In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, accuracy_score, roc_auc_score

# Load data
df = pd.read_csv("Ecommerce_Consumer_Behavior_Analysis_Data.csv")

# Inspect data
print(df.head())
print(df.info())
print(df.describe())

# Handle missing values
df = df.dropna()  # Simple approach, can be refined

# Define features and target (replace 'target_column' with actual column name)
target_column = "your_target_column_name_here"
X = df.drop(columns=[target_column])
y = df[target_column]

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

# Identify numerical and categorical features
num_features = X.select_dtypes(include=["int64", "float64"]).columns.tolist()
cat_features = X.select_dtypes(include=["object"]).columns.tolist()

# Preprocessing: Scaling & Encoding
preprocessor = ColumnTransformer([
    ("num", StandardScaler(), num_features),
    ("cat", OneHotEncoder(handle_unknown="ignore"), cat_features)
])

# Logistic Regression Baseline Model
log_reg_pipeline = Pipeline([
    ("preprocessor", preprocessor),
    ("classifier", LogisticRegression(max_iter=1000))
])
log_reg_pipeline.fit(X_train, y_train)
y_pred_log = log_reg_pipeline.predict(X_test)

def evaluate_model(name, y_true, y_pred):
    print(f"\n{name} Performance:")
    print(classification_report(y_true, y_pred))

evaluate_model("Logistic Regression", y_test, y_pred_log)

# Decision Tree with Hyperparameter Tuning
dt_pipeline = Pipeline([
    ("preprocessor", preprocessor),
```

```
    ("classifier", DecisionTreeClassifier(random_state=42))
])

param_grid = {
    "classifier__max_depth": [3, 5, 10],
    "classifier__min_samples_split": [2, 5, 10]
}

grid_search = GridSearchCV(dt_pipeline, param_grid, cv=5, scoring="accuracy")
grid_search.fit(X_train, y_train)

y_pred_tree = grid_search.best_estimator_.predict(X_test)
evaluate_model("Decision Tree (Tuned)", y_test, y_pred_tree)

# Feature Importance Analysis
best_tree = grid_search.best_estimator_.named_steps["classifier"]
importances = best_tree.feature_importances_
feature_names = preprocessor.get_feature_names_out()

sns.barplot(x=importances, y=feature_names)
plt.title("Feature Importance in Decision Tree")
plt.xlabel("Importance")
plt.ylabel("Feature")
plt.show()
```

```
   Customer_ID  Age  Gender Income_Level Marital_Status Education_Level  \
0  37-611-6911   22  Female       Middle        Married      Bachelor's
1  29-392-9296   49    Male         High        Married     High School
2  84-649-5117   24  Female       Middle         Single        Master's
3  48-980-6078   29  Female       Middle         Single        Master's
4  91-170-9072   33  Female       Middle        Widowed     High School

  Occupation  Location     Purchase_Category Purchase_Amount  ...  \
0     Middle      Évry  Gardening & Outdoors        $333.80   ...
1       High  Huocheng      Food & Beverages        $222.22   ...
2       High    Huzhen       Office Supplies        $426.22   ...
3     Middle    Wiwilí       Home Appliances        $101.31   ...
4     Middle      Nara             Furniture        $211.70   ...

   Customer_Satisfaction Engagement_with_Ads  Device_Used_for_Shopping  \
0                      7                 NaN                   Tablet
1                      5                High                   Tablet
2                      7                 Low               Smartphone
3                      1                 NaN               Smartphone
4                     10                 NaN               Smartphone

   Payment_Method  Time_of_Purchase Discount_Used  \
0     Credit Card          3/1/2024          True
1          PayPal         4/16/2024          True
2      Debit Card         3/15/2024          True
3           Other         10/4/2024          True
4      Debit Card         1/30/2024         False

  Customer_Loyalty_Program_Member  Purchase_Intent  Shipping_Preference  \
0                           False       Need-based        No Preference
1                           False      Wants-based             Standard
2                            True        Impulsive        No Preference
3                            True       Need-based              Express
4                           False      Wants-based        No Preference

  Time_to_Decision
0                2
1                6
2                3
3               10
4                4

[5 rows x 28 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 28 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   Customer_ID                           1000 non-null   object
 1   Age                                   1000 non-null   int64
 2   Gender                                1000 non-null   object
 3   Income_Level                          1000 non-null   object
 4   Marital_Status                        1000 non-null   object
 5   Education_Level                       1000 non-null   object
 6   Occupation                            1000 non-null   object
 7   Location                              1000 non-null   object
 8   Purchase_Category                     1000 non-null   object
 9   Purchase_Amount                       1000 non-null   object
 10  Frequency_of_Purchase                 1000 non-null   int64
 11  Purchase_Channel                      1000 non-null   object
```

```
 12  Brand_Loyalty                          1000 non-null   int64
 13  Product_Rating                         1000 non-null   int64
 14  Time_Spent_on_Product_Research(hours)  1000 non-null   float64
 15  Social_Media_Influence                 753 non-null    object
 16  Discount_Sensitivity                   1000 non-null   object
 17  Return_Rate                            1000 non-null   int64
 18  Customer_Satisfaction                  1000 non-null   int64
 19  Engagement_with_Ads                    744 non-null    object
 20  Device_Used_for_Shopping               1000 non-null   object
 21  Payment_Method                         1000 non-null   object
 22  Time_of_Purchase                       1000 non-null   object
 23  Discount_Used                          1000 non-null   bool
 24  Customer_Loyalty_Program_Member        1000 non-null   bool
 25  Purchase_Intent                        1000 non-null   object
 26  Shipping_Preference                    1000 non-null   object
 27  Time_to_Decision                       1000 non-null   int64
dtypes: bool(2), float64(1), int64(7), object(18)
memory usage: 205.2+ KB
None
                Age  Frequency_of_Purchase  Brand_Loyalty  Product_Rating  \
count  1000.000000            1000.000000    1000.000000     1000.000000
mean     34.304000               6.945000       3.026000        3.033000
std       9.353238               3.147361       1.416803        1.436654
min      18.000000               2.000000       1.000000        1.000000
25%      26.000000               4.000000       2.000000        2.000000
50%      34.500000               7.000000       3.000000        3.000000
75%      42.000000              10.000000       4.000000        4.000000
max      50.000000              12.000000       5.000000        5.000000

       Time_Spent_on_Product_Research(hours)  Return_Rate  \
count                            1000.000000  1000.000000
mean                                1.013030     0.954000
std                                 0.791802     0.810272
min                                 0.000000     0.000000
25%                                 0.000000     0.000000
50%                                 1.000000     1.000000
75%                                 2.000000     2.000000
max                                 2.000000     2.000000

       Customer_Satisfaction  Time_to_Decision
count            1000.000000       1000.000000
mean                5.399000          7.547000
std                 2.868454          4.035849
min                 1.000000          1.000000
25%                 3.000000          4.000000
50%                 5.000000          8.000000
75%                 8.000000         11.000000
max                10.000000         14.000000
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[1], line 26
     24 # Define features and target (replace 'target_column' with actual column
name)
     25 target_column = "your_target_column_name_here"
---> 26 X = df.drop(columns=[target_column])
     27 y = df[target_column]
     29 # Train-test split

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:5581, in DataFrame.drop(s
elf, labels, axis, index, columns, level, inplace, errors)
   5433 def drop(
   5434     self,
   5435     labels: IndexLabel | None = None,
   (...)
   5442     errors: IgnoreRaise = "raise",
   5443 ) -> DataFrame | None:
   5444     """
   5445     Drop specified labels from rows or columns.
   5446
   (...)
   5579             weight  1.0     0.8
   5580     """
-> 5581     return super().drop(
   5582         labels=labels,
   5583         axis=axis,
   5584         index=index,
   5585         columns=columns,
   5586         level=level,
   5587         inplace=inplace,
   5588         errors=errors,
   5589     )

File ~\anaconda3\Lib\site-packages\pandas\core\generic.py:4788, in NDFrame.drop(s
elf, labels, axis, index, columns, level, inplace, errors)
   4786 for axis, labels in axes.items():
   4787     if labels is not None:
-> 4788         obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   4790 if inplace:
   4791     self._update_inplace(obj)

File ~\anaconda3\Lib\site-packages\pandas\core\generic.py:4830, in NDFrame._drop_
axis(self, labels, axis, level, errors, only_slice)
   4828         new_axis = axis.drop(labels, level=level, errors=errors)
   4829     else:
-> 4830         new_axis = axis.drop(labels, errors=errors)
   4831     indexer = axis.get_indexer(new_axis)
   4833 # Case for non-unique axis
   4834 else:

File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:7070, in Index.dro
p(self, labels, errors)
   7068 if mask.any():
   7069     if errors != "ignore":
-> 7070         raise KeyError(f"{labels[mask].tolist()} not found in axis")
   7071     indexer = indexer[~mask]
   7072 return self.delete(indexer)

KeyError: "['your_target_column_name_here'] not found in axis"
```

In [ ]:
```
print(f"Dataset contains {df.shape[0]} rows and {df.shape[1]} columns.")
```

In [ ]:
```

```