Model Optimization and Tuning Phase Template

Date	15 July 2024
Team ID	SWTID1720160264
Project Title	Predicting Compressive Strength Of Concrete 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

Gradient Boosting Regression. (Randomized searchcv)	`from sklearn.model_selection import RandomizedSearchCV`
	`from scipy.stats import uniform, randint`
	`gb = GradientBoostingRegressor()`
	`param_dist = {`
	`'n_estimators': randint(100, 500),`
	`'learning_rate': uniform(0.01, 0.2),`
	`'max_depth': randint(1, 10),`
	`'min_samples_split': randint(2, 20),`
	`'min_samples_leaf': randint(1, 20),`
	`'subsample': uniform(0.6, 0.4),`
	`'max_features': ['auto', 'sqrt', 'log2', None]`
	`}`
	`random_search = RandomizedSearchCV(estimator=gb, param_distributions=param_dist,`
	`n_iter=100, cv=5, verbose=2, random_state=42, n_jobs=-1)`
	`random_search.fit(x_train, y_train)`
	`best_gb = random_search.best_estimator_`
	`y_pred = best_gb.predict(x_test)`
		`r2=r2_score(y_pred,y_test)`
		`r2`
		
		`0.9434708311366856`

Gradient Boosting Regression.
 	
(Gridsearchcv)
	GradientBoostingRegressor using GridSearchCV
	
	`from sklearn.model_selection import GridSearchCV`
	
	`from sklearn.ensemble import GradientBoostingRegressor`
	
	`gb = GradientBoostingRegressor()`
	
	`param_grid = {`
	`'n_estimators': [100, 200, 300],`
	`'learning_rate': [0.01, 0.1, 0.05],`
	`'max_depth': [3, 4, 5],`
	`'min_samples_split': [2, 5, 10],`
	`'min_samples_leaf': [1, 2, 4]`
	`}`
	
	`# Initialize GridSearchCV`
	`grid_search = GridSearchCV(estimator=gb, param_grid=param_grid, cv=5, n_jobs=-1, verbose=2)`
	
	`# Fit the model`
	`grid_search.fit(x_train, y_train)`
		`best_gb = grid_search.best_estimator_`
		`y_pred0 = best_gb.predict(x_test)`
		
		`grid_search.best_score_`
		
		0.916712297190122

XGBoost Regression
 	
(Randomized searchcv)
	XGBRegressor using RandomizedSearchCV
	
	`xgb_reg = XGBRegressor()`
	
	`# Define the parameter distribution`
	`param_dist = {`
	`'n_estimators': randint(100, 500),`
	`'learning_rate': uniform(0.01, 0.2),`
	`'max_depth': randint(3, 10),`
	`'min_child_weight': randint(1, 10),`
	`'subsample': uniform(0.5, 0.5),`
	`'colsample_bytree': uniform(0.5, 0.5)`
	`}`
	
	`# Initialize RandomizedSearchCV`
	`random_search = RandomizedSearchCV(estimator=xgb_reg, param_distributions=param_dist,`
	`n_iter=100, cv=5, verbose=2, random_state=42, n_jobs=-1)`
	
	`# Fit the model`
	`random_search.fit(x_train, y_train)`
		`best_xgb_reg = random_search.best_estimator_`
		`y_pred = best_xgb_reg.predict(x_test)`
		
		`r2 = r2_score(y_test, y_pred)`
		`r2`
		
		0.940805163311364

XGBoost Regression
 	
(Gridsearchcv)
	XGBRegressor using GridSearchCV
	
	`from xgboost import XGBRegressor`
	`# Initialize the XGBRegressor`
	`xgb = XGBRegressor()`
	
	`# Define the parameter grid`
	`param_grid = {`
	`'n_estimators': [100, 200, 300],`
	`'learning_rate': [0.01, 0.1, 0.05],`
	`'max_depth': [3, 4, 5],`
	`'min_child_weight': [1, 2, 3],`
	`'subsample': [0.6, 0.8, 1.0],`
	`'colsample_bytree': [0.6, 0.8, 1.0]`
	`}`
	
	`# Initialize GridSearchCV`
	`grid_search = GridSearchCV(estimator=xgb, param_grid=param_grid, cv=5, n_jobs=-1, verbose=2)`
	
	`# Fit the model`
	`grid_search.fit(x_train, y_train)`
		`best_xgb = grid_search.best_estimator_`
		`y_pred = best_xgb.predict(x_test)`
		
		`grid_search.best_score_`
		
		0.9225956483105566

RandomForest Regression (Randomized searchcv)	RandomForestRegressor Using RandomizedSearchCV
	`from sklearn.ensemble import RandomForestRegressor`
	`from sklearn.model_selection import RandomizedSearchCV`
	`from scipy.stats import randint`
	`rfr = RandomForestRegressor()`
	`param_dist = { 'n_estimators': randint(100, 1000), 'max_depth': randint(1, 20), 'min_samples_split': randint(2, 20), 'min_samples_leaf': randint(1, 20), 'bootstrap': [True, False] }`
	`random_search = RandomizedSearchCV(estimator=rfr, param_distributions=param_dist, n_iter=100, cv=5, verbose=2, random_state=42, n_jobs=-1)`
	`random_search.fit(x_train, y_train)`
		`best_rfr = random_search.best_estimator_`
		`y_pred = best_rfr.predict(x_test)`
		`r2 = r2_score(y_test, y_pred)`
		`r2`
		0.8978707797134569

RandomForest Regression (Gridsearchcv)	`rfr = RandomForestRegressor()`
	`param_grid = { 'n_estimators': [100, 200, 300, 400, 500], 'max_depth': [None, 10, 20, 30, 40, 50], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4], 'bootstrap': [True, False] }`
	`grid_search = GridSearchCV(estimator=rfr, param_grid=param_grid, cv=5, n_jobs=-1, verbose=2)`
	`# Fit the model`
	`grid_search.fit(x_train, y_train)`
		`# Evaluate the model with the test set`
		`best_rfr = grid_search.best_estimator_`
		`y_pred = best_rfr.predict(x_test)`
		`r2 = r2_score(y_test, y_pred)`
		`r2`
		0.908834984084111

Performance Metrics Comparison Report (2 Marks):

Model	Baseline Metric	Optimized Metric

Gradient Boosting Regression. (Randomized searchcv)	`score=r2_score(y_pred,y_test)`
	`print("R2 score: ",score*100)`
	R2 score: 88.81108653166656
	`print("MSE: ",metrics.mean_squared_error(y_pred,y_test))`
	MSE: 27.01177924274266
	`print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_pred,y_test)))`
	RMSE: 5.197285757272026
		`print("MAE: ",metrics.mean_absolute_error(y_pred,y_test))`
		`print("MSE: ",metrics.mean_squared_error(y_pred,y_test))`
		`print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_pred,y_test)))`
		MAE: 2.646201211431639
		MSE: 14.909267426711827
		RMSE: 3.8612520542839244

Gradient Boosting Regression.
 	
(Gridsearchcv)
	`score=r2_score(y_pred,y_test)`
	`print("R2 score: ",score*100)`
	R2 score: 88.81108653166656
	`print("MSE: ",metrics.mean_squared_error(y_pred,y_test))`
	MSE: 27.01177924274266
	`print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_pred,y_test)))`
	RMSE: 5.197285757272026
		`print("MAE: ",metrics.mean_absolute_error(y_pred,y_test))`
		`print("MSE: ",metrics.mean_squared_error(y_pred,y_test))`
		`print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_pred,y_test)))`
		MAE: 3.0454332544375693
		MSE: 20.446855694652488
		RMSE: 4.521819953807591

XGBoost Regression
 	
(Randomized searchcv)
	`print("R2 score: ",score*100)`
	R2 score: 93.11918057409176
	`print("MSE: ",metrics.mean_squared_error(y_pred,y_test))`
	MSE: 20.446855694652488
	`print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_pred,y_test)))`
	RMSE: 4.521819953807591
		`print("MAE: ",metrics.mean_absolute_error(y_pred,y_test))`
		`print("MSE: ",metrics.mean_squared_error(y_pred,y_test))`
		`print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_pred,y_test)))`
		MAE: 2.9416536297431355
		MSE: 17.590176528739523
		RMSE: 4.1940644402225775

XGBoost Regression
 	
(Gridsearchcv)
	`print("R2 score: ",score*100)`
	R2 score: 93.11918057409176
	`print("MSE: ",metrics.mean_squared_error(y_pred,y_test))`
	MSE: 20.446855694652488
	`print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_pred,y_test)))`
	RMSE: 4.521819953807591
		`print("MAE: ",metrics.mean_absolute_error(y_pred,y_test))`
		`print("MSE: ",metrics.mean_squared_error(y_pred,y_test))`
		`print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_pred,y_test)))`
		MAE: 2.8977640539316036
		MSE: 17.411553524793284
		RMSE: 4.172715365897042

RandomForest Regression
 	
(Randomized searchcv)
	`score=r2_score(y_test,pred)`
	`print('R2 score of Random Forest Regression:',score*100)`
	R2 score of Random Forest Regression: 91.08323808304432
	`print("MSE: ",metrics.mean_squared_error(pred,y_test))`
	MSE: 26.496806978116595
	`print("RMSE: ",np.sqrt(metrics.mean_squared_error(pred,y_test)))`
	RMSE: 5.147504927449472
		`print("MAE: ",metrics.mean_absolute_error(y_pred,y_test))`
		`print("MSE: ",metrics.mean_squared_error(y_pred,y_test))`
		`print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_pred,y_test)))`
		MAE: 3.9719623149595416
		MSE: 30.34844108164759
		RMSE: 5.508941920337116

RandomForest Regression
 	
(Gridsearchcv)
	`score=r2_score(y_test,pred)`
	`print('R2 score of Random Forest Regression:',score*100)`
	R2 score of Random Forest Regression: 91.08323808304432
	`print("MSE: ",metrics.mean_squared_error(pred,y_test))`
	MSE: 26.496806978116595
	`print("RMSE: ",np.sqrt(metrics.mean_squared_error(pred,y_test)))`
	RMSE: 5.147504927449472
		`print("MAE: ",metrics.mean_absolute_error(y_pred,y_test))`
		`print("MSE: ",metrics.mean_squared_error(y_pred,y_test))`
		`print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_pred,y_test)))`
		MAE: 3.6378726166056214
		MSE: 26.334695538705002
		RMSE: 5.1317341648515855

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Gradient Boosting Regression.	In our analysis, we chose Gradient Boosting Regression due to its high accuracy and robust performance across various datasets. After hyperparameter tuning with RandomizedSearchcv, we achieved a remarkable accuracy of 94.3%, making it the best-performing model among all others we tested.