

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import LabelEncoder
from mlxtend.frequent_patterns import apriori, association_rules
```

```
data = pd.read_csv('e-shop-clothing-2008.csv', delimiter=';')
```

```
data.head()
```

	year	month	day	order	country	session ID	page 1 (main category)	page 2 (clothing model)	colour	location
0	2008	4	1	1	29	1	1	A13	1	
1	2008	4	1	2	29	1	1	A16	1	
2	2008	4	1	3	29	1	2	B4	10	
3	2008	4	1	4	29	1	2	B17	6	
4	2008	4	1	5	29	1	2	B8	4	

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 165474 entries, 0 to 165473
Data columns (total 14 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   year                                165474 non-null  int64
1   month                              165474 non-null  int64
2   day                                165474 non-null  int64
3   order                              165474 non-null  int64
4   country                            165474 non-null  int64
5   session ID                         165474 non-null  int64
6   page 1 (main category)             165474 non-null  int64
7   page 2 (clothing model)            165474 non-null  object
8   colour                             165474 non-null  int64
9   location                           165474 non-null  int64
10  model photography                   165474 non-null  int64
11  price                              165474 non-null  int64
12  price 2                            165474 non-null  int64
13  page                               165474 non-null  int64
dtypes: int64(13), object(1)
memory usage: 17.7+ MB
```

```
data.describe()
```

	year	month	day	order	country	session ID
count	165474.0	165474.000000	165474.000000	165474.000000	165474.000000	165474.000000
mean	2008.0	5.585887	14.524554	9.817476	26.952621	12058.417056
std	0.0	1.328160	8.830374	13.478411	7.150691	7008.418903
min	2008.0	4.000000	1.000000	1.000000	1.000000	1.000000
25%	2008.0	4.000000	7.000000	2.000000	29.000000	5931.000000
50%	2008.0	5.000000	14.000000	6.000000	29.000000	11967.500000
75%	2008.0	7.000000	22.000000	12.000000	29.000000	18219.000000
max	2008.0	8.000000	31.000000	195.000000	47.000000	24026.000000

```
blouses_data = data[data['page 1 (main category)'] == 3]
```

```
selected_features = ['year', 'month', 'day', 'country', 'session ID', 'colour', 'location', 'model photography', 'price']
```

```
X = blouses_data[selected_features].copy()
y = blouses_data['page']
```

```
label_encoders = {}
for feature in ['country', 'colour', 'location']:
    label_encoders[feature] = LabelEncoder()
    X[feature] = label_encoders[feature].fit_transform(X[feature])
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Step 3: Model Training and Evaluation
model = LogisticRegression(max_iter=1000, solver='liblinear', penalty='l2', C=1.0)
# Adjust penalty and C parameters as needed
model.fit(X_train, y_train)
```

```
LogisticRegression
LogisticRegression(max_iter=1000, solver='liblinear')
```

```
y_pred = model.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.5533955417314671
Classification Report:
```

	precision	recall	f1-score	support
1	0.52	0.85	0.65	3367
2	0.46	0.12	0.19	1865
3	0.64	0.35	0.45	1779
4	0.73	0.78	0.76	705
accuracy			0.55	7716
macro avg	0.59	0.53	0.51	7716
weighted avg	0.55	0.55	0.50	7716

```
threshold = 0.5
likelihood_to_buy = model.predict_proba(X_test)[: , 1]
recommendations = likelihood_to_buy > threshold
```

Discussion

Although logistic regression offers feature coefficients, it may not be able to describe complex relationships. Predictive models rely on dataset attributes that may not fully capture all factors impacting a purchase decision. There may not be enough previous data available for new users or categories to make reliable forecasts. Over time, customer preferences may vary, and the model may not adjust to these changes as rapidly. If overfitting takes place, the model may work well on training data but may not generalize to new data.

```
import warnings
warnings.filterwarnings("ignore", message="`should_run_async` will not call `transform_cell` automatically")
warnings.filterwarnings("ignore", message="np.find_common_type is deprecated")
```

```
data = pd.read_csv('e-shop-clothing-2008.csv', delimiter=';')
```

```

transactions = data.pivot_table(index='session ID', columns='page 1 (main category)',
                                values='page 2 (clothing model)', aggfunc=lambda x: list(x)).reset_index().fillna(0)

# Convert the list of clothing models into a binary format
basket_sets_binary = basket_sets.applymap(lambda x: True if x != 0 else False)

# Mine frequent itemsets using Apriori
frequent_itemsets = apriori(basket_sets_binary, min_support=0.01, use_colnames=True)

# Step 3: Association Rule Generation
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
blouse_rules = rules[rules['consequents'].astype(str).str.contains("'category=3'")]

print(transactions.columns)

# Step 4: Evaluation of Missed Transactions
missed_transactions = transactions[(transactions[1] != 3) & (transactions['session ID'].isin(rules[rules['consequents']]['session ID'])
additional_revenue = missed_transactions[missed_transactions[1] == 3][4].sum()

print(data.head())

```

```
Index(['session ID', 1, 2, 3, 4], dtype='object', name='page 1 (main category)')
```

	year	month	day	order	country	session ID	page 1 (main category) \
0	2008	4	1	1	29	1	1
1	2008	4	1	2	29	1	1
2	2008	4	1	3	29	1	2
3	2008	4	1	4	29	1	2
4	2008	4	1	5	29	1	2

	page 2 (clothing model)	colour	location	model photography	price \
0	A13	1	5	1	28
1	A16	1	6	1	33
2	B4	10	2	1	52
3	B17	6	6	2	38
4	B8	4	3	2	52

	price 2	page
0	2	1
1	2	1
2	1	1
3	2	1
4	1	1