Model Development Phase Template

Date	15 March 2024
Team ID	SWTID1720116037
Project Title	Ecommerce Shipping Prediction Using Machine Learning
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:

Support Vector Machine

```
from sklearn import svm
svm_model = svm.SVC(gamma='auto',C=5,kernel='rbf')
svm_model.fit(X_train,y_train)
y_pred = svm_model.predict(X_test)
print(classification_report(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

Artificial Neural Network

```
ann = Sequential()
ann.add(Dense(64, input_dim=8, activation='relu'))
ann.add(Dense(128, activation='relu'))
ann.add(Dense(128, activation='relu'))
ann.add(Dense(1, activation='sigmoid'))
ann.compile(loss="binary_crossentropy", optimizer='SGD',metrics=['accuracy'])
ann.fit(X_train, y_train, epochs=200, batch_size=15)
```

```
predictions = (ann.predict(X_test) > 0.5)
print(classification_report(y_test,predictions))
```

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, classification_report

# Define the hyperparameter grid
params = {
    'n_estimators': [200, 250, 300],  # Increase the number of estimators
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 5, 10],  # Add max_depth parameter
    'min_samples_split': [2, 5, 10],  # Add min_samples_split parameter
    'min_samples_leaf': [1, 5, 10]  # Add min_samples_leaf parameter
}

# Perform hyperparameter tuning
rf_model = GridSearchCV(estimator=RandomForestClassifier(), param_grid=params, scoring='accuracy', cv=5)
rf_model.fit(X_train, y_train)

# Get the best-performing model and its hyperparameters
best_model = rf_model.best_estimator_
best_params = rf_model.best_params_

# Make predictions on the test data
y_pred = best_model.predict(X_test)

# Evaluate the model's performance
print("Best Parameters:", best_params)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

Model Validation and Evaluation Report:

Model	Classification Report	Accuracy	Confusion Matrix
Random Forest	print(classification_report(y_test, y_pred)) Classification Report: precision recall f1-score support 0 0.57 0.96 0.72 895 1 0.94 0.51 0.66 1305 accuracy 0.69 2200 macro avg 0.76 0.73 0.69 2200 weighted avg 0.79 0.69 0.69 2200	69%	print(confusion_matrix(y_test,y_pred)) array([[858, 37], [636, 669]])

ANN	print(classification_report(y_test,predictions)) 69/69 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step precision recall f1-score support 0 0.57 0.84 0.68 895 1 0.84 0.57 0.68 1305 accuracy 0.68 2200 macro avg 0.71 0.71 0.68 2200 weighted avg 0.73 0.68 0.68 2200	68%	print(confusion_matrix(y_test,y_pred)) array([[769, 126], [600, 705]])
Support Vector Machine	print(classification_report(y_test,y_pred)) precision recall f1-score support 0 0.55 0.85 0.67 895 1 0.83 0.53 0.65 1305 accuracy 0.66 2200 macro avg 0.69 0.69 0.66 2200 weighted avg 0.72 0.66 0.66 2200	66%	print(confusion_matrix(y_test,y_pred)) array([[757, 138], [612, 693]])