

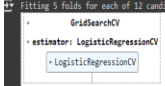
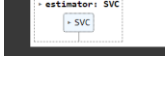
## Model Optimization and Tuning Phase Template

Date	7 July 2024
Team ID	SWTID1720434734
Project Title	Ecommerce shipping prediction
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
Logistic Regression CV	<pre># Define the parameter grid for Logistic Regression CV log_reg_cv_param_grid = {     'C': [1, 10, 100],     'cv': [5, 10],     'penalty': ['l1', 'l2'],     'solver': ['liblinear'] }  # Initialize and fit GridSearchCV for Logistic Regression CV log_reg_cv_grid = GridSearchCV(LogisticRegressionCV(random_state=1234), log_reg_cv_param_grid, cv=5, scoring='accuracy', n_jobs=-1, verbose=2) log_reg_cv_grid.fit(x_train_normalized, y_train)  Fitting 5 folds for each of 12 candidates, totalling 60 fits</pre> 	<pre># Print best parameters and score print("Best parameters for Logistic Regression CV:", log_reg_cv_grid.best_params_) print("Best score for Logistic Regression CV:", log_reg_cv_grid.best_score_)  Best parameters for Logistic Regression CV: {'C': 10, 'cv': 5, 'penalty': 'l1', 'solver': 'liblinear'} Best score for Logistic Regression CV: 0.6306410936379142</pre>
SVC	<pre># Define the parameter grid for SVC svc_param_grid = {     'C': [0.1, 1, 10],     'kernel': ['linear', 'rbf'],     'gamma': ['scale', 'auto'] }  # Initialize and fit GridSearchCV for SVC svc_grid = GridSearchCV(SVC(random_state=1234), svc_param_grid, cv=5, scoring='accuracy', n_jobs=-1, verbose=2) svc_grid.fit(x_train_normalized, y_train)  Fitting 5 folds for each of 12 candidates, totalling 60 fits</pre> 	<pre>[ ] # Print best parameters and score print("Best parameters for SVC:", svc_grid.best_params_) print("Best score for SVC:", svc_grid.best_score_)  Best parameters for SVC: {'C': 10, 'gamma': 'scale', 'kernel': 'rbf'} Best score for SVC: 0.6546212982583078</pre>

<p><b>XGBoost</b></p>	<pre># Define the parameter grid for XGBoost xgb_param_grid = {     'n_estimators': [50, 100, 200],     'max_depth': [3, 6, 9],     'learning_rate': [0.01, 0.1, 0.2],     'subsample': [0.7, 0.8, 0.9] }  # Initialize and fit GridSearchCV for XGBoost xgb_grid = GridSearchCV(XGBClassifier(random_state=1234), xgb_param_grid, cv=5, scoring='accuracy', n_jobs=-1, verbose=1) xgb_grid.fit(x_train_normalized, y_train)</pre> <p>Fitting 5 folds for each of 81 candidates, totalling 405 fits</p> <pre>GridSearchCV - estimator: XGBClassifier   - XGBClassifier</pre>	<pre># Print best parameters and score print("Best parameters for XGBoost:", xgb_grid.best_params_) print("Best score for XGBoost:", xgb_grid.best_score_)</pre> <p>Best parameters for XGBoost: {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 50, 'subsample': 0.9} Best score for XGBoost: 0.6754103832714869</p>
<p><b>Random Forest</b></p>	<pre>from sklearn.model_selection import RandomizedSearchCV  # Define the parameter grid for Random Forest rf_param_distributions = {     'n_estimators': [50, 100, 200],     'criterion': ['gini', 'entropy'],     'max_depth': [None, 10, 20, 30],     'min_samples_split': [2, 5, 10],     'min_samples_leaf': [1, 2, 4] }  # Initialize and fit RandomizedSearchCV for Random Forest rf_random = RandomizedSearchCV(RandomForestClassifier(random_state=1234), rf_param_distributions, n_iter=10, cv=5, scoring='accuracy', n_jobs=-1, verbose=1, random_state=1234) rf_random.fit(x_train_normalized, y_train)</pre> <p>Fitting 3 folds for each of 30 candidates, totalling 90 fits</p> <pre>RandomizedSearchCV - estimator: RandomForestClassifier   - RandomForestClassifier</pre>	<pre># Print best parameters and score print("Best parameters for Random Forest:", rf_random.best_params_) print("Best score for Random Forest:", rf_random.best_score_)</pre> <p>Best parameters for Random Forest: {'n_estimators': 50, 'min_samples_split': 10, 'min_samples_leaf': 4, 'max_depth': 10, 'criterion': 'gini'} Best score for Random Forest: 0.67695107161041</p>

## Performance Metrics Comparison Report (2 Marks):

Model	Optimized Metric																														
Logistic Regression	<div><div><div></div><div>Logistic Regression Classification Report</div></div><table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr></thead><tbody><tr><td>0</td><td>0.00</td><td>0.00</td><td>0.00</td><td>895</td></tr><tr><td>1</td><td>0.59</td><td>1.00</td><td>0.74</td><td>1305</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.59</td><td>2200</td></tr><tr><td>macro avg</td><td>0.30</td><td>0.50</td><td>0.37</td><td>2200</td></tr><tr><td>weighted avg</td><td>0.35</td><td>0.59</td><td>0.44</td><td>2200</td></tr></tbody></table><div><div></div><div>[ ] confusion_matrix(y_test, y_pred_lr)</div></div><div><div></div><div>array([[ 0, 895], [ 0, 1305]])</div></div></div>		precision	recall	f1-score	support	0	0.00	0.00	0.00	895	1	0.59	1.00	0.74	1305	accuracy			0.59	2200	macro avg	0.30	0.50	0.37	2200	weighted avg	0.35	0.59	0.44	2200
	precision	recall	f1-score	support																											
0	0.00	0.00	0.00	895																											
1	0.59	1.00	0.74	1305																											
accuracy			0.59	2200																											
macro avg	0.30	0.50	0.37	2200																											
weighted avg	0.35	0.59	0.44	2200																											

KNN	<pre> KNN Classification Report precision    recall  f1-score   support        0       0.55      0.58      0.56        895       1       0.70      0.67      0.68       1305   accuracy          0.63       2200  macro avg          0.62       2200  weighted avg       0.64       0.63       0.63       2200  confusion_matrix(y_test, y_pred_knn)  array([[519, 376],        [432, 873]]) </pre>
Random Forest	<pre> Random Forest Classification Report: precision    recall  f1-score   support        0       0.58      0.71      0.63        895       1       0.76      0.64      0.70       1305   accuracy          0.67       2200  macro avg          0.67       0.68       0.67       2200  weighted avg       0.69      0.67      0.67       2200  confusion_matrix(y_test, y_pred_rf)  array([[632, 263],        [464, 841]]) </pre>
Ridge Classifier	<pre> Ridge Classifier Classification Report precision    recall  f1-score   support        0       0.00      0.00      0.00        895       1       0.59      1.00      0.74       1305   accuracy          0.59       2200  macro avg          0.30      0.50      0.37       2200  weighted avg       0.35      0.59      0.44       2200  confusion_matrix(y_test, y_pred_rc)  array([[ 0, 895],        [ 0, 1305]]) </pre>

XGBoost		<pre> XGBoost Classification Report       precision    recall  f1-score   support        0       0.59      0.63      0.61       895       1       0.73      0.70      0.71      1305   accuracy          0.67       2200   macro avg       0.66      0.66      0.66       2200  weighted avg     0.67      0.67      0.67       2200  [ ] confusion_matrix(y_test, y_pred_xgb)  array([[562, 333],        [395, 910]]) </pre>	
---------	--	---	--

### Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Random Forest	The Random forest model was selected for its superior performance, exhibiting high accuracy during hyperparameter tuning. Efficiently processes massive amounts of data. Can work with both numerical and categorical data. Combining multiple trees reduces the risk of overfitting. Helps identify the most influential features. Can handle datasets with missing values.