Model Optimization and Tuning Phase

Date	4 June,2024
Team ID	738220
Project Title	Walmart Sales Analysis for Retail Industry with 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:

Model	Tuned Hyperparameters	Optimal Values

Random Forest	from sklearn.ensemble import RandomForestRegressor
	# Create a Random Forest Regressor model
	rf_model = RandomForestRegressor(n_estimators=150, max_depth=30, min_samples_split=5, min_samples_leaf=1, random_state=42)
	
	# Train the model on the training set
	rf_model.fit(X_train, y_train)
	
	# Make predictions on the test set
	rf_predictions = rf_model.predict(X_test)
	
	# Calculate the R^2 score of the model
	rf_score = rf_model.score(X_test, y_test) * 100
	
	# Calculate the Mean Absolute Error (MAE)
	rf_mae = mean_absolute_error(y_test, rf_predictions)
	
	# Calculate the Root Mean Squared Error (RMSE)
	rf_rmse = np.sqrt(mean_squared_error(y_test, rf_predictions))
	
	# Calculate the training accuracy for the Random Forest model
	rf_train_accuracy = rf_model.score(X_train, y_train) * 100
		# Print the MAE and RMSE values
		print(f"Random Forest MAE: {rf_mae:.2f}")
		print(f"Random Forest RMSE: {rf_rmse:.2f}")
		
		Random Forest MAE: 1626.49
		Random Forest RMSE: 4402.19
		
		# Print the R^2 score
		print(f"Random Forest R^2 Score: {rf_score:.2f}%")
		
		Random Forest R^2 Score: 96.35%
		
		print(f"Random Forest Training Accuracy: {r
		
		Random Forest Training Accuracy: 99.05%

Decision Tree	from sklearn.tree import DecisionTreeRegressor
	# Create a decision tree regressor model
	dt_model = DecisionTreeRegressor(random_state=42)
	
	# Train the model on the training set
	dt_model.fit(X_train, y_train)
	
	# Make predictions on the test set
	dt_predictions = dt_model.predict(X_test)
	
	# Calculate the R^2 score of the model
	dt_score = dt_model.score(X_test, y_test) * 100
	
	# Print the R^2 score
	print(f"Decision Tree R^2 Score: {dt_score:.2f}%")
	
	# Calculate the Mean Absolute Error (MAE)
	dt_mae = mean_absolute_error(y_test, dt_predictions)
	
	# Calculate the Root Mean Squared Error (RMSE)
	dt_rmse = np.sqrt(mean_squared_error(y_test, dt_predictions))
	
	# Calculate the training accuracy for the Decision Tree model
	dt_train_accuracy = dt_model.score(X_train, y_train) * 100
		# Print the MAE and RMSE values
		print(f"Decision Tree MAE: {dt_mae:.2f}")
		print(f"Decision Tree RMSE: {dt_rmse:.2f}")
		
		Decision Tree R^2 Score: 94.19%
		Decision Tree MAE: 2075.26
		Decision Tree RMSE: 5558.13
		
		print(f"Decision Tree Training Accuracy: {dt_train_accu
		
		Decision Tree Training Accuracy: 100.00%

<table>
<tr>
<td>ARIMA</td>
<td><pre>
# Calculate the Root Mean Squared Error (RMSE)
arima_rmse = np.sqrt(mean_squared_error(test_data, forecast))

# Calculate the Mean Squared Error (MSE)
arima_mse = mean_squared_error(test_data, forecast)

# Calculate the Mean Absolute Error (MAD)
arima_mad = mean_absolute_error(test_data, forecast)
</pre></td>
<td><pre>
# Print the RMSE, MSE, and MAD
print(f"RMSE: {rmse:.2f}")
print(f"MSE: {mse:.2f}")
print(f"MAD: {mad:.2f}")

RMSE: 686.64
MSE: 471475.70
MAD: 448.51
</pre></td>
</tr>
<tr>
<td>XGBoost</td>
<td><pre>
import xgboost as xgb
from sklearn.metrics import mean_squared_error, mean_absolute_error

# Create an XGBoost regressor model
xgb_model = xgb.XGBRegressor(objective='reg:squarederror', nthread=4, n_estimators=1000, max_depth=5, learning_rate=0.5)

# Train the model on the training set
xgb_model.fit(X_train, y_train)

# Make predictions on the test set
xgb_predictions = xgb_model.predict(X_test)

# Calculate the R^2 score of the model
xgb_score = xgb_model.score(X_test, y_test) * 100

# Print the R^2 score
print(f"XGBoost R^2 Score: {xgb_score:.2f}%")

# Calculate the Mean Absolute Error (MAE)
xgb_mae = mean_absolute_error(y_test, xgb_predictions)

# Calculate the Root Mean Squared Error (RMSE)
xgb_rmse = np.sqrt(mean_squared_error(y_test, xgb_predictions))

# Print the MAE and RMSE values
print(f"XGBoost MAE: {xgb_mae:.2f}")
print(f"XGBoost RMSE: {xgb_rmse:.2f}")

# Calculate the training accuracy for the XGBoost model
xgb_train_accuracy = xgb_model.score(X_train, y_train) * 100

# Print the training accuracy
print(f"XGBoost Training Accuracy: {xgb_train_accuracy:.2f}%")
</pre></td>
<td><pre>
XGBoost R^2 Score: 96.11%
XGBoost MAE: 2094.81
XGBoost RMSE: 4546.16
XGBoost Training Accuracy: 97.50%
</pre></td>
</tr>
</table>

Performance Metrics Comparison Report :

<table>
<tr><th>Model</th><th>Optimized Metric</th></tr>
<tr><td>Decision Tree</td><td><pre>
+---------------+--------------------+--------------------+--------------------+--------------------+
| Model | Training Accuracy | Test Accuracy | RMSE | MAE /MAD (Arima) |
+---------------+--------------------+--------------------+--------------------+--------------------+
| Decision Tree | 100.0 | 94.18924713750879 | 5558.131862816404 | 2075.26
</pre></td></tr>
<tr><td>Random Forest</td><td><pre>
+---------------+--------------------+--------------------+--------------------+--------------------+
| Model | Training Accuracy | Test Accuracy | RMSE | MAE /MAD (Arima) |
+---------------+--------------------+--------------------+--------------------+--------------------+
| Random Forest | 99.05071258942313 | 96.35487250338254 | 4402.192252783296 | 1626.4858674570846 |
</pre></td></tr>
<tr><td>ARIMA</td><td><pre>
+---------------+--------------------+--------------------+--------------------+--------------------+
| Model | Training Accuracy | Test Accuracy | RMSE | MAE /MAD (Arima) |
+---------------+--------------------+--------------------+--------------------+--------------------+
| Arima | - | - | 686.6408829232673 | 448
</pre></td></tr>
<tr><td>XGBoost</td><td><pre>
+---------------+--------------------+--------------------+--------------------+--------------------+
| Model | Training Accuracy | Test Accuracy | RMSE | MAE /MAD (Arima) |
+---------------+--------------------+--------------------+--------------------+--------------------+
| XGBooost | 97.50340544072975 | 96.112549042831 | 4546.164067935629 | 2094.8089620184737 |
</pre></td></tr>
</table>

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Random Forest	Both decision tree and random forest models achieved high accuracy, with decision tree reaching 100% accuracy and random forest achieving 99.05% accuracy. However, decision tree models often result in overfitting and high loss due to their complexity, whereas random forest models balance accuracy with reduced loss across different models. This suggests that random forest is the most effective model, as it maintains high accuracy while retaining the most useful information.