Customer-purchase-behaviour-prediction

September 4, 2024

[2]:
```
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score

# Load the dataset
file_path = '/content/customer_purchase_data.csv'
data = pd.read_csv(file_path)

# Separate features and target variable
X = data.drop('PurchaseStatus', axis=1)
y = data['PurchaseStatus']

# One-hot encode categorical variables
X = pd.get_dummies(X, columns=['Gender', 'ProductCategory', 'LoyaltyProgram'],␣
 ↪drop_first=True)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Define the RandomForest model
rf_model = RandomForestClassifier(random_state=42)

# Define parameter grid for GridSearchCV
rf_param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

```
# Set up GridSearchCV
rf_grid_search = GridSearchCV(estimator=rf_model, param_grid=rf_param_grid,␣
 ↪scoring='accuracy', cv=5, verbose=1)

# Fit GridSearchCV
rf_grid_search.fit(X_train, y_train)

# Get the best parameters and model
rf_best_params = rf_grid_search.best_params_
rf_best_model = rf_grid_search.best_estimator_

# Make predictions with the best model
rf_y_pred = rf_best_model.predict(X_test)

# Evaluate the model
rf_accuracy = accuracy_score(y_test, rf_y_pred)
rf_report = classification_report(y_test, rf_y_pred)

print(f"Random Forest Best Parameters: {rf_best_params}")
print(f"Accuracy: {rf_accuracy:.2f}")
print("Classification Report:")
print(rf_report)
```

```
Fitting 5 folds for each of 108 candidates, totalling 540 fits
Random Forest Best Parameters: {'max_depth': None, 'min_samples_leaf': 4,
'min_samples_split': 10, 'n_estimators': 300}
Accuracy: 0.95
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.98      0.96       172
           1       0.97      0.91      0.94       128

    accuracy                           0.95       300
   macro avg       0.95      0.94      0.95       300
weighted avg       0.95      0.95      0.95       300
```

[3]:
```
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from xgboost import XGBClassifier
from sklearn.metrics import classification_report, accuracy_score

# Load the dataset
file_path = '/content/customer_purchase_data.csv'
```

```
data = pd.read_csv(file_path)

# Separate features and target variable
X = data.drop('PurchaseStatus', axis=1)
y = data['PurchaseStatus']

# One-hot encode categorical variables
X = pd.get_dummies(X, columns=['Gender', 'ProductCategory', 'LoyaltyProgram'],␣
 ↪drop_first=True)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Define the XGBoost model
model = XGBClassifier(eval_metric='logloss')

# Define a more extensive parameter grid
param_grid = {
    'n_estimators': [100, 200, 300, 400],
    'max_depth': [3, 5, 7, 9],
    'learning_rate': [0.01, 0.05, 0.1, 0.2],
    'subsample': [0.8, 0.9, 1.0],
    'colsample_bytree': [0.8, 0.9, 1.0]
}

# Set up GridSearchCV with k-fold cross-validation
grid_search = GridSearchCV(estimator=model, param_grid=param_grid,␣
 ↪scoring='accuracy', cv=10, verbose=1)

# Fit GridSearchCV
grid_search.fit(X_train, y_train)

# Get the best parameters and model
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_

# Make predictions with the best model
y_pred = best_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
```

```
report = classification_report(y_test, y_pred)

print(f"Best Parameters: {best_params}")
print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:")
print(report)
```

```
Fitting 10 folds for each of 576 candidates, totalling 5760 fits
Best Parameters: {'colsample_bytree': 0.8, 'learning_rate': 0.05, 'max_depth':
3, 'n_estimators': 100, 'subsample': 0.8}
Accuracy: 0.95
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.98      0.96       172
           1       0.97      0.91      0.94       128

    accuracy                           0.95       300
   macro avg       0.96      0.95      0.95       300
weighted avg       0.95      0.95      0.95       300
```

[ ]: