Model Development Phase

Date	10 July 2024
Team ID	SWTID1720084775
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 is showcased in the following table through screenshots. The model validation and evaluation report include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

Initial Model Training Code:

```
[82] #INITIALISING MULTIPLE MODELS

     from sklearn.linear_model import LogisticRegression
     lr=LogisticRegression()

     from sklearn.neighbors import KNeighborsClassifier
     knn=KNeighborsClassifier(n_neighbors=7)

     from sklearn.svm import SVC
     model = SVC(kernel="linear")

     from sklearn.naive_bayes import GaussianNB
     nb = GaussianNB()

     from sklearn.ensemble import RandomForestClassifier
     rf = RandomForestClassifier(n_estimators=7, criterion='entropy',random_state=0)

     import xgboost as xgb
     xg = xgb.XGBClassifier()
```

```
[83] #Fitting x_train and y_train

     lr.fit(x_train, y_train)

     knn.fit(x_train, y_train)

     model.fit(x_train, y_train)

     nb.fit(x_train, y_train)

     rf.fit(x_train, y_train)

     xg.fit(x_train, y_train)
```

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)
```

```
[84] #predictions

     pred_lr = lr.predict(x_test)

     pred_knn = knn.predict(x_test)

     pred_model = model.predict(x_test)

     pred_nb = nb.predict(x_test)

     pred_rf = rf.predict(x_test)

     pred_xg = xg.predict(x_test)
```

```
[85] from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

```
[86] cr_lr = classification_report(y_test, pred_lr)
     cr_knn = classification_report(y_test, pred_knn)
     cr_svc = classification_report(y_test, pred_model)
     cr_nb = classification_report(y_test, pred_nb)
     cr_rf = classification_report(y_test, pred_rf)
     cr_xg = classification_report(y_test, pred_xg)
```

```
[87] as_lr = accuracy_score(y_test, pred_lr)
     as_knn = accuracy_score(y_test, pred_knn)
     as_svc = accuracy_score(y_test, pred_model)
     as_nb = accuracy_score(y_test, pred_nb)
     as_rf = accuracy_score(y_test, pred_rf)
     as_xg = accuracy_score(y_test, pred_xg)
```

```
[88] cm_lr = confusion_matrix(y_test, pred_lr)
     cm_knn = confusion_matrix(y_test, pred_knn)
     cm_svc = confusion_matrix(y_test, pred_model)
     cm_nb = confusion_matrix(y_test, pred_nb)
     cm_rf = confusion_matrix(y_test, pred_rf)
     cm_xg = confusion_matrix(y_test, pred_xg)
```

```
[89] from prettytable import PrettyTable

     # Specify the Column Names while initializing the Table
     myTable = PrettyTable(["Metrics", "Logistic Regression", "KNN", "SVC", "Naive Bayes", "Random Forest", "XGBoost"])

     # Add rows
     myTable.add_row(["Classification Report", cr_lr, cr_knn, cr_svc, cr_nb, cr_rf, cr_xg])
     myTable.add_row(["Accuracy Score", as_lr, as_knn, as_svc, as_nb, as_rf, as_xg])
     myTable.add_row(["Confusion Matrix", cm_lr, cm_knn, cm_svc, cm_nb, cm_rf, cm_xg])
     print(myTable)
```

```
+----------------------+-----------------------------------------------------+-----------------------------------------------------+
|       Metrics        |                 Logistic Regression                 |                         KNN                         |
+----------------------+-----------------------------------------------------+-----------------------------------------------------+
| Classification Report |              precision    recall  f1-score   support |              precision    recall  f1-score   support |
|                      |                                                     |                                                     |
|                      |           0       0.63      0.67      0.65       872 |           0       0.62      0.72      0.67       872 |
|                      |           1       0.66      0.63      0.64       903 |           1       0.68      0.58      0.63       903 |
|                      |                                                     |                                                     |
|                      |    accuracy                           0.65      1775 |    accuracy                           0.65      1775 |
|                      |   macro avg       0.65      0.65      0.65      1775 |   macro avg       0.65      0.65      0.65      1775 |
|                      | weighted avg      0.65      0.65      0.65      1775 | weighted avg      0.65      0.65      0.65      1775 |
|                      |                                                     |                                                     |
|    Accuracy Score    |                  0.6467605633802817                 |                  0.6473239436619719                 |
|   Confusion Matrix   |                     [[583 289]                      |                     [[627 245]                      |
|                      |                     [338 565]]                      |                     [381 522]]                      |
+----------------------+-----------------------------------------------------+-----------------------------------------------------+
```

```
[ ] from tensorflow import keras
    from keras.models import Sequential
    from keras.layers import Dense
```

```
[ ] model = Sequential()
```

```
[ ] x_train_ANN.shape
```

```
(7097, 5)
```

```
[ ] model.add(Dense(units = 12,kernel_initializer = 'random_uniform',activation = 'relu')) #input layer
```

```
[ ] model.add(Dense(units =12,kernel_initializer = 'random_uniform',activation = 'relu')) # 1st hidden layer
```

```
[ ] model.add(Dense(units =6,kernel_initializer = 'random_uniform',activation = 'relu')) # 1st hidden layer
```

```
[ ] model.add(Dense(units = 1 ,kernel_initializer = 'random_uniform' ,activation = 'sigmoid')) #Output Layer
```

```
[ ] model.compile (optimizer = 'adam',loss = 'binary_crossentropy',metrics = ['accuracy'])
```

```
[ ] model.fit(x_train_ANN, y_train_ANN, batch_size=32, epochs = 250)
```

```
[ ] y_pred_prob = model.predict(x_test_ANN)
    y_pred_ANN = (y_pred_prob > 0.5).astype(int)
```

```
56/56 [==============================] - 0s 1ms/step
```

```
[ ] y_pred_ANN
```

```
array([[0],
       [0],
       [0],
       ...,
       [0],
       [0],
       [0]])
```

```
[ ] y_test_ANN.flatten()
```

```
array([0, 1, 0, ..., 0, 0, 1])
```

```
[ ] y_test_ANN[0]
```

```
0
```

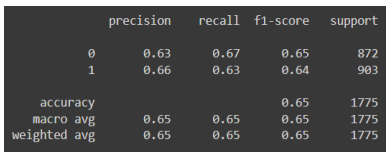
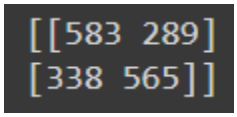
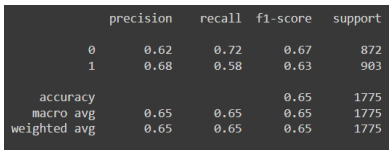
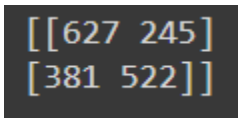
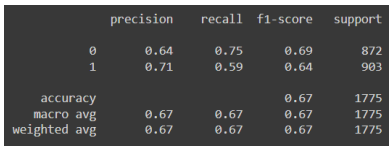
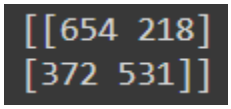
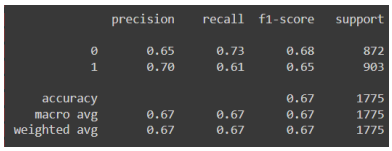
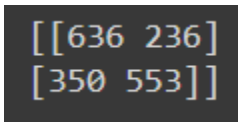
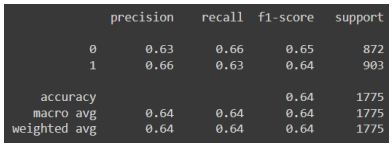
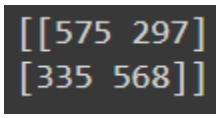
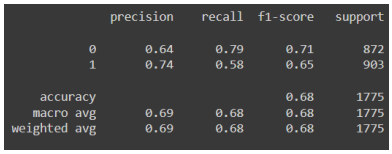
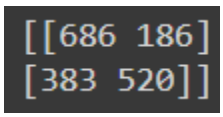
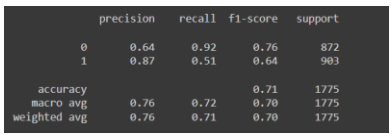
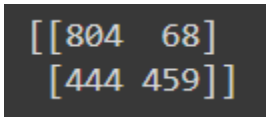
```
[ ] y_pred_ANN[0]
```

```
array([0.3316071], dtype=float32)
```

```
[ ] ANN_cr = classification_report(y_test_ANN, y_pred_ANN)
    ANN_as = accuracy_score(y_test_ANN, y_pred_ANN)
    ANN_cm = confusion_matrix(y_test_ANN, y_pred_ANN)
```

```
print(f"{ANN_cr}\n{ANN_as}\n{ANN_cm}")
```

Model Validation and Evaluation Report:

Model	Classification Report	Accuracy	Confusion Matrix
Logistic Regressio		0.64676056	
K-Nearest Neighbours (KNN)		0.64732394	
Support Vector Machine (SVM)		0.66760563	
Gaussian Naive Bayes		0.66985915	
Random Forest Generator		0.64394366	
XG Boost		0.67943662	
ANN		0.71154929	

	precision	recall	f1-score	support
0	0.63	0.67	0.65	872
1	0.66	0.63	0.64	903
accuracy			0.65	1775
macro avg	0.65	0.65	0.65	1775
weighted avg	0.65	0.65	0.65	1775

```
[[583 289]
 [338 565]]
```

	precision	recall	f1-score	support
0	0.62	0.72	0.67	872
1	0.68	0.58	0.63	903
accuracy			0.65	1775
macro avg	0.65	0.65	0.65	1775
weighted avg	0.65	0.65	0.65	1775

```
[[627 245]
 [381 522]]
```

	precision	recall	f1-score	support
0	0.64	0.75	0.69	872
1	0.71	0.59	0.64	903
accuracy			0.67	1775
macro avg	0.67	0.67	0.67	1775
weighted avg	0.67	0.67	0.67	1775

```
[[654 218]
 [372 531]]
```

	precision	recall	f1-score	support
0	0.65	0.73	0.68	872
1	0.70	0.61	0.65	903
accuracy			0.67	1775
macro avg	0.67	0.67	0.67	1775
weighted avg	0.67	0.67	0.67	1775

```
[[636 236]
 [350 553]]
```

	precision	recall	f1-score	support
0	0.63	0.66	0.65	872
1	0.66	0.63	0.64	903
accuracy			0.64	1775
macro avg	0.64	0.64	0.64	1775
weighted avg	0.64	0.64	0.64	1775

```
[[575 297]
 [335 568]]
```

	precision	recall	f1-score	support
0	0.64	0.79	0.71	872
1	0.74	0.58	0.65	903
accuracy			0.68	1775
macro avg	0.69	0.68	0.68	1775
weighted avg	0.69	0.68	0.68	1775

```
[[686 186]
 [383 520]]
```

	precision	recall	f1-score	support
0	0.64	0.92	0.76	872
1	0.87	0.51	0.64	903
accuracy			0.71	1775
macro avg	0.76	0.72	0.70	1775
weighted avg	0.76	0.71	0.70	1775

```
[[804  68]
 [444 459]]
```