Model Development Phase Template

Date	11 July 2024
Team ID	SWTID1720097765
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. Our initial model training code, displayed in the screenshot below, outlines the process of preparing and training the model using our dataset. The model validation and evaluation report provides comprehensive insights into the model's performance, including classification reports, accuracy scores, and confusion matrices for various models. These evaluations highlight the effectiveness of our approach and assist in selecting the optimal model for deployment. Detailed results are showcased through the following screenshots, demonstrating the comparative analysis of each model.

Initial Model Training Code:

DecisionTreeClassifier Model

```
# Train and Build the model using DecisionTreeClassifier
def decision_tree_model(x_train,y_train,x_test,y_test):
    df=make_pipeline(StandardScaler(),DecisionTreeClassifier(criterion='entropy',random_state=1))
    df.fit(x_train,y_train)
    print('--DecisionTreeClassifier')
    print('Train Score:',df.score(x_train,y_train))
    print('Test Score:',df.score(x_test,y_test))
    print()
    return df
```

```
df=decision_tree_model(x_train,y_train,x_test,y_test)
pred=df.predict(x_test)
print(pred) # predicted values
print(y_test) # actual values
accuracy=accuracy_score(y_test,pred)
print()
# Printing Accuracy Scorce of DecisionTreeClassifier model
print("accuracy score of DecisionTreeClassifier model is:",accuracy)
```

```
--DecisionTreeClassifier
Train Score: 1.0
Test Score: 0.6468181818181818

[0 1 0 ... 1 0 0]
7212    1
7220    0
4637    1
2709    1
8161    0
       ..
1628    1
901     1
8903    0
7018    1
8349    1
Name: Reached.on.Time_Y.N, Length: 2200, dtype: int64

accuracy score of DecisionTreeClassifier model is: 0.6468181818181818
```

LogisticRegression Model

```
def logistic_regression_model(x_train,y_train,x_test,y_test):
    lg = make_pipeline(StandardScaler(),LogisticRegression(random_state=1234))
    lg.fit(x_train,y_train)
    print('--Logistic Regression')
    print('Train Score:',lg.score(x_train,y_train))
    print('Test Score:',lg.score(x_test,y_test))
    print()
    return lg
```

```
lg=logistic_regression_model(x_train,y_train,x_test,y_test)
pred=lg.predict(x_test)
print(pred) # predicted values
print(y_test) # actual values
accuracy=accuracy_score(y_test,pred)
print()
# Printing Accuracy Scorce of LogisticRegression model
print("accuracy score of LogisticRegression model is:",accuracy)
```

```
--Logistic Regression
Train Score: 0.6416638254347085
Test Score: 0.6409090909090909

[1 1 0 ... 1 0 1]
7212    1
7220    0
4637    1
2709    1
8161    0
       ..
1628    1
901     1
8903    0
7018    1
8349    1
Name: Reached.on.Time_Y.N, Length: 2200, dtype: int64

accuracy score of LogisticRegression model is: 0.6409090909090909
```

LogisticRegressionCV Model

```
def logistic_regressionCV_model(x_train,y_train,x_test,y_test):
    lcv = make_pipeline(StandardScaler(),LogisticRegressionCV(random_state=1234))
    lcv.fit(x_train,y_train)
    print('--Logistic Regression CV')
    print('Train Score:',lcv.score(x_train,y_train))
    print('Test Score:',lcv.score(x_test,y_test))
    print()
    return lcv
```

```
lcv=logistic_regressionCV_model(x_train,y_train,x_test,y_test)
pred=lcv.predict(x_test)
print(pred) # predicted values
print(y_test) # actual values
accuracy=accuracy_score(y_test,pred)
print()
 # Printing Accuracy Scorce of LogisticRegressionCV Model
print("accuracy score of LogisticRegressionCV Model is:",accuracy)
```

```
--Logistic Regression CV
Train Score: 0.6446187066712127
Test Score: 0.6413636363636364

[1 1 0 ... 1 0 1]
7212    1
7220    0
4637    1
2709    1
8161    0
       ..
1628    1
901     1
8903    0
7018    1
8349    1
Name: Reached.on.Time_Y.N, Length: 2200, dtype: int64

accuracy score of LogisticRegressionCV Model is: 0.6413636363636364
```

XGBClassifier Model

```
def XGB_classifier_model(x_train,y_train,x_test,y_test):
    xgb = make_pipeline(StandardScaler(),XGBClassifier(n_estimators=300,n_jobs=-1,random_state=1234))
    xgb.fit(x_train,y_train)
    print('--XGBoost')
    print('Train Score:',xgb.score(x_train,y_train))
    print('Test Score:',xgb.score(x_test,y_test))
    print()
    return xgb
```

```
xgb=XGB_classifier_model(x_train,y_train,x_test,y_test)
pred=xgb.predict(x_test)
print(pred) # predicted values
print(y_test) # actual values
accuracy=accuracy_score(y_test,pred)
print()
# Printing Accuracy Scorce of XGBClassifier Model
print("accuracy score of XGBClassifier Model is:",accuracy)
```

```
--XGBoost
Train Score: 0.99181725196045
Test Score: 0.6463636363636364

[1 1 0 ... 1 0 0]
7212    1
7220    0
4637    1
2709    1
8161    0
       ..
1628    1
901     1
8903    0
7018    1
8349    1
Name: Reached.on.Time_Y.N, Length: 2200, dtype: int64

accuracy score of XGBClassifier Model is: 0.6463636363636364
```

RidgeClassifier Model

```
def ridge_classifier_model(x_train,y_train,x_test,y_test):
    rg = make_pipeline(StandardScaler(),RidgeClassifier(random_state=1234))
    rg.fit(x_train,y_train)
    print('--Ridge Classifier')
    print('Train Score:',rg.score(x_train,y_train))
    print('Test Score:',rg.score(x_test,y_test))
    print()
    return rg
```

```
rg=ridge_classifier_model(x_train,y_train,x_test,y_test)
pred=rg.predict(x_test)
print(pred) # predicted values
print(y_test) # actual values
accuracy=accuracy_score(y_test,pred)
print()
# Printing Accuracy Scorce of RidgeClassifier Model
print("accuracy score of RidgeClassifier Model is:",accuracy)
```

```
--Ridge Classifier
Train Score: 0.6529151039890897
Test Score: 0.649090909090909

[1 1 0 ... 1 0 1]
7212    1
7220    0
4637    1
2709    1
8161    0
       ..
1628    1
901     1
8903    0
7018    1
8349    1
Name: Reached.on.Time_Y.N, Length: 2200, dtype: int64

accuracy score of RidgeClassifier Model is: 0.649090909090909
```

KNeighborsClassifier Model

```
def k_neighbors_classifier_model(x_train,y_train,x_test,y_test):
    knn = make_pipeline(StandardScaler(),KNeighborsClassifier())
    knn.fit(x_train,y_train)
    print('--KNN')
    print('Train Score:',knn.score(x_train,y_train))
    print('Test Score:',knn.score(x_test,y_test))
    print()
    return knn
```

```
knn=k_neighbors_classifier_model(x_train,y_train,x_test,y_test)
pred=knn.predict(x_test)
print(pred) # predicted values
print(y_test) # actual values
accuracy=accuracy_score(y_test,pred)
print()
# Printing Accuracy Scorce of KNeighborsClassifier Model
print("accuracy score of KNeighborsClassifier Model is:",accuracy)
```

```
--KNN
Train Score: 0.7734969882941243
Test Score: 0.64

[0 1 0 ... 1 0 1]
7212    1
7220    0
4637    1
2709    1
8161    0
       ..
1628    1
901     1
8903    0
7018    1
8349    1
Name: Reached.on.Time_Y.N, Length: 2200, dtype: int64

accuracy score of KNeighborsClassifier Model is: 0.64
```

RandomForestClassifier Model

```
def random_forest_classifier_model(x_train,y_train,x_test,y_test):
    rf = make_pipeline(StandardScaler(),RandomForestClassifier(random_state=1234))
    rf.fit(x_train,y_train)
    print('--Random Forest')
    print('Train Score:',rf.score(x_train,y_train))
    print('Test Score:',rf.score(x_test,y_test))
    print()
    return rf
```

```
rf=random_forest_classifier_model(x_train,y_train,x_test,y_test)
pred=rf.predict(x_test)
print(pred) # predicted values
print(y_test) # actual values
accuracy=accuracy_score(y_test,pred)
print()
# Printing Accuracy Scorce of RandomForestClassifier Model
print("accuracy score of RandomForestClassifier Model is:",accuracy)
```

```
--Random Forest
Train Score: 1.0
Test Score: 0.6563636363636364

[0 1 0 ... 1 0 0]
7212    1
7220    0
4637    1
2709    1
8161    0
       ..
1628    1
901     1
8903    0
7018    1
8349    1
Name: Reached.on.Time_Y.N, Length: 2200, dtype: int64

accuracy score of RandomForestClassifier Model is: 0.6563636363636364
```

Model Validation and Evaluation Report:

Model	Classification Report	Accuracy	Confusion Matrix
Decision Tree	`# Printing classification report of DecisionTreeClassifier model` `print("classification report of DecisionTreeClassifier model: ")` `print(classification_report(y_test,pred))` classification report of DecisionTreeClassifier model: precision recall f1-score support 0 0.56 0.58 0.57 896 1 0.71 0.69 0.70 1304 accuracy 0.65 2200 macro avg 0.64 0.64 0.64 2200 weighted avg 0.65 0.65 0.65 2200	64.68%	`# Printing confusion matrix of DecisionTreeClassifier model` `print("confusion matrix of DecisionTreeClassifier model:")` `print(confusion_matrix(y_test,pred))` confusion matrix of DecisionTreeClassifier model: [[524 372] [405 899]]
Logistic Regression	`# Printing classification report of LogisticRegression model` `print("classification report of LogisticRegression model: ")` `print(classification_report(y_test,pred))` classification report of LogisticRegression model: precision recall f1-score support 0 0.56 0.59 0.57 896 1 0.70 0.68 0.69 1304 accuracy 0.64 2200 macro avg 0.63 0.63 0.63 2200 weighted avg 0.64 0.64 0.64 2200	64.09%	`# Printing confusion matrix of LogisticRegression model` `print("confusion matrix of LogisticRegression model:")` `print(confusion_matrix(y_test,pred))` confusion matrix of LogisticRegression model: [[526 370] [420 884]]

Logistic Regression CV	```# Printing classification report of LogisticRegressionCV Model print("classification report of LogisticRegressionCV Model: ") print(classification_report(y_test,pred))``` ```classification report of LogisticRegressionCV Model: precision recall f1-score support 0 0.56 0.55 0.56 896 1 0.70 0.70 0.70 1304 accuracy 0.64 2200 macro avg 0.63 0.63 0.63 2200 weighted avg 0.64 0.64 0.64 2200```	64.13%	```# Printing confusion matrix of LogisticRegressionCV Model print("confusion matrix of LogisticRegressionCV Model:") print(confusion_matrix(y_test,pred))``` ```confusion matrix of LogisticRegressionCV Model: [[497 399] [390 914]]```
XGB classifier	```# Printing classification report of XGBClassifier Model print("classification report of XGBClassifier Model: ") print(classification_report(y_test,pred))``` ```classification report of XGBClassifier Model: precision recall f1-score support 0 0.56 0.58 0.57 896 1 0.71 0.69 0.70 1304 accuracy 0.65 2200 macro avg 0.63 0.64 0.64 2200 weighted avg 0.65 0.65 0.65 2200```	64.63%	```# Printing confusion matrix of XGBClassifier Model print("confusion matrix of XGBClassifier Model:") print(confusion_matrix(y_test,pred))``` ```confusion matrix of XGBClassifier Model: [[521 375] [403 901]]```
Ridge classifier	```# Printing classification report of RidgeClassifier Model print("classification report of RidgeClassifier Model: ") print(classification_report(y_test,pred))``` ```classification report of RidgeClassifier Model: precision recall f1-score support 0 0.57 0.59 0.58 896 1 0.71 0.69 0.70 1304 accuracy 0.65 2200 macro avg 0.64 0.64 0.64 2200 weighted avg 0.65 0.65 0.65 2200```	64.9%	```# Printing confusion matrix of RidgeClassifier Model print("confusion matrix of RidgeClassifier Model:") print(confusion_matrix(y_test,pred))``` ```confusion matrix of RidgeClassifier Model: [[532 364] [408 896]]```

<table>
<tr>
<td>KNN</td>
<td>

```
# Printing classification report of KNeighborsClassifier Model
print("classification report of KNeighborsClassifier Model: ")
print(classification_report(y_test,pred))
```

```
classification report of KNeighborsClassifier Model:
              precision    recall  f1-score   support

           0       0.56      0.58      0.57       896
           1       0.70      0.68      0.69      1304

    accuracy                           0.64      2200
   macro avg       0.63      0.63      0.63      2200
weighted avg       0.64      0.64      0.64      2200
```

</td>
<td>64%</td>
<td>

```
# Printing confusion matrix of KNeighborsClassifier Model
print("confusion matrix of KNeighborsClassifier Model:")
print(confusion_matrix(y_test,pred))
```

```
confusion matrix of KNeighborsClassifier Model:
[[523 373]
 [419 885]]
```

</td>
</tr>
<tr>
<td>Random Forest</td>
<td>

```
# Printing classification report of RandomForestClassifier Model
print("classification report of RandomForestClassifier Model: ")
print(classification_report(y_test,pred))
```

```
classification report of RandomForestClassifier Model:
              precision    recall  f1-score   support

           0       0.56      0.68      0.62       896
           1       0.74      0.64      0.69      1304

    accuracy                           0.66      2200
   macro avg       0.65      0.66      0.65      2200
weighted avg       0.67      0.66      0.66      2200
```

</td>
<td>65.63%</td>
<td>

```
# Printing confusion matrix of RandomForestClassifier Model
print("confusion matrix of RandomForestClassifier Model:")
print(confusion_matrix(y_test,pred))
```

```
confusion matrix of RandomForestClassifier Model:
[[609 287]
 [469 835]]
```

</td>
</tr>
</table>