

Model Development Phase Template

Date	9 July 2024
Team ID	SWTID1720096271
Project Title	Machine learning approach for Predicting the price of natural gas
Maximum Marks	6 Marks

Model Selection Report

In the forthcoming Model Selection Report, various models will be outlined, detailing their descriptions, hyperparameters, and performance metrics, including Accuracy or F1 Score. This comprehensive report will provide insights into the chosen models and their effectiveness.

Model Selection Report:

Model	Description	Hyperparameters	Performance Metric (e.g., Accuracy, F1 Score)
Linear Regression	Linear Regression predicts a continuous target variable by fitting a linear relationship between input features and the target variable.	null	<pre> from sklearn.linear_model import LinearRegression # Initialize and train the Linear Regression model lr_model = LinearRegression() lr_model.fit(x_train, y_train) # LinearRegression LinearRegression() # Make predictions on the test set y_pred = lr_model.predict(x_test) # Calculate evaluation metrics rmse = np.sqrt(mean_squared_error(y_test, y_pred)) print(metrics.r2_score(y_test, y_pred)) 0.9871686856443018 rmse 0.014620445192057996 </pre>

Decision Tree Regressor	Decision Tree Regressor predicts continuous target values by splitting data into branches based on feature thresholds, optimizing prediction accuracy.	max_depth=5, min_samples_split=10 random_state=2	<pre># Split data into training and testing sets x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=2) print(x.shape, x_train.shape, x_test.shape) (5946, 6) (4756, 6) (1190, 6) # Train a Decision Tree Regressor model_dt = DecisionTreeRegressor(max_depth=5, min_samples_split=10, random_state=2) model_dt.fit(x_train, y_train) # DecisionTreeRegressor DecisionTreeRegressor(max_depth=5, min_samples_split=10, random_state=2) # Predictions y_pred_dt = model_dt.predict(x_test) # Evaluating the model mse_dt = mean_squared_error(y_test, y_pred_dt) rmse_dt = np.sqrt(mse_dt) rmse_dt 0.017540079972366295 from sklearn import metrics print(metrics.r2_score(y_test, y_pred_dt)) 0.9815322881744076</pre>
Random Forest Regressor	Random Forest Regressor predicts continuous values by averaging predictions from multiple decision trees, reducing overfitting and improving accuracy.	n_estimators =100, random_state = 2	<pre># Train a Random Forest Regressor model_rf = RandomForestRegressor(n_estimators=100, random_state=2) model_rf.fit(x_train, y_train) # RandomForestRegressor RandomForestRegressor(random_state=2) y_pred_rf = model_rf.predict(x_test) # Evaluate the model mse_rf = mean_squared_error(y_test, y_pred_rf) rmse_rf = np.sqrt(mse_rf) rmse_rf 0.016233192522670137 print(metrics.r2_score(y_test, y_pred_rf)) 0.9841817720586651</pre>
SVM	Support Vector Regression (SVR) uses support vectors and kernel functions to predict continuous target values with high accuracy and robustness.	kernel='rbf', C=100, gamma=0.1	<pre># Train a Support Vector Regressor model_svm = SVR(kernel='rbf', C=100, gamma=0.1) model_svm.fit(x_train, y_train) # SVR SVR(C=100, gamma=0.1) # Make predictions y_pred_svm = model_svm.predict(x_test) # Evaluate the model mse_svm = mean_squared_error(y_test, y_pred_svm) rmse_svm = np.sqrt(mse_svm) rmse_svm 0.06461344492463574 print(metrics.r2_score(y_test, y_pred_svm)) 0.7493915307908854</pre>