SMARTBRIDGE Let's Bridge the Gap a Veranda Enterprise | Smart Internz

Model Optimization and Tuning Phase Template

Date	20 July 2024
Team ID	SWTID1720110595
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>XGBoost</td><td><pre><code># Hyperparameter tuning for XGBoost
param_dist_xgb = {
    'n_estimators': [100, 200, 300],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 5, 7],
    'subsample': [0.8, 0.9, 1.0]
}

random_search_xgb =
RandomizedSearchCV(estimator=XGBClassifier(random_state=42),
param_distributions=param_dist_xgb,
n_iter=50, cv=3, verbose=2,
random_state=42, n_jobs=-1)
random_search_xgb.fit(X_train_res,
y_train_res)
best_xgb =
random_search_xgb.best_estimator_</code></pre></td><td>n_estimators: 200,
learning_rate: 0.1,
max_depth: 5, subsample: 0.9</td></tr>
</table>

<table>
<tr><td></td><td></td><td></td></tr>
<tr><td>Random Forest</td><td><pre><code># Hyperparameter tuning for Random Forest
param_dist_rf = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

random_search_rf = RandomizedSearchCV(estimator=RandomForestClassifier(random_state=42), param_distributions=param_dist_rf, n_iter=50, cv=3, verbose=2, random_state=42, n_jobs=-1)
random_search_rf.fit(X_train_res, y_train_res)
best_rf = random_search_rf.best_estimator_</code></pre></td><td>n_estimators: 200,

max_depth: 10,

min_samples_split: 2,

min_samples_leaf: 1</td></tr>
<tr><td>Logistic

Regression CV</td><td><pre><code># Hyperparameter tuning for Logistic Regression CV
best_lcv = LogisticRegressionCV(cv=5, random_state=42, solver='lbfgs', max_iter=1000).fit(X_train_res, y_train_res)

# Hyperparameter tuning for Ridge Classifier
param_dist_rg = {
    'alpha': [0.1, 1.0, 10.0],
    'solver': ['auto', 'svd', 'cholesky', 'lsqr', 'sag']
}

random_search_rg = RandomizedSearchCV(estimator=RidgeClassifier(random_state=42),</code></pre></td><td>Cs: 10, cv: 5, solver: 'lbfgs'</td></tr>
</table>

<table>
<tr>
<td></td>
<td><pre>param_distributions=param_dist_rg,
n_iter=50, cv=3, verbose=2,
random_state=42, n_jobs=-1)
random_search_rg.fit(X_train_res,
y_train_res)
best_rg =
random_search_rg.best_estimator_</pre></td>
<td></td>
</tr>
<tr>
<td>Ridge Classifier</td>
<td><pre># Hyperparameter tuning for Ridge
Classifier
param_dist_rg = {
    'alpha': [0.1, 1.0, 10.0],
    'solver': ['auto', 'svd',
'cholesky', 'lsqr', 'sag']
}

random_search_rg =
RandomizedSearchCV(estimator=RidgeClas
sifier(random_state=42),
param_distributions=param_dist_rg,
n_iter=50, cv=3, verbose=2,
random_state=42, n_jobs=-1)
random_search_rg.fit(X_train_res,
y_train_res)
best_rg =
random_search_rg.best_estimator_</pre></td>
<td>alpha: 1.0, solver: 'auto'</td>
</tr>
<tr>
<td>KNN</td>
<td><pre># Hyperparameter tuning for KNN
param_dist_knn = {
    'n_neighbors': [3, 5, 7],
    'weights': ['uniform',
'distance'],
    'algorithm': ['auto', 'ball_tree',
'kd_tree', 'brute']
}

random_search_knn =
RandomizedSearchCV(estimator=KNeighbor
sClassifier(),
param_distributions=param_dist_knn,</pre></td>
<td>n_neighbors: 5, weights:

'uniform', algorithm: 'auto'</td>
</tr>
</table>

<table>
<tr>
<td></td>
<td>

```
n_iter=50, cv=3, verbose=2,
random_state=42, n_jobs=-1)
random_search_knn.fit(X_train_res,
y_train_res)
best_knn =
random_search_knn.best_estimator_
```

</td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>

```
# Hyperparameter tuning for SVM
param_dist_svm = {
    'C': [0.1, 1.0, 10.0],
    'kernel': ['linear', 'poly',
'rbf', 'sigmoid'],
    'gamma': ['scale', 'auto']
}

random_search_svm =
RandomizedSearchCV(estimator=svm.SVC(p
robability=True, random_state=42),
param_distributions=param_dist_svm,
n_iter=50, cv=3, verbose=2,
random_state=42, n_jobs=-1)
random_search_svm.fit(X_train_res,
y_train_res)
best_svm =
random_search_svm.best_estimator_
```

</td>
<td>C: 1.0, kernel: 'rbf', gamma: 'scale'</td>
</tr>
</table>

Performance Metrics Comparison Report (2 Marks):

Model	Baseline Metric	Optimized Metric
Logistic Regression CV	70.37	72.10
XGBoost	72.54	76.32

Random Forest	73.99	75.80
Ridge Classifier	71.21	72.30
KNN	71.33	72.50
SVM	73.53	74.20

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Stacking Classifier (RandomForestClassifier, XGBClassifier, LogisticRegression)	The Stacking Classifier was chosen as the final optimized model for several reasons: 1. **Improved Performance through Ensemble Learning**: o The Stacking Classifier combines the strengths of multiple models, particularly the RandomForestClassifier and the XGBClassifier, to create a more robust and accurate prediction model. By leveraging the advantages of different algorithms, the stacking ensemble can capture various patterns in the data that individual models might miss. 2. **Balanced Approach to Bias-Variance Trade-off**: o The RandomForestClassifier is known for its ability to handle high variance, while the XGBClassifier excels in reducing bias through boosting techniques. Combining these models in a stacking ensemble helps balance the bias-variance trade-off, leading to better generalization on unseen data. 3. **Hyperparameter Optimization**:

	○ The XGBClassifier within the stacking ensemble has been fine-tuned using RandomizedSearchCV to find the optimal hyperparameters, ensuring that the model operates at its best performance. This optimization step enhances the overall effectiveness of the stacking classifier. 4. **Superior Evaluation Metrics**: ○ The Stacking Classifier demonstrated superior performance metrics compared to other individual models during the evaluation phase. Specifically, it achieved higher accuracy, F1-score, recall, and precision, indicating its effectiveness in both predicting the correct class and minimizing false positives and false negatives. 5. **Cross-Validation Results**: ○ The Stacking Classifier achieved the highest mean cross-validation score among all tested models, suggesting that it generalizes well to different subsets of the data and is less likely to overfit. 6. **Flexibility and Robustness**: ○ By using LogisticRegression as the final estimator, the Stacking Classifier benefits from a probabilistic approach to combining the base models' predictions, adding an additional layer of flexibility and robustness to the ensemble.