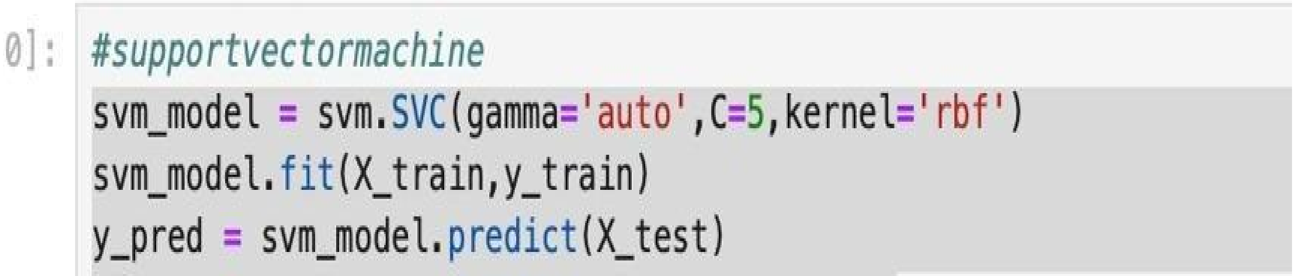
Model Development Phase Template

Date	04 June 2024
Team ID	SWTID1720183095
Project Title	Ecommerce Shipping Prediction Using Machine Learning
Maximum Marks	4 Marks

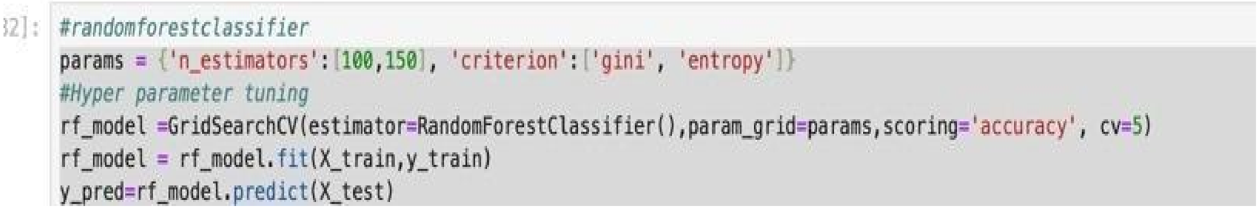
Initial Model Training Code, Model Validation and Evaluation Report

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.

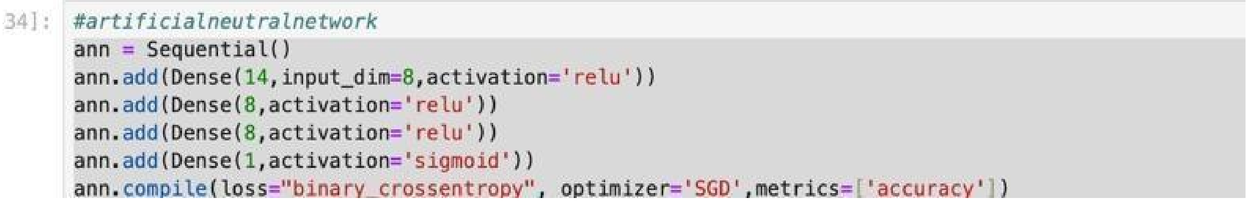
Initial Model Training Code:



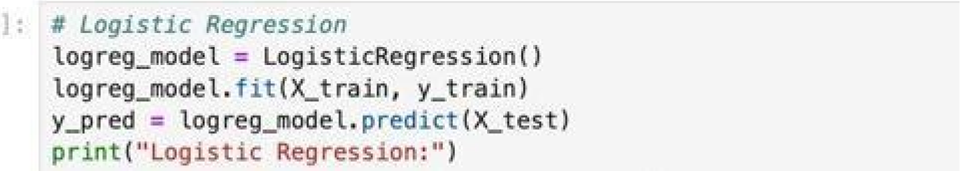
```
#supportvectormachine
svm_model = svm.SVC(gamma='auto',C=5,kernel='rbf')
svm_model.fit(X_train,y_train)
y_pred = svm_model.predict(X_test)
```



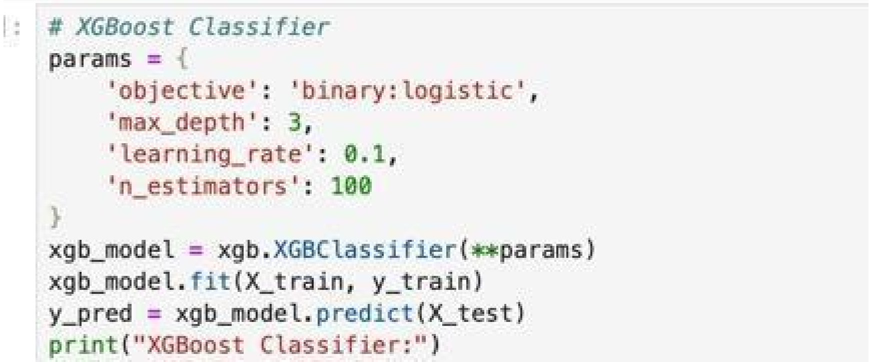
```
#randomforestclassifier
params = {'n_estimators':[100,150], 'criterion':['gini', 'entropy']}
#Hyper parameter tuning
rf_model =GridSearchCV(estimator=RandomForestClassifier(),param_grid=params,scoring='accuracy', cv=5)
rf_model = rf_model.fit(X_train,y_train)
y_pred=rf_model.predict(X_test)
```



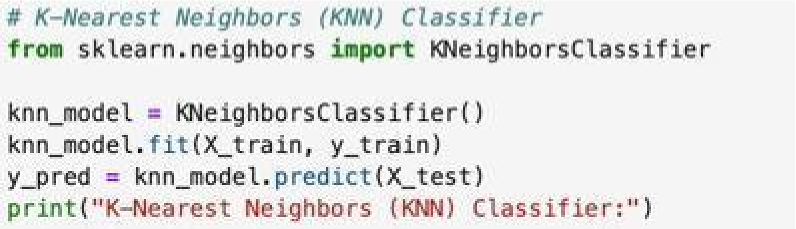
```
34]: #artificialneutralnetwork
     ann = Sequential()
     ann.add(Dense(14,input_dim=8,activation='relu'))
     ann.add(Dense(8,activation='relu'))
     ann.add(Dense(8,activation='relu'))
     ann.add(Dense(1,activation='sigmoid'))
     ann.compile(loss="binary_crossentropy", optimizer='SGD',metrics=['accuracy'])
```



```
]: # Logistic Regression
   logreg_model = LogisticRegression()
   logreg_model.fit(X_train, y_train)
   y_pred = logreg_model.predict(X_test)
   print("Logistic Regression:")
```



```
]: # XGBoost Classifier
   params = {
       'objective': 'binary:logistic',
       'max_depth': 3,
       'learning_rate': 0.1,
       'n_estimators': 100
   }
   xgb_model = xgb.XGBClassifier(**params)
   xgb_model.fit(X_train, y_train)
   y_pred = xgb_model.predict(X_test)
   print("XGBoost Classifier:")
```

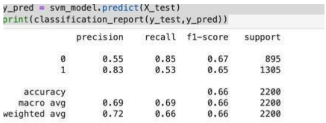
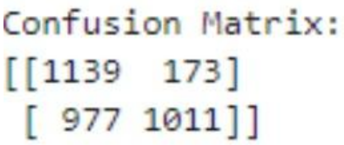
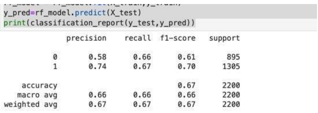
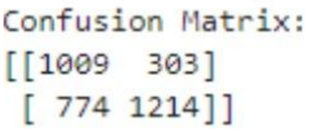
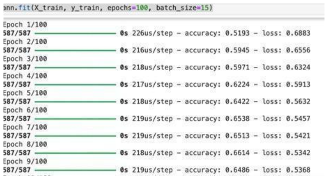
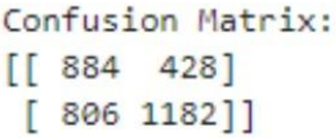
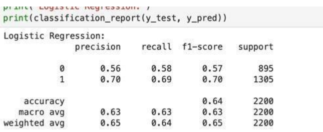
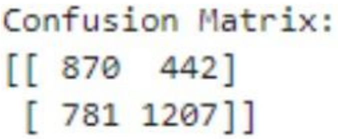
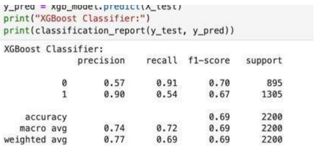
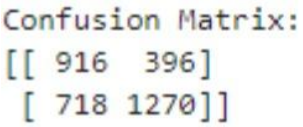
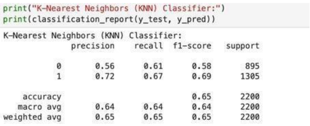
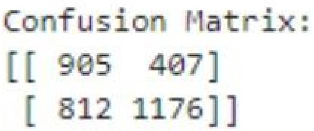


```
# K-Nearest Neighbors (KNN) Classifier
from sklearn.neighbors import KNeighborsClassifier

knn_model = KNeighborsClassifier()
knn_model.fit(X_train, y_train)
y_pred = knn_model.predict(X_test)
print("K-Nearest Neighbors (KNN) Classifier:")
```

Model Validation and Evaluation Report:

Model	Classification Report	Accuracy	Confusion Matrix

Support Vector Machine	 y_pred = svm_model.predict(X_test) 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%	 Confusion Matrix: [[1139 173] [977 1011]]
Random Forest Classifier	 y_pred=rf_model.predict(X_test) print(classification_report(y_test,y_pred)) precision recall f1-score support 0 0.58 0.66 0.61 895 1 0.74 0.67 0.70 1305 accuracy 0.67 2200 macro avg 0.66 0.66 0.66 2200 weighted avg 0.67 0.67 0.67 2200	68%	 Confusion Matrix: [[1009 303] [774 1214]]
Artificial Neutral Network	 ann.fit(X_train, y_train, epochs=100, batch_size=15) Epoch 1/100 587/587 ━━━━━━━━━━━━━━━━━━━━ 0s 226us/step - accuracy: 0.5193 - loss: 0.6883 Epoch 2/100 587/587 ━━━━━━━━━━━━━━━━━━━━ 0s 216us/step - accuracy: 0.5945 - loss: 0.6556 Epoch 3/100 587/587 ━━━━━━━━━━━━━━━━━━━━ 0s 218us/step - accuracy: 0.5971 - loss: 0.6324 Epoch 4/100 587/587 ━━━━━━━━━━━━━━━━━━━━ 0s 217us/step - accuracy: 0.6224 - loss: 0.5913 Epoch 5/100 587/587 ━━━━━━━━━━━━━━━━━━━━ 0s 218us/step - accuracy: 0.6422 - loss: 0.5632 Epoch 6/100 587/587 ━━━━━━━━━━━━━━━━━━━━ 0s 219us/step - accuracy: 0.6538 - loss: 0.5457 Epoch 7/100 587/587 ━━━━━━━━━━━━━━━━━━━━ 0s 219us/step - accuracy: 0.6513 - loss: 0.5421 Epoch 8/100 587/587 ━━━━━━━━━━━━━━━━━━━━ 0s 218us/step - accuracy: 0.6614 - loss: 0.5342 Epoch 9/100 587/587 ━━━━━━━━━━━━━━━━━━━━ 0s 219us/step - accuracy: 0.6486 - loss: 0.5368	67%	 Confusion Matrix: [[884 428] [806 1182]]
Logistic Classifier	 print("Logistic Regression:") print(classification_report(y_test, y_pred)) Logistic Regression: precision recall f1-score support 0 0.56 0.58 0.57 895 1 0.70 0.69 0.70 1305 accuracy 0.64 2200 macro avg 0.63 0.63 0.63 2200 weighted avg 0.65 0.64 0.65 2200	68%	 Confusion Matrix: [[870 442] [781 1207]]
XGBoost Classifier	 y_pred = xgb_model.predict(X_test) print("XGBoost Classifier:") print(classification_report(y_test, y_pred)) XGBoost Classifier: precision recall f1-score support 0 0.57 0.91 0.70 895 1 0.90 0.54 0.67 1305 accuracy 0.69 2200 macro avg 0.74 0.72 0.69 2200 weighted avg 0.77 0.69 0.69 2200	69%	 Confusion Matrix: [[916 396] [718 1270]]
KNN Neighbours Classifier	 print("K-Nearest Neighbors (KNN) Classifier:") print(classification_report(y_test, y_pred)) K-Nearest Neighbors (KNN) Classifier: precision recall f1-score support 0 0.56 0.61 0.58 895 1 0.72 0.67 0.69 1305 accuracy 0.65 2200 macro avg 0.64 0.64 0.64 2200 weighted avg 0.65 0.65 0.65 2200	65%	 Confusion Matrix: [[905 407] [812 1176]]