Model Optimization and Tuning Phase Template

Date	04 June 2024
Team ID	SWTID1720096620
Project Title	E-commerce 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 Classifier	`smote = SMOTE(random_state=42)` `classifier = RandomForestClassifier(n_estimators=200, criterion='entropy', random_state=56,max_depth=5)` `pipeline = Pipeline([` `('smote', smote),` `('classifier', classifier)` `])` `param_grid = {` `'classifier__n_estimators': [100, 200, 300],` `'classifier__max_depth': [None, 10, 20, 30],` `'classifier__min_samples_split': [2, 5, 10],` `'classifier__min_samples_leaf': [1, 2, 4]` `}`	Best Parameters: {'classifier__max_depth': 10, 'classifier__min_samples_leaf': 1, 'classifier__min_samples_split': 2, 'classifier__n_estimators': 200} Best Cross-Validation Score: 0.6839847936339164 Accuracy with Hyperparameter Tuning and SMOTE: 0.6893939393939394
K-Nearest Neighbors Classifier	`smote = SMOTE(random_state=42)` `classifier = KNeighborsClassifier(n_neighbors=12, weights='uniform', metric='euclidean', p=5)` `pipeline = Pipeline([` `('smote', smote),` `('classifier', classifier)` `])` `param_grid = {` `'classifier__n_neighbors': [3, 5, 7, 9],` `'classifier__weights': ['uniform', 'distance'],` `'classifier__metric': ['euclidean', 'manhattan', 'minkowski'],` `'classifier__p': [1, 2]` `}`	Best Parameters: {'classifier__metric': 'euclidean', 'classifier__n_neighbors': 9, 'classifier__p': 1, 'classifier__weights': 'uniform'} Best Cross-Validation Score: 0.6530739869775356 Accuracy with Hyperparameter Tuning and SMOTE: 0.6633333333333333

Logistic Regression	``` smote = SMOTE(random_state=42) classifier = LogisticRegression(random_state=42) pipeline = Pipeline([('smote', smote), ('classifier', classifier)]) param_grid = { 'classifier__solver': ['liblinear', 'saga'], 'classifier__penalty': ['l1', 'l2'], 'classifier__C': [0.01, 0.1, 1, 10, 100], 'classifier__max_iter': [100, 200, 300] } ```	Best Parameters: {'classifier__C': 0.01, 'classifier__max_iter': 100, 'classifier__penalty': 'l2', 'classifier__solver': 'liblinear'} Best Cross-Validation Score: 0.6511227264283181 Accuracy with Hyperparameter Tuning and SMOTE: 0.64363636
XGB Classifier	``` smote = SMOTE(random_state=42) classifier = XGBClassifier(eval_metric='mlogloss', random_state=42) pipeline = Pipeline([('smote', smote), ('classifier', classifier)]) param_grid = { 'classifier__n_estimators': [100, 200, 300], 'classifier__max_depth': [3, 5, 7], 'classifier__learning_rate': [0.01, 0.1, 0.2], 'classifier__subsample': [0.7, 0.8, 0.9] } ```	Best Parameters: {'classifier__learning_rate': 0.01, 'classifier__max_depth': 5, 'classifier__n_estimators': 200, 'classifier__subsample': 0.7} Best Cross-Validation Score: 0.6822955536990625 Accuracy with Hyperparameter Tuning and SMOTE: 0.690606
Support Vector Classifier	``` smote = SMOTE(random_state=42) classifier = SVC(random_state=42) pipeline = Pipeline([('smote', smote), ('classifier', classifier)]) param_grid = { 'classifier__kernel': ['linear', 'poly', 'rbf', 'sigmoid'], 'classifier__C': [0.1, 1, 10, 100], 'classifier__gamma': ['scale', 'auto'] } ```	Best Parameters: {'classifier__C': 10, 'classifier__gamma': 'auto', 'classifier__kernel': 'poly'} Best Cross-Validation Score: 0.6680090799389043 Accuracy with Hyperparameter Tuning and SMOTE: 0.670303
Decision Tree Classifier	``` smote = SMOTE(random_state=42) classifier = DecisionTreeClassifier(random_state=42) pipeline = Pipeline([('smote', smote), ('classifier', classifier)]) param_grid = { 'classifier__criterion': ['gini', 'entropy'], 'classifier__max_depth': [None, 10, 20, 30, 40, 50], 'classifier__min_samples_split': [2, 5, 10], 'classifier__min_samples_leaf': [1, 2, 4] } ```	Best Parameters: {'classifier__criterion': 'gini', 'classifier__max_depth': 10, 'classifier__min_samples_leaf': 4, 'classifier__min_samples_split': 2} Best Cross-Validation Score: 0.6646317814738868 Accuracy with Hyperparameter Tuning and SMOTE: 0.677576

Naive Bayes Classifier	`smote = SMOTE(random_state=42)` `classifier = GaussianNB()` `pipeline = Pipeline([` `('smote', smote),` `('classifier', classifier)` `])` `param_grid = {` `'classifier__var_smoothing': [1e-9, 1e-8, 1e-7]` `}`	`Best Parameters: {'classifier__var_smoothing': 1e-09}` `Best Cross-Validation Score: 0.6559285967608465` `Accuracy with Hyperparameter Tuning and SMOTE: 0.650909`
Ada Boost Classifier	`smote = SMOTE(random_state=42)` `classifier = AdaBoostClassifier(random_state=42)` `pipeline = Pipeline([` `('smote', smote),` `('classifier', classifier)` `])` `param_grid = {` `'classifier__n_estimators': [50, 100, 200],` `'classifier__learning_rate': [0.01, 0.1, 1]` `}`	`Best Parameters: {'classifier__learning_rate': 1, 'classifier__n_estimators': 200}` `Best Cross-Validation Score: 0.6747627486223978` `Accuracy with Hyperparameter Tuning and SMOTE: 0.688485`
Gradient Boost Classifier	`smote = SMOTE(random_state=42)` `classifiergb = GradientBoostingClassifier(random_state=42)` `pipeline = Pipeline([` `('smote', smote),` `('classifier', classifiergb)` `])` `param_grid = {` `'classifier__n_estimators': [100, 200, 300],` `'classifier__learning_rate': [0.01, 0.1, 0.2],` `'classifier__max_depth': [3, 5, 7],` `'classifier__subsample': [0.7, 0.8, 0.9]` `}`	`Best Parameters: {'classifier__learning_rate': 0.01, 'classifier__max_depth': 5, 'classifier__n_estimators': 200, 'classifier__subsample': 0.9}` `Best Cross-Validation Score: 0.6825556315029999` `Accuracy with Hyperparameter Tuning and SMOTE: 0.692121`

Performance Metrics Comparison Report (2 Marks):

Model	Baseline Metric	Optimized Metric
Random Forest Classifier	Accuracy without Hyperparameter Tuning and SMOTE: 0.690000 Classification Report without Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.58 0.91 0.71 1379 1 0.89 0.53 0.67 1921 accuracy 0.69 3300 macro avg 0.74 0.72 0.69 3300 weighted avg 0.76 0.69 0.69 3300 Confusion Matrix without Hyperparameter Tuning and SMOTE: [[1250 129] [894 1027]]	Accuracy with Hyperparameter Tuning and SMOTE: 0.6893939393939394 Classification Report with Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.58 0.95 0.72 1379 1 0.93 0.50 0.65 1921 accuracy 0.69 3300 macro avg 0.75 0.73 0.69 3300 weighted avg 0.78 0.69 0.68 3300 Confusion Matrix with Hyperparameter Tuning and SMOTE: [[1306 73] [952 969]]
K-Nearest Neighbors Classifier	Accuracy without Hyperparameter Tuning and SMOTE: 0.666969696969697 Classification Report without Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.58 0.75 0.65 1379 1 0.77 0.61 0.68 1921 accuracy 0.67 3300 macro avg 0.68 0.68 0.67 3300 weighted avg 0.69 0.67 0.67 3300 Confusion Matrix without Hyperparameter Tuning and SMOTE: [[1035 344] [755 1166]]	Accuracy with Hyperparameter Tuning and SMOTE: 0.6633333333333333 Classification Report with Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.57 0.81 0.67 1379 1 0.80 0.56 0.66 1921 accuracy 0.66 3300 macro avg 0.68 0.68 0.66 3300 weighted avg 0.70 0.66 0.66 3300 Confusion Matrix with Hyperparameter Tuning and SMOTE: [[1111 268] [843 1078]]
Logistic Regression	Accuracy without Hyperparameter Tuning and SMOTE: 0.6284848484848485 Classification Report without Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.56 0.53 0.54 1379 1 0.67 0.70 0.69 1921 accuracy 0.63 3300 macro avg 0.62 0.61 0.61 3300 weighted avg 0.63 0.63 0.63 3300 Confusion Matrix without Hyperparameter Tuning and SMOTE: [[725 654] [572 1349]]	Accuracy with Hyperparameter Tuning and SMOTE: 0.64363636 Classification Report with Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.55 0.76 0.64 1379 1 0.77 0.56 0.65 1921 accuracy 0.64 3300 macro avg 0.66 0.66 0.64 3300 weighted avg 0.68 0.64 0.64 3300 Confusion Matrix with Hyperparameter Tuning and SMOTE: [[1053 326] [850 1071]]

XGB Classifier	Accuracy without Hyperparameter Tuning and SMOTE: 0.644545 Classification Report without Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.57 0.62 0.59 1379 1 0.71 0.66 0.68 1921 accuracy 0.64 3300 macro avg 0.64 0.64 0.64 3300 weighted avg 0.65 0.64 0.65 3300 Confusion Matrix without Hyperparameter Tuning and SMOTE: [[853 526] [647 1274]]	Accuracy with Hyperparameter Tuning and SMOTE: 0.690606 Classification Report with Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.58 0.96 0.72 1379 1 0.94 0.50 0.65 1921 accuracy 0.69 3300 macro avg 0.76 0.73 0.69 3300 weighted avg 0.79 0.69 0.68 3300 Confusion Matrix with Hyperparameter Tuning and SMOTE: [[1320 59] [962 959]]
Support Vector Classifier	Accuracy without Hyperparameter Tuning and SMOTE: 0.668788 Classification Report without Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.57 0.82 0.67 1379 1 0.81 0.56 0.66 1921 accuracy 0.67 3300 macro avg 0.69 0.69 0.67 3300 weighted avg 0.71 0.67 0.67 3300 Confusion Matrix without Hyperparameter Tuning and SMOTE: [[1129 250] [843 1078]]	Accuracy with Hyperparameter Tuning and SMOTE: 0.670303 Classification Report with Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.56 0.92 0.70 1379 1 0.90 0.49 0.63 1921 accuracy 0.67 3300 macro avg 0.73 0.71 0.67 3300 weighted avg 0.76 0.67 0.66 3300 Confusion Matrix with Hyperparameter Tuning and SMOTE: [[1274 105] [983 938]]
Decision Tree Classifier	Accuracy without Hyperparameter Tuning and SMOTE: 0.647576 Classification Report without Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.58 0.57 0.58 1379 1 0.70 0.70 0.70 1921 accuracy 0.65 3300 macro avg 0.64 0.64 0.64 3300 weighted avg 0.65 0.65 0.65 3300 Confusion Matrix without Hyperparameter Tuning and SMOTE: [[787 592] [571 1350]]	Accuracy with Hyperparameter Tuning and SMOTE: 0.677576 Classification Report with Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.57 0.89 0.70 1379 1 0.87 0.53 0.66 1921 accuracy 0.68 3300 macro avg 0.72 0.71 0.68 3300 weighted avg 0.74 0.68 0.67 3300 Confusion Matrix with Hyperparameter Tuning and SMOTE: [[1224 155] [909 1012]]

Naive Bayes Classifier	Accuracy without Hyperparameter Tuning and SMOTE: 0.646364 Classification Report without Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.55 0.78 0.65 1379 1 0.78 0.55 0.64 1921 accuracy 0.65 3300 macro avg 0.67 0.67 0.65 3300 weighted avg 0.68 0.65 0.65 3300 Confusion Matrix without Hyperparameter Tuning and SMOTE: [[1079 300] [867 1054]]	Accuracy with Hyperparameter Tuning and SMOTE: 0.650909 Classification Report with Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.55 0.89 0.68 1379 1 0.86 0.48 0.62 1921 accuracy 0.65 3300 macro avg 0.70 0.68 0.65 3300 weighted avg 0.73 0.65 0.64 3300 Confusion Matrix with Hyperparameter Tuning and SMOTE: [[1226 153] [999 922]]
Ada Boost Classifier	Accuracy without Hyperparameter Tuning and SMOTE: 0.679394 Classification Report without Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.59 0.75 0.66 1379 1 0.78 0.63 0.69 1921 accuracy 0.68 3300 macro avg 0.69 0.69 0.68 3300 weighted avg 0.70 0.68 0.68 3300 Confusion Matrix without Hyperparameter Tuning and SMOTE: [[1039 340] [718 1203]]	Accuracy with Hyperparameter Tuning and SMOTE: 0.688485 Classification Report with Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.58 0.89 0.71 1379 1 0.87 0.54 0.67 1921 accuracy 0.69 3300 macro avg 0.73 0.72 0.69 3300 weighted avg 0.75 0.69 0.68 3300 Confusion Matrix with Hyperparameter Tuning and SMOTE: [[1229 150] [878 1043]]
Gradient Boost Classifier	Accuracy without Hyperparameter Tuning and SMOTE: 0.683939 Classification Report without Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.58 0.87 0.70 1379 1 0.85 0.55 0.67 1921 accuracy 0.68 3300 macro avg 0.72 0.71 0.68 3300 weighted avg 0.74 0.68 0.68 3300 Confusion Matrix without Hyperparameter Tuning and SMOTE: [[1194 185] [858 1063]]	Accuracy with Hyperparameter Tuning and SMOTE: 0.692121 Classification Report with Hyperparameter Tuning and SMOTE: precision recall f1-score support 0 0.58 0.95 0.72 1379 1 0.93 0.51 0.66 1921 accuracy 0.69 3300 macro avg 0.76 0.73 0.69 3300 weighted avg 0.79 0.69 0.68 3300 Confusion Matrix with Hyperparameter Tuning and SMOTE: [[1310 69] [947 974]]

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Gradient Boost Classifier	Gradient Boosting was selected as the preferred algorithm for our classification task due to its superior accuracy in capturing complex patterns within the dataset. This algorithm excels in iteratively building an ensemble of weak learners, typically decision trees, where each subsequent tree corrects the errors of the previous ones. This boosting process focuses on improving the performance of areas where previous models underperformed, leading to a highly refined and accurate final model. The ability of Gradient Boosting to effectively minimize errors and improve predictive power with each iteration makes it particularly suitable for datasets with intricate feature interactions and dependencies. In contrast to other ensemble methods like Random Forest, XGBoost, and AdaBoost, Gradient Boosting's approach of sequentially improving weak learners allows it to model non-linear relationships more effectively. Random Forest, while robust due to its use of multiple trees and averaging, does not sequentially correct errors and may not capture complex patterns as efficiently. XGBoost, although similar to Gradient Boosting, uses different optimization techniques and regularization methods, which might not have been as well-suited to our specific data characteristics. AdaBoost, focusing on adjusting the weights of misclassified instances, may not achieve the same level of refinement in capturing intricate data patterns. Therefore, Gradient Boosting's iterative and corrective approach provided a distinct advantage in modelling our dataset, leading to its selection for the highest accuracy performance.