Model Development Phase Template

Date	1 July 2024
Team ID	SWTID1720434734
Project Title	Ecommerce Shipping Prediction
Maximum Marks	4 Marks

Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

Initial Model Training Code:

```
def models_eval(x_train, y_train, x_test, y_test):
    models = {
        'Logistic Regression': LogisticRegression(random_state=1234),
        'Logistic Regression CV': LogisticRegressionCV(random_state=1234),
        'XGBoost': XGBClassifier(random_state=1234),
        'Ridge Classifier': RidgeClassifier(random_state=1234),
        'KNN': KNeighborsClassifier(),
        'Random Forest': RandomForestClassifier(random_state=1234),
        'SVM Classifier': svm.SVC(random_state=1234)
    }

    for name, model in models.items():
        model.fit(x_train, y_train)
        train_score = model.score(x_train, y_train)
        test_score = model.score(x_test, y_test)
        print(f' -- {name}')
        print(f'Train Score: {train_score:.4f}')
        print(f'Test Score: {test_score:.4f}')
        print()

    return models
```

```
# Train and evaluate Logistic Regression model
logistic_regression = LogisticRegression(random_state=1234)
logistic_regression.fit(x_train_normalized, y_train)
y_pred_lr = logistic_regression.predict(x_test_normalized)

# Classification report and confusion matrix for Logistic Regression
print("Logistic Regression Classification Report")
print(classification_report(y_test, y_pred_lr))
```

```
logistic_regression_cv = LogisticRegressionCV(random_state=1234)
logistic_regression_cv.fit(x_train_normalized, y_train)
y_pred_lrcv = logistic_regression_cv.predict(x_test_normalized)

# Classification report and confusion matrix for Logistic Regression CV
print("Logistic Regression CV Classification Report")
print(classification_report(y_test, y_pred_lrcv))
```

```
xgboost = XGBClassifier(random_state=1234)
xgboost.fit(x_train_normalized, y_train)
y_pred_xgb = xgboost.predict(x_test_normalized)

# Classification report and confusion matrix for XGBoost
print("XGBoost Classification Report")
print(classification_report(y_test, y_pred_xgb))
```

```
ridge_classifier = RidgeClassifier(random_state=1234)
ridge_classifier.fit(x_train_normalized, y_train)
y_pred_rc = ridge_classifier.predict(x_test_normalized)

# Classification report and confusion matrix for Ridge Classifier
print("Ridge Classifier Classification Report")
print(classification_report(y_test, y_pred_rc))
```

```
knn = KNeighborsClassifier()
knn.fit(x_train_normalized, y_train)
y_pred_knn = knn.predict(x_test_normalized)

# Classification report and confusion matrix for KNN
print("KNN Classification Report")
print(classification_report(y_test, y_pred_knn))
```

```
rf_model = RandomForestClassifier(random_state=1234)
rf_model.fit(x_train_normalized, y_train)

# Predict and evaluate the model
y_pred_rf = rf_model.predict(x_test_normalized)
print("Random Forest Classification Report:")
print(classification_report(y_test, y_pred_rf))
```

```
svm_model = svm.SVC(random_state=1234)
svm_model.fit(x_train_normalized, y_train)

# Predict and evaluate the model
y_pred_svm = svm_model.predict(x_test_normalized)
print("SVM Classification Report:")
print(classification_report(y_test, y_pred_svm))
```

Model Validation and Evaluation Report:

Model	Classification Report	Accuracy	Confusion Matrix
Logistic Regression	`print("Logistic Regression Classification Report")` Logistic Regression 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	59%	`confusion_matrix(y_test, y_pred_lr)` array([[0, 895], [0, 1305]])
Logistic Regression CV	`print("Logistic Regression CV Classification Report")` Logistic Regression CV Classification Report precision recall f1-score support 0 0.54 0.52 0.53 895 1 0.68 0.69 0.68 1305 accuracy 0.62 2200 macro avg 0.61 0.61 0.61 2200 weighted avg 0.62 0.62 0.62 2200	62%	`confusion_matrix(y_test, y_pred_lrcv)` array([[467, 428], [403, 902]])
XGBoost	`print("XGBoost Classification Report")`	67%	`confusion_matrix(y_test, y_pred_xgb)` array([[562, 333], [395, 910]])

	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		
Ridge Classifier	print("Ridge Classifier Classification Report") 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	59%	confusion_matrix(y_test, y_pred_rc) array([[0, 895], [0, 1305]])
KNN	print("KNN Classification Report") 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 0.62 0.62 2200 weighted avg 0.64 0.63 0.63 2200	63%	confusion_matrix(y_test, y_pred_knn) array([[519, 376], [432, 873]])
Random Forest	print("Random Forest Classification Report:") 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	67%	confusion_matrix(y_test, y_pred_rf) array([[632, 263], [464, 841]])
SVM Classifier	print("SVM Classification Report:") SVM 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	59%	confusion_matrix(y_test, y_pred_svm) array([[0, 895], [0, 1305]])