Model Optimization and Tuning Phase Template

Date	15 March 2024
Team ID	SWTID1720116037
Project Title	Ecommerce Shipping Prediction 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

Random Forest	# Define the hyperparameter grid
	params = {
	    'n_estimators': [200, 250, 300], # Increase the number of estimators
	    'criterion': ['gini', 'entropy'],
	    'max_depth': [None, 5, 10], # Add max_depth parameter
	    'min_samples_split': [2, 5, 10], # Add min_samples_split parameter
	    'min_samples_leaf': [1, 5, 10] # Add min_samples_leaf parameter
	}
	
	# Perform hyperparameter tuning
	rf_model = GridSearchCV(estimator=RandomForestClassifier(), param_grid=params, scoring='accuracy', cv=5)
	rf_model.fit(X_train, y_train)
	
	# Get the best-performing model and its hyperparameters
	best_model = rf_model.best_estimator_
	best_params = rf_model.best_params_
	
	# Make predictions on the test data
	y_pred = best_model.predict(X_test)
		params = {
		    'n_estimators': [100, 150, 200],
		    'criterion': ['gini'],
		    'max_depth': [None, 5, 10],
		    'min_samples_split': [2, 5, 10],
		    'min_samples_leaf': [1, 5, 10]
		}

ANN	ann = Sequential() ann.add(Dense(64, input_dim=8, activation='relu')) ann.add(Dense(128, activation='relu')) ann.add(Dense(128, activation='relu')) ann.add(Dense(1, activation='sigmoid')) ann.compile(loss="binary_crossentropy", optimizer='SGD',metrics=['accuracy']) ann.fit(X_train, y_train, epochs=200, batch_size=15)	ann = Sequential() ann.add(Dense(8, input_dim=8, activation='relu')) ann.add(Dense(16, activation='relu')) ann.add(Dense(16, activation='relu')) ann.add(Dense(1, activation='sigmoid')) ann.compile(loss="binary_crossentropy", optimizer='SGD',metrics=['accuracy']) ann.fit(X_train, y_train, epochs=100, batch_size=10)

Support Vector Machine	```param_grid = { 'C': [0.1, 1, 5, 10, 50, 100], 'gamma': ['scale', 'auto', 0.1, 1, 10], 'kernel': ['rbf', 'poly', 'linear'] }```	```svm_model = svm.SVC(gamma='auto',C=5,kernel='rbf') svm_model.fit(X_train,y_train) y_pred = svm_model.predict(X_test)```

Performance Metrics Comparison Report (2 Marks):

<table>
<tr><th>Model</th><th>Optimized Metric</th></tr>
<tr><td>Random Forest</td><td><pre>
print(classification_report(y_test, y_pred))

Classification Report:
              precision   recall f1-score   support

           0   0.57   0.96   0.72   895
           1   0.94   0.51   0.66   1305

    accuracy   0.69   2200
   macro avg   0.76   0.73   0.69   2200
weighted avg   0.79   0.69   0.69   2200

print(confusion_matrix(y_test,y_pred))
</pre></td></tr>
<tr><td>ANN</td><td><pre>
print(classification_report(y_test,predictions))
69/69 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
              precision   recall f1-score   support

           0   0.57   0.84   0.68   895
           1   0.84   0.57   0.68   1305

    accuracy   0.68   2200
   macro avg   0.71   0.71   0.68   2200
weighted avg   0.73   0.68   0.68   2200
</pre></td></tr>
</table>

<table>
<tr><td></td><td><pre>print(confusion_matrix(y_test,y_pred))</pre></td></tr>
<tr><td>Support Vector Machine</td><td><pre>print(classification_report(y_test,y_pred))

              precision   recall f1-score   support

           0   0.55   0.85   0.67   895
           1   0.83   0.53   0.65   1305

    accuracy   0.66   2200
   macro avg   0.69   0.69   0.66   2200
weighted avg   0.72   0.66   0.66   2200

print(confusion_matrix(y_test,y_pred))</pre></td></tr>
</table>

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Random Forest	The random forest model was selected due to its robustness, accuracy, and ability to handle large, complex datasets with both numerical and categorical features. Its ensemble approach, combining multiple decision trees, helps reduce overfitting, can manage missing values and improve generalization with project objectives, justifying its selection as the final model.