

Model Development Phase Template

Date	15 July 2024
Team ID	740057
Project Title	Airline Reviews Classification
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:

```

from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier(n_estimators=10, criterion='entropy', random_state=2)

rfc.fit(X_train,y_train)

RandomForestClassifier
RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=2)

pred_rfc=rfc.predict(X_test)
pred_rfc
array([1, 0, 0, ..., 1, 1, 0])

from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier(criterion='entropy', random_state=50)

dtc.fit(X_train,y_train)

DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', random_state=50)

pred_dt=dtc.predict(X_test)
pred_dt
array([1, 0, 0, ..., 1, 1, 0])

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score,
fpr_dt, tpr_dt, threshold_dt=roc_curve(y_test, pred_dt)
print(classification_report(y_test, pred_dt))
roc_auc_dt=auc(fpr_dt, tpr_dt)
print("roc_auc_dt", roc_auc_dt)
cm_dt=confusion_matrix(y_test,pred_dt)
print("cm_dt:",cm_dt)
as_dt=accuracy_score(y_test, pred_dt)
print("as_dt:", as_dt)

precision    recall  f1-score   support

0           0.95      0.95      0.95        3116
1           0.95      0.95      0.95        3030

accuracy          0.95
macro avg          0.95
weighted avg       0.95
  
```

Model Validation and Evaluation Report:

Model	Classification Report	Accuracy	Confusion Matrix
Random forest classifier	<pre> from sklearn.ensemble import RandomForestClassifier rfc=RandomForestClassifier(n_estimators=10, criterion='entropy', random_state=2) rfc.fit(X_train,y_train) RandomForestClassifier(RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=2) pred_rfc=rfc.predict(X_test) pred_rfc array([1, 0, 0, ..., 1, 1, 0]) from sklearn.metrics import classification_report, confusion_matrix, accuracy_score fpr_rfc, tpr_rfc, threshold_rfc=roc_curve(y_test,pred_rfc) print(classification_report(y_test, pred_rfc)) roc_auc_rfc=auc(fpr_rfc, tpr_rfc) print("roc_auc_rfc", roc_auc_rfc) cm_rfc=confusion_matrix(y_test, pred_rfc) print("cm_rfc",cm_rfc) as_rfc=accuracy_score(y_test, pred_rfc) print("as_rfc",as_rfc) precision recall f1-score support 0 0.95 0.96 0.96 3116 1 0.96 0.95 0.96 3030 accuracy 0.96 0.96 0.96 6146 macro avg 0.96 0.96 0.96 6146 weighted avg 0.96 0.96 0.96 6146 </pre>	95%	<pre> cm_rfc=confusion_matrix(y_test, pred_rfc) print("cm_rfc",cm_rfc) as_rfc=accuracy_score(y_test, pred_rfc) </pre>
Decision tree	<pre> from sklearn.tree import DecisionTreeClassifier dtc=DecisionTreeClassifier(criterion='entropy', random_state=50) dtc.fit(X_train,y_train) DecisionTreeClassifier(DecisionTreeClassifier(criterion='entropy', random_state=50) pred_dt=dtc.predict(X_test) pred_dt array([1, 0, 0, ..., 1, 1, 0]) from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, fpr_dt, tpr_dt, threshold_dt=roc_curve(y_test, pred_dt) print(classification_report(y_test, pred_dt)) roc_auc_dt=auc(fpr_dt, tpr_dt) print("roc_auc_dt", roc_auc_dt) cm_dt=confusion_matrix(y_test, pred_dt) print("cm_dt",cm_dt) as_dt=accuracy_score(y_test, pred_dt) print("as_dt",as_dt) precision recall f1-score support 0 0.95 0.95 0.95 3116 1 0.95 0.95 0.95 3030 accuracy 0.95 0.95 0.95 6146 macro avg 0.95 0.95 0.95 6146 weighted avg 0.95 0.95 0.95 6146 </pre>	94%	<pre> cm_dt=confusion_matrix(y_test,pred_dt) print("cm_dt",cm_dt) as_dt=accuracy_score(y_test, pred_dt) </pre>

Logistic Regression

92%

```
[ ] from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()

[ ] lr.fit(X_train,y_train)

[ ] lr.predict(X_test)
pred_lr
array([1, 0, 0, ..., 1, 1, 0])

[ ] from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
fpr_lr, tpr_lr, threshold_lr=roc_curve(y_test, pred_lr)
print(classification_report(y_test, pred_lr))
roc_auc_lr=auc(fpr_lr, tpr_lr)
print("roc_auc_lr:", roc_auc_lr)
cm_lr=confusion_matrix(y_test, pred_lr)
print("cm_lr:",cm_lr)
as_lr=accuracy_score(y_test, pred_lr)
print("as_lr:",as_lr)
```

	precision	recall	f1-score	support
0	0.93	0.92	0.92	3116
1	0.92	0.93	0.92	3030
accuracy			0.92	6146
macro avg	0.92	0.92	0.92	6146
weighted avg	0.92	0.92	0.92	6146

```
In [83]: confusion_matrix(y_test, y_test_pred)
Out[83]: array([[8467, 1924],
               [1042, 4198]], dtype=int64)
```

```
roc_auc_lr: 0.922439384583277
cm_lr: [[2803 255]
        [248 2088]]
as_lr: 0.9223885453953791
```

XGB

96%

```
[ ] from xgboost import XGBClassifier
xgb=XGBClassifier()

[ ] xgb.fit(X_train,y_train)

[ ] xgb.predict(X_test)

[ ] from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
fpr_xgb, tpr_xgb, threshold_xgb=roc_curve(y_test, pred_xgb)
print(classification_report(y_test, pred_xgb))
roc_auc_xgb=auc(fpr_xgb, tpr_xgb)
print("roc_auc_xgb:", roc_auc_xgb)
cm_xgb=confusion_matrix(y_test, pred_xgb)
print("cm_xgb:",cm_xgb)
as_xgb=accuracy_score(y_test, pred_xgb)
print("as_xgb:",as_xgb)
```

	precision	recall	f1-score	support
0	0.96	0.97	0.96	3116
1	0.97	0.96	0.96	3030
accuracy			0.96	6146
macro avg	0.96	0.96	0.96	6146
weighted avg	0.96	0.96	0.96	6146

```
roc_auc_xgb: 0.9630194630502845
cm_xgb: [[3011 105]
         [122 2908]]
as_xgb: 0.9630654883957045
```