
Lasso Regression Model	<pre>lasso = Lasso(alpha = 0.001) lasso.fit(x_train, y_train)</pre> 	<pre># testing score mse_test = mean_squared_error(y_test, y_pred_test) rmse_test = np.sqrt(mse_test) print('Lasso Test Root Mean Squared Error:', rmse_test)</pre> 

<p>Ridge Regression Model</p>	<pre>ridge = Ridge(alpha = 1.9) ridge.fit(x_train, y_train)</pre> <p>Ridge Ridge(alpha=1.9)</p>	<pre># testing score mse_test = mean_squared_error(y_test, y_pred_test) rmse_test = np.sqrt(mse_test) print('Ridge Test Root mean Squared Error:', rmse_test)</pre> <p>Ridge Test Root mean Squared Error: 0.1611521889</p>
<p>Decision Tree Regressor Model</p>	<pre>dtr = DecisionTreeRegressor(max_depth= 4, min_samples_split= 3, min_samples_leaf= 2) dtr.fit(x_train,y_train)</pre> <p>DecisionTreeRegressor DecisionTreeRegressor(max_depth=4, min_samples_leaf=2, min_samples_split=3)</p>	<pre>[59] # testing score mse_test = mean_squared_error(y_test, y_pred_test) rmse_test = np.sqrt(mse_test) print('Decision Tree Test Root mean Squared Error:', rmse_test)</pre> <p>Decision Tree Test Root mean Squared Error: 0.12918875</p>
<p>Random forest Regressor Model</p>	<pre>rfr = RandomForestRegressor(n_estimators=100, max_depth=6, min_weight_fraction_leaf=0.05, max_features=0.8, random_state=42) rfr.fit(x_train,y_train)</pre> <p>RandomForestRegressor RandomForestRegressor(max_depth=6, max_features=0.8, min_weight_fraction_leaf=0.05, random_state=42)</p>	<pre>[65] # testing score mse_test = mean_squared_error(y_test, y_pred_test) rmse_test = np.sqrt(mse_test) print('Random Forest Test Root mean Squared Error:', rmse_test)</pre> <p>Random Forest Test Root mean Squared Error: 0.1272406</p>
<p>Gradient Boosting Regressor Model</p>	<pre>gbr = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, max_depth=1, random_state=42) gbr.fit(x_train,y_train)</pre> <p>GradientBoostingRegressor GradientBoostingRegressor(max_depth=1, random_state=42)</p>	<pre>[69] # testing score mse_test = mean_squared_error(y_test, y_pred_test) rmse_test = np.sqrt(mse_test) print('Gradient Boosting Test Root mean Squared Error:', rmse_test)</pre> <p>Gradient Boosting Test Root mean Squared Error: 0.1395308</p>
<p>Extreme Gradient Boost Regressor</p>	<pre>xgb = XGBRegressor (n_estimators=300, learning_rate=0.05 max_leaves = 3, random_state = 1) xgb.fit(x_train, y_train)</pre> <p>XGBRegressor XGBRegressor(base_score=None, booster=None, callbacks=None, colsample_bylevel=None, colsample_bynode=None, colsample_bynode=None, device=None, early_stopping_rounds=None, enable_categorical=False, eval_metric=None, feature_types=None, gamma=None, grow_policy=None, importance_type=None, interaction_constraints=None, learning_rate=0.05, max_bin=None, max_cat_threshold=None, max_cat_to_onehot=None, max_delta_step=None, max_depth=None, max_leaves=3, min_child_weight=None, missing=nan, monotone_constraints=None, multi_strategy=None, n_estimators=300, n_jobs=None, num_parallel_tree=None, random_state=1, ...)</p>	<pre># testing score mse_test = mean_squared_error(y_test, y_pred_test) rmse_test = np.sqrt(mse_test) print('Extreme Gradient Boosting Test Root mean Squared Error:', rmse_test)</pre> <p>Extreme Gradient Boosting Test Root mean Squared Error: 0.1208573039635658</p>

Bagging Regressor Model	<pre># Define base model base_model = XGBRegressor(n_estimators=700, learning_rate=0.06, max_depth=2, max_leaves=3, random_state=1) # Create bagging regressor bagging_reg = BaggingRegressor(base_model, n_estimators=100, random_state=42) # Fit bagging regressor bagging_reg.fit(x_train, y_train)</pre> 	<pre>[77] # testing score mse_test = mean_squared_error(y_test, y_pred_test) rmse_test = np.sqrt(mse_test) print('Bagging Test Root mean Squared Error:', rmse_test)</pre> <pre>Bagging Test Root mean Squared Error: 0.11683354</pre>
Boosting Regressor Model	<pre># Define base model base_model = XGBRegressor(n_estimators=700, learning_rate=0.06, max_depth=2, max_leaves=3, random_state=1) # Create AdaBoost regressor boosting_reg = AdaBoostRegressor (base_model, n_estimators=100, learning_rate=0.1, random_state=42) # Fit AdaBoost regressor boosting_reg.fit(x_train, y_train)</pre> 	<pre>[82] # testing score mse_test = mean_squared_error(y_test, y_pred_test) rmse_test = np.sqrt(mse_test) print('Boosting Test Root mean Squared Error:', rmse_test)</pre> <pre>Boosting Test Root mean Squared Error: 0.12712298</pre>

Performance Metrics Comparison Report (2 Marks):

Model	Baseline Metric	Optimized Metric
Linear Regression	--	<pre>[85] # training score mse_train = mean_squared_error(y_train, y_pred_train) rmse_train = np.sqrt(mse_train) print('Linear Train Root Mean Squared Error:', rmse_train)</pre> <pre>Linear Train Root Mean Squared Error: 0.16226529653729893</pre> <pre>[86] # testing score mse_test = mean_squared_error(y_test, y_pred_test) rmse_test = np.sqrt(mse_test) print('Linear Test Root mean Squared Error:',rmse_test)</pre> <pre>Linear Test Root mean Squared Error: 0.16116562949494234</pre>

Lasso Regression	--	<pre># training score mse_train = mean_squared_error(y_train, y_pred_train) rmse_train = np.sqrt(mse_train) print('Lasso Train Root Mean Squared Error:', rmse_train) Lasso Train Root Mean Squared Error: 0.16246420183571206 # testing score mse_test = mean_squared_error(y_test, y_pred_test) rmse_test = np.sqrt(mse_test) print('Lasso Test Root Mean Squared Error:', rmse_test) Lasso Test Root Mean Squared Error: 0.16121034106828316</pre>
Ridge Regression	--	<pre># training score mse_train = mean_squared_error(y_train, y_pred_train) rmse_train = np.sqrt(mse_train) print('Ridge Train Root Mean Squared Error:', rmse_train) Ridge Train Root Mean Squared Error: 0.16226837609384914 # testing score mse_test = mean_squared_error(y_test, y_pred_test) rmse_test = np.sqrt(mse_test) print('Ridge Test Root mean Squared Error:', rmse_test) Ridge Test Root mean Squared Error: 0.16115218890295427</pre>
Decision Tree Regressor	--	<pre># training score mse_train = mean_squared_error(y_train, y_pred_train) rmse_train = np.sqrt(mse_train) print('Decision Tree Train Root Mean Squared Error:', rmse_train) Decision Tree Train Root Mean Squared Error: 0.13187559206436333 # testing score mse_test = mean_squared_error(y_test, y_pred_test) rmse_test = np.sqrt(mse_test) print('Decision Tree Test Root mean Squared Error:', rmse_test) Decision Tree Test Root mean Squared Error: 0.12918875831022705</pre>

Random Forest Regressor	--	<pre># training score mse_train = mean_squared_error(y_train, y_pred_train) rmse_train = np.sqrt(mse_train) print('Random Forest Train Root Mean Squared Error:', rmse_train) Random Forest Train Root Mean Squared Error: 0.13062916799216504 # testing score mse_test = mean_squared_error(y_test, y_pred_test) rmse_test = np.sqrt(mse_test) print('Random Forest Test Root mean Squared Error:', rmse_test) Random Forest Test Root mean Squared Error: 0.1272406384012021</pre>
Gradient Boosting Regressor	--	<pre># training score mse_train = mean_squared_error(y_train, y_pred_train) rmse_train = np.sqrt(mse_train) print('Gradient Boosting Train Root Mean Squared Error:', rmse_train) Gradient Boosting Train Root Mean Squared Error: 0.1424427737607694 # testing score mse_test = mean_squared_error(y_test, y_pred_test) rmse_test = np.sqrt(mse_test) print('Gradient Boosting Test Root mean Squared Error:', rmse_test) Gradient Boosting Test Root mean Squared Error: 0.13953081336729123</pre>
Extreme Gradient Boost Regressor	--	<pre># training score mse_train = mean_squared_error(y_train, y_pred_train) rmse_train = np.sqrt(mse_train) print('Extreme Gradient Boosting Train Root Mean Squared Error:', rmse_train) Extreme Gradient Boosting Train Root Mean Squared Error: 0.12323119231675213 # testing score mse_test = mean_squared_error(y_test, y_pred_test) rmse_test = np.sqrt(mse_test) print('Extreme Gradient Boosting Test Root mean Squared Error:', rmse_test) Extreme Gradient Boosting Test Root mean Squared Error: 0.12085573039635658</pre>

Bagging Regressor	--	<pre># training score mse_train = mean_squared_error(y_train, y_pred_train) rmse_train = np.sqrt(mse_train) print('Bagging Train Root Mean Squared Error:', rmse_train) Bagging Train Root Mean Squared Error: 0.11512255799809959 # testing score mse_test = mean_squared_error(y_test, y_pred_test) rmse_test = np.sqrt(mse_test) print('Bagging Test Root mean Squared Error:', rmse_test) Bagging Test Root mean Squared Error: 0.11683354248554284</pre>
Boosting Regressor	--	<pre># training score mse_train = mean_squared_error(y_train, y_pred_train) rmse_train = np.sqrt(mse_train) print('Boosting Train Root Mean Squared Error:', rmse_train) Boosting Train Root Mean Squared Error: 0.11456846365002314 # testing score mse_test = mean_squared_error(y_test, y_pred_test) rmse_test = np.sqrt(mse_test) print('Boosting Test Root mean Squared Error:', rmse_test) Boosting Test Root mean Squared Error: 0.12712298242901834</pre>

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Bagging Regressor Model	The Bagging Regressor Model was selected for its superior performance, exhibiting high accuracy during hyperparameter tuning. Its ability to handle complex relationships, minimize overfitting, and optimize predictive accuracy aligns with project objectives, justifying its selection as the final model.