Project Development Phase Model Performance Test

Team ID: 592662

Project Name : AIRLINE REVIEW CLASSIFICATION USING MACHINE LEARNING

Maximum Marks: 4 Marks

Model Performance Testing:

Project team shall fill the following information in model performance testing template.

S.	Parame	Values	Screenshot
N	ter		
0.	"		
<u> </u>	Metrics	Regression Model:	In [84]: #5.Evaluation of the model
		MAE - 9798.736732205687,	7- for), for all and and a fine
		MSE - 96015241.547077,	In [85]: from sklearm import metrics
		RMSE - 9798.736732205687, R2 score - 0.9258962347553819	In [86]: # R- Square # evaluating testing accuracy #ACCURACY
			print(metrics.r2_score(y_test,y_pred))
			0.9258962347553819
			<pre>In [87]: #mean squared error print(metrics.mean_squared_error(y_test,y_pred))</pre>
			96015241.547077
			In [88]: # RMSE (Root Mean Square Error) print(np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
			9798.736732205687
		Classification Model:	File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (pythemel) O
		Confusion Matrix –	□ + 3x ② E ↑ + ► Run ■ C → Code ∨ □ In [18]: # Frint classification results
		[[2999 66]	<pre>print(f"Accuracy: (accuracy)") print(f"Precision: {precision}")</pre>
		[118 1452]]	print(f"Recall: [recall]") print(f"Fi Score: [f]")
		Accuracy Score- 0.9603020496224379	print(f*1core: {r1}) print(f*8cc-Auc Score: {roc_auc}*)
		1	Accuracy: 0.9603020090224379
		Classification Report –	Precision: 0.9565217391384348
		precision rec all f1-score support	Recall: 0.924846743312102 F1 Score: 0.9404145077720207 ROC-AUC Score: 0.9516536611215595
			9600 (2010) (2010) (2010) (2010) (2010)
		0 0.96 0	In [19]: # Print Confusion Matrix print("Confusion Matrix:") print(a)
		.98 0.97 3065	Confusion Matrix:
		1 0.96 0	[[299 66] [118 1452]]
		.92 0.94 1570	1 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
			In [20]: # Print Classification Report print("Classification Report") print(classification rep)
		accuracy	Classification Report:
		0.96 4635	precision recall f1-score support
		macro avg 0.96 0	0 0.96 0.98 0.97 3065 1 0.96 0.92 0.94 1570
		.95 0.96 4635	accuracy 9.96 4635
		weighted avg 0.96 0	macro avg 0.96 0.95 0.96 4635 weighted avg 0.96 0.96 0.96 4635
		.96 0.96 4635	

Tune the Model

Hyperparameter Tuning –

```
Best Hyperparameters:
{'classifier__n_estimators': 5
0, 'classifier__min_samples_sp
lit': 5, 'classifier__min_samp
les_leaf': 1, 'classifier__max
_depth': None, 'classifier__bo
otstrap': False}
```

Perform RandomizedSearchCV(pipeline, param_distributions=param_dist, n_iter=10, cv=5, scoring='accuracy', random_state=47 random_search.fit(X, y) # Print the best hyperparameters best params = random_search.best_params_ print("Best Hyperparameters:") print(best_params)

Best Hyperparameters:

{'classifier_n_estimators': 50, 'classifier_min_samples_split': 5, 'classifier_min_samples_leaf': 1, 'classifier_max_dept h': None, 'classifier_bootstrap': False}

Validation Method -

Cross-Validation Results:
Mean Accuracy: 0.9502394151571
402
Standard Deviation: 0.00148976

Standard Deviation: 0.00148976

97005680104

```
categorical transformer = Pipeline(steps=)
    ('imputer', SimpleImputer(strategy='most_frequent')),
('onehot', OneHotEncoder(handle_unknown='ignore'))
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_features),
        ('cat', categorical_transformer, categorical_features)
# Define the RandomForestClassifier pipeline
# Define the stratified k-fold cross-validation
stratified_kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
# Perform cross-validation
cv_results = cross_val_score(pipeline, X, y, cv=stratified_kfold, scoring='accuracy')
# Print the cross-validation results
print("Cross-Validation Results:")
print("Mean Accuracy:", np.mean(cv_results))
print("Standard Deviation:", np.std(cv_results))
```

Cross-Validation Results: Mean Accuracy: 0.9502394151571402 Standard Deviation: 0.0014897697005680104 1.

```
In [84]: #5.Evaluation of the model
       In [85]: from sklearn import metrics
       In [86]: # R- Square
                # evaluating testing accuracy
                #ACCURACY
                print(metrics.r2_score(y_test,y_pred))
                0.9258962347553819
       In [87]: #mean squared error
                print(metrics.mean_squared_error(y_test,y_pred))
                96015241,547077
       In [88]: # RMSE (Root Mean Square Error)
                print(np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
                9798.736732205687
2.
   File Edit View Insert
                              Cell Kernel Widgets
                                                                                                                 Not Trusted
                                                                                                                              Python 3 (ipykernel) O
                                                        Help
   B + 3< ② B ↑ ↓ ▶ Run ■ C ≫ Code</p>
                                                            v 🖼
        In [18]: # Print classification results
                 print(f"Accuracy: (accuracy)")
                 print(f"Precision: {precision}")
                 print(f"Recall: {recall}")
                 print(f"F1 Score: {f1}")
                 print(f"ROC-AUC Score: {roc_auc}")
                 Accuracy: 0.9603020496224379
                 Precision: 0.9565217391384348
                 Recall: 0.9248407643312102
                 F1 Score: 0.9404145077720207
                 ROC-AUC Score: 0.9516536611215595
        In [19]: # Print Confusion Matrix
                 print("Confusion Matrix:")
                 print(cm)
                 Confusion Matrix:
                 [[2999 66]
                  [ 118 1452]]
        In [20]: # Print Classification Report
                 print("Classification Report:")
                 print(classification_rep)
                 Classification Report:
                              precision
                                          recall f1-score support
                            8
                                   0,96
                                             0.98
                                                       0.97
                                                                 3065
```

0.96

0.96

0.96

1

accuracy

macro avg

weighted avg

0.92

0.95

0.96

0.94

8.96

0.96

0.96

1570

4635

4635

4635

```
3.
  # Perform RandomizedSearchCV
  random search = RandomizedSearchCV(pipeline, param distributions=param dist, n iter=10, cv=5, scoring='accuracy', random state=45
  random search.fit(X, v)
  # Print the best hyperparameters
  best params = random search.best params
  print("Best Hyperparameters:")
  print(best params)
  Best Hyperparameters:
  {'classifier n estimators': 50, 'classifier min samples split': 5, 'classifier min samples leaf': 1, 'classifier max dept
  h': None, 'classifier bootstrap': False}
4.
 categorical transformer = Pipeline(steps=|
      ('imputer', SimpleImputer(strategy='most frequent')),
      ('onehot', OneHotEncoder(handle unknown='ignore'))
 1)
  preprocessor = ColumnTransformer(
      transformers=[
          ('num', numerical transformer, numerical features),
          ('cat', categorical transformer, categorical features)
      1)
  # Define the RandomForestClassifier pipeline
  pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                               ('classifier', RandomForestClassifier())])
  # Define the stratified k-fold cross-validation
  stratified kfold = StratifiedKFold(n splits=5, shuffle=True, random state=42)
  # Perform cross-validation
  cv results = cross_val_score(pipeline, X, y, cv=stratified_kfold, scoring='accuracy')
  # Print the cross-validation results
  print("Cross-Validation Results:")
  print("Mean Accuracy:", np.mean(cv results))
  print("Standard Deviation:", np.std(cv_results))
```

Cross-Validation Results: Mean Accuracy: 0.9502394151571402 Standard Deviation: 0.0014897697005680104