

Model Optimization and Tuning Phase Template

Date	11 July 2024
Team ID	SWTID1720096271
Project Title	Machine learning approach for Predicting the price of natural gas
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
Decision Tree Regressor	max_depth, min_samples_split, min_samples_leaf	<pre># Define the model model = DecisionTreeRegressor() # Define the hyperparameters grid param_grid = { 'max_depth': [1, 3, 5, 7, 10], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 5] } # Perform grid search grid_search = GridSearchCV(model, param_grid, cv=5, scoring='neg_mean_squared_error') grid_search.fit(x, y) # Print best parameters print('Best parameters: ', grid_search.best_params_) # Best Hyperparameters print('Best parameters: ', grid_search.best_params_) Best parameters: {'max_depth': 5, 'min_samples_leaf': 5, 'min_samples_split': 2}</pre>
SVR	c, epsilon, kernel	<pre>from sklearn.svm import SVR from sklearn.model_selection import GridSearchCV # Define the model svr_model = SVR() # Define the hyperparameters grid param_grid_svr = { 'C': [0.1, 1, 10, 100], 'epsilon': [0.01, 0.1, 0.2], 'kernel': ['linear', 'poly', 'rbf', 'sigmoid'] } # Perform grid search grid_search_svr = GridSearchCV(svr_model, param_grid_svr, cv=5, scoring='neg_mean_squared_error') grid_search_svr.fit(x, y) # Print best parameters print('Best parameters: ', grid_search_svr.best_params_) # Best Hyperparameters print('Best parameters for SVR: ', grid_search_svr.best_params_) Best parameters for SVR: {'C': 0.1, 'epsilon': 0.1, 'kernel': 'linear'}</pre>

Random Forest Regressor	n_estimators, max_depth, min_samples_split, min_samples_leaf	<pre># Train and evaluate Random Forest Regressor # Import the model rf_model = RandomForestRegressor() # Define the hyperparameters grid param_grid_rf = { 'n_estimators': [50, 100, 200], 'max_depth': [None, 10, 20, 30], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4] } # Define grid search grid_search_rf = GridSearchCV(rf_model, param_grid_rf, cv=5, scoring='neg_mean_squared_error') # Perform grid search grid_search_rf.fit(x_train, y_train) # Best hyperparameters best_params_rf = grid_search_rf.best_params_ # Test parameters for Random Forest: ['n_estimators': 100, 'max_depth': 30, 'min_samples_split': 5, 'min_samples_leaf': 4]</pre>
-------------------------	--	--

Performance Metrics Comparison Report (2 Marks):

Model	Baseline Metric	Optimized Metric
Decision Tree Regressor	<pre># Train and evaluate Decision Tree Regressor dt_model.fit(x_train, y_train) y_pred_dt = dt_model.predict(x_test) baseline_dt = mean_squared_error(y_test, y_pred_dt, squared=False)</pre> <p>Baseline RMSE for Decision Tree: 0.020540231956418322</p>	<pre># Perform hyperparameter tuning and evaluation # Decision Tree Regressor param_grid_dt = { 'max_depth': [3, 5, 7, 10], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 5] } grid_search_dt = GridSearchCV(DecisionTreeRegressor(), param_grid_dt, cv=5, scoring='neg_mean_squared_error') grid_search_dt.fit(x_train, y_train) y_pred_dt_opt = grid_search_dt.predict(x_test) optimized_dt = mean_squared_error(y_test, y_pred_dt_opt, squared=False)</pre> <p>Optimized RMSE for Decision Tree: 0.017249806930300513</p>
SVR	<pre># Train and evaluate SVR svr_model.fit(x_train, y_train) y_pred_svr = svr_model.predict(x_test) baseline_svr = mean_squared_error(y_test, y_pred_svr, squared=False)</pre> <p>Baseline RMSE for SVR: 0.12873558875775146</p>	<pre># SVR param_grid_svr = { 'C': [0.1, 1, 10, 100], 'epsilon': [0.01, 0.1, 0.2], 'kernel': ['linear', 'poly', 'rbf', 'sigmoid'] } grid_search_svr = GridSearchCV(SVR(), param_grid_svr, cv=5, scoring='neg_mean_squared_error') grid_search_svr.fit(x_train, y_train) y_pred_svr_opt = grid_search_svr.predict(x_test) optimized_svr = mean_squared_error(y_test, y_pred_svr_opt, squared=False)</pre> <p>Optimized RMSE for SVR: 0.10890416589873322</p>
Random Forest Regressor	<pre># Train and evaluate Random Forest Regressor rf_model.fit(x_train, y_train) y_pred_rf = rf_model.predict(x_test) baseline_rf = mean_squared_error(y_test, y_pred_rf, squared=False)</pre> <p>Baseline RMSE for Random Forest: 0.01611641473655759</p>	<pre># Random Forest Regressor param_grid_rf = { 'n_estimators': [50, 100, 200], 'max_depth': [None, 10, 20, 30], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4] } grid_search_rf = GridSearchCV(RandomForestRegressor(), param_grid_rf, cv=5, scoring='neg_mean_squared_error') grid_search_rf.fit(x_train, y_train) y_pred_rf_opt = grid_search_rf.predict(x_test) optimized_rf = mean_squared_error(y_test, y_pred_rf_opt, squared=False)</pre> <p>Optimized RMSE for Random Forest: 0.01622791715705109</p>

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
DesicionTreeRegres sor	I chose the Decision Tree Regressor for predicting natural gas prices because it handles non-linear relationships well, is easy to interpret, and requires minimal data preprocessing. Its ability to model complex patterns in data and provide clear visualizations makes it a suitable choice for this regression task.