```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import classification_report, accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.impute import SimpleImputer
```

```
df = pd.read_csv('/content/ab_testing.csv')
df.head()
```

	User ID	Group	Page Views	Time Spent	Conversion	Device	Location
0	14292	B	3	424	No	Mobile	Northern Ireland
1	11682	A	9	342	No	Mobile	Scotland
2	19825	A	2	396	No	Desktop	Northern Ireland
3	16080	B	4	318	No	Desktop	Wales
4	18851	A	1	338	Yes	Desktop	Scotland

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.shape
```

```
(5000, 7)
```

```
# Preprocess the data
# 1. Convert 'Group' to numerical labels
le = LabelEncoder()
df['Group'] = le.fit_transform(df['Group'])
```

```
# 2. Convert 'Device' and 'Location' to numerical labels
df['Device'] = le.fit_transform(df['Device'])
df['Location'] = le.fit_transform(df['Location'])
```

```
# 3. Convert 'Conversion' to numerical (assuming 'Yes'/'No' values)
# Ensure this is executed *before* defining X and splitting the data
df['Conversion'] = df['Conversion'].map({'Yes': 1, 'No': 0})
```

```
# Define features (X) and target (y)
X = df[['Page Views', 'Time Spent', 'Conversion', 'Device', 'Location']]
y = df['Group']
```

```
# Split 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)
```

```
# Scale numerical features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
# Impute missing values using SimpleImputer before splitting data
imputer = SimpleImputer(strategy='mean') # or 'median', 'most_frequent'
X = imputer.fit_transform(X)
```

```
# Split 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)
```

```
# Scale numerical features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
# Initialize and train the MLP classifier
mlp = MLPClassifier(hidden_layer_sizes=(10, 5), max_iter=1000, random_state=42)
mlp.fit(X_train, y_train)
```

```
MLPClassifier
MLPClassifier(hidden_layer_sizes=(10, 5), max_iter=1000, random_state=42)
```

```
# Make predictions
y_pred = mlp.predict(X_test)

# Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.529
              precision    recall  f1-score   support

           0       0.50      0.79      0.62       478
           1       0.60      0.29      0.39       522

    accuracy                           0.53      1000
   macro avg       0.55      0.54      0.50      1000
weighted avg       0.56      0.53      0.50      1000
```