Model Optimization and Tuning Phase Report

Date	20 July,2024
Team ID	SWTID1720108903
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):

<table>
<tr><th>Model</th><th>Tuned Hyperparameters</th><th>Optimal Values</th></tr>
<tr><td>Logistic Regression</td><td>lg = LogisticRegressionCV(n_jobs=-1, random_state=1234)

lg_param_grid = {
'Cs': [6, 8, 10, 15, 20],
'max_iter': [60, 80, 100]
}</td><td>lg_cv.fit(x_train_normalized, y_train)

print("Best Score:", lg_cv.best_score_)
print("Best Parameters:", lg_cv.best_params_)

Fitting 5 folds for each of 15 candidates, totalling 75 fits
Best Score: 0.6412089126053026
Best Parameters: {'Cs': 20, 'max_iter': 60}</td></tr>
<tr><td>Random

Forest</td><td>rf = RandomForestClassifier(random_state=1234)

rf_param_grid = {
'n_estimators': [200, 300, 500],
'criterion': ['entropy', 'gini'],
'max_depth': [7, 8, 60, 80, 100],
'max_features': ['auto', 'sqrt', 'log2']
}</td><td>rf_cv = GridSearchCV(rf, param_grid=rf_param_grid, cv=7, scoring="accuracy", n_jobs=-1, verbose=3)

rf_cv.fit(x_train_normalized, y_train)

print("Best Score:", rf_cv.best_score_)
print("Best Parameters:", rf_cv.best_params_)

Best Score: 0.6801909307875896

Best Parameters: {'criterion': 'entropy', 'max_depth': 8, 'max_features': 'sqrt', 'n_estimators': 200}</td></tr>
</table>

<table>
<tr>
<td>KNN</td>
<td>knn = KNeighborsClassifier()

# Define the parameter grid for KNN
knn_param_grid = {
'n_neighbors': [3, 5, 7, 9, 11],
'weights': ['uniform', 'distance'],
'metric': ['euclidean', 'manhattan', 'minkowski']
}</td>
<td># Initialize GridSearchCV
knn_cv = GridSearchCV(knn, knn_param_grid, cv=7, scoring='accuracy', n_jobs=-1, verbose=3)

# Fit the model
knn_cv.fit(x_train_normalized, y_train)

# Output the best score and parameters
print("Best Score: " + str(knn_cv.best_score_))
print("Best Parameters: " + str(knn_cv.best_params_))

Fitting 7 folds for each of 30 candidates, totalling 210 fits
Best Score: 0.6537106489373793
Best Parameters: {'metric': 'euclidean', 'n_neighbors': 9, 'weights': 'distance'}</td>
</tr>
<tr>
<td>Gradient Boosting</td>
<td>xgb = XGBClassifier(learning_rate=0.5, n_estimators=100, objective='binary:logistic', nthread=3)

# Define the parameter grid for XGBoost
params = {
'min_child_weight': [10, 20],
'gamma': [1.5, 2.0, 2.5],
'colsample_bytree': [0.6, 0.8, 0.9],
'max_depth': [4, 5, 6]
}</td>
<td># Initialize GridSearchCV for XGBoost
fitmodel = GridSearchCV(xgb, param_grid=params, cv=5, refit=True, scoring="accuracy", n_jobs=-1, verbose=3)

# Fit the model using the normalized training data
fitmodel.fit(x_train_normalized, y_train)

# Print the best estimator, parameters, and score
print("Best Estimator:", fitmodel.best_estimator_)
print("Best Parameters:", fitmodel.best_params_)
print("Best Score:", fitmodel.best_score_)

Fitting 5 folds for each of 54 candidates, totalling 270 fits
Best Estimator: XGBClassifier(base_score=None, booster=None, callbacks=None,
colsample_bylevel=None, colsample_bynode=None,
colsample_bytree=0.9, device=None, early_stopping_rounds=None,
enable_categorical=False, eval_metric=None, feature_types=None,
gamma=2.0, grow_policy=None, importance_type=None,
interaction_constraints=None, learning_rate=0.5, max_bin=None,
max_cat_threshold=None, max_cat_to_onehot=None,
max_delta_step=None, max_depth=4, max_leaves=None,
min_child_weight=20, missing=nan, monotone_constraints=None,
multi_strategy=None, n_estimators=100, n_jobs=None, nthread=3,
num_parallel_tree=None, ...)
Best Parameters: {'colsample_bytree': 0.9, 'gamma': 2.0, 'max_depth': 4, 'min_child_weight': 20}
Best Score: 0.6763268127551811</td>
</tr>
</table>

Performance Metrics Comparison Report (2 Marks):

<table>
<tr>
<th>Model</th>
<th>Optimized Metric</th>
</tr>
<tr>
<td>KNN</td>
<td># Initialize GridSearchCV
knn_cv = GridSearchCV(knn, knn_param_grid, cv=7, scoring='accuracy', n_jobs=-1, verbose=3)

# Fit the model
knn_cv.fit(x_train_normalized, y_train)

# Output the best score and parameters
print("Best Score: " + str(knn_cv.best_score_))
print("Best Parameters: " + str(knn_cv.best_params_))

Fitting 7 folds for each of 30 candidates, totalling 210 fits
Best Score: 0.6537106489373793
Best Parameters: {'metric': 'euclidean', 'n_neighbors': 9, 'weights': 'distance'}</td>
</tr>
</table>

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
KNN	The KNN model was selected for its balanced performance across various metrics. Its ability to classify based on the nearest neighbors makes it adaptable to data patterns and effective for capturing local variations in loan approval criteria. The high F1 score and recall values indicate its robustness in correctly identifying loan approvals, which aligns with project objectives, justifying its selection as the final model.