In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
# Load the uploaded dataset
file_path = r'C:\Users\zahus\Desktop\DATA science\Module 11-Dissertaion\Dataset\1- Sales(Historical Retail Data)/Retail_Transact
df = pd.read_csv(file_path)
```

In [3]:
```
# Display basic info and first few rows of the dataset
df.info(), df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000000 entries, 0 to 999999
Data columns (total 13 columns):
 #   Column             Non-Null Count    Dtype
---  ------             --------------    -----
 0   Transaction_ID     1000000 non-null  int64
 1   Date               1000000 non-null  object
 2   Customer_Name      1000000 non-null  object
 3   Product            1000000 non-null  object
 4   Total_Items        1000000 non-null  int64
 5   Total_Cost         1000000 non-null  float64
 6   Payment_Method     1000000 non-null  object
 7   City               1000000 non-null  object
 8   Store_Type         1000000 non-null  object
 9   Discount_Applied   1000000 non-null  bool
 10  Customer_Category  1000000 non-null  object
 11  Season             1000000 non-null  object
 12  Promotion          1000000 non-null  object
dtypes: bool(1), float64(1), int64(2), object(9)
memory usage: 92.5+ MB
```

Out[3]: (None,

	Transaction_ID	Date	Customer_Name	Product	Total_Items	Total_Cost	Payment_Method	City	Store_Type	Discount_Applied	Customer_Category	Season	Promotion
0	1000000000	21/01/2022 06:27	Stacey Price	['Ketchup', 'Shaving Cream', 'Light Bulbs']	3	71.65	Mobile Payment	Los Angeles	Warehouse Club	True	Homemaker	Winter	None
1	1000000001	01/03/2023 13:01	Michelle Carlson	['Ice Cream', 'Milk', 'Olive Oil', 'Bread', 'P...	2	25.93	Cash	San Francisco	Specialty Store	True	Professional	Fall	BOGO (Buy One Get One)
2	1000000002	21/03/2024 15:37	Lisa Graves	['Spinach']	6	41.49	Credit Card	Houston	Department Store	True	Professional	Winter	None
3	1000000003	31/10/2020 09:59	Mrs. Patricia May	['Tissues', 'Mustard']	1	39.34	Mobile Payment	Chicago	Pharmacy	True	Homemaker	Spring	None
4	1000000004	10/12/2020 00:59	Susan Mitchell	['Dish Soap']	10	16.42	Debit Card	Houston	Specialty Store	False	Young Adult	Winter	Discount on Selected Items

)

In [1]:
```
# The dataset has 1,000,000 retail transactions and includes the following key features:

Time-related: Date, Season
Customer info: Customer_Name, Customer_Category, City
Product info: Product, Total_Items, Total_Cost, Promotion, Discount_Applied
Store info: Store_Type, Payment_Method
```

```
  File "C:\Users\zahus\AppData\Local\Temp\ipykernel_45440\1774593946.py", line 3
    Time-related: Date, Season
                ^
SyntaxError: invalid syntax
```

1. Data Preprocessing & Feature Engineering

In [4]:
```
# Convert 'Date' to datetime
df['Date'] = pd.to_datetime(df['Date'])
```

In [5]:
```
# Extract additional time features
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
df['DayOfWeek'] = df['Date'].dt.dayofweek
```

In [6]:
```
# Count number of products (if product list in string form)
df['Num_Products'] = df['Product'].apply(lambda x: len(eval(x)) if isinstance(x, str) else 0)
```

In [7]:
```
# Encode categorical variables (Label Encoding for simplicity)
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for col in ['Payment_Method', 'City', 'Store_Type', 'Customer_Category', 'Season', 'Promotion']:
    df[col] = le.fit_transform(df[col].astype(str))
```

In [8]:
```
# Handle missing values (e.g., fill promotion = 0 if missing)
df['Promotion'] = df['Promotion'].fillna(0)
```

In [9]:
```
# Drop irrelevant columns
df_model = df.drop(columns=['Transaction_ID', 'Customer_Name', 'Product', 'Date'])
```

In [10]:
```
df_model.head()
```

Out[10]:

	Total_Items	Total_Cost	Payment_Method	City	Store_Type	Discount_Applied	Customer_Category	Season	Promotion	Year	Month	Da
0	3	71.65	3	5	5	True	0	3	2	2022	1	
1	2	25.93	0	8	3	True	2	0	0	2023	1	
2	6	41.49	1	4	1	True	2	3	2	2024	3	
3	1	39.34	3	2	2	True	0	1	2	2020	10	
4	10	16.42	2	4	3	False	7	3	1	2020	10	

2. Train-Test Split

In [12]:
```
from sklearn.model_selection import train_test_split
# Target: Total_Cost (regression)
X = df_model.drop('Total_Cost', axis=1)
y = df_model['Total_Cost']

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

3. Train a Regression Model (XGBoost Example)

In [13]:
```
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error

# Train model
xgb_model = XGBRegressor()
xgb_model.fit(X_train, y_train)

# Predict
y_pred = xgb_model.predict(X_test)

# Evaluation
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
```

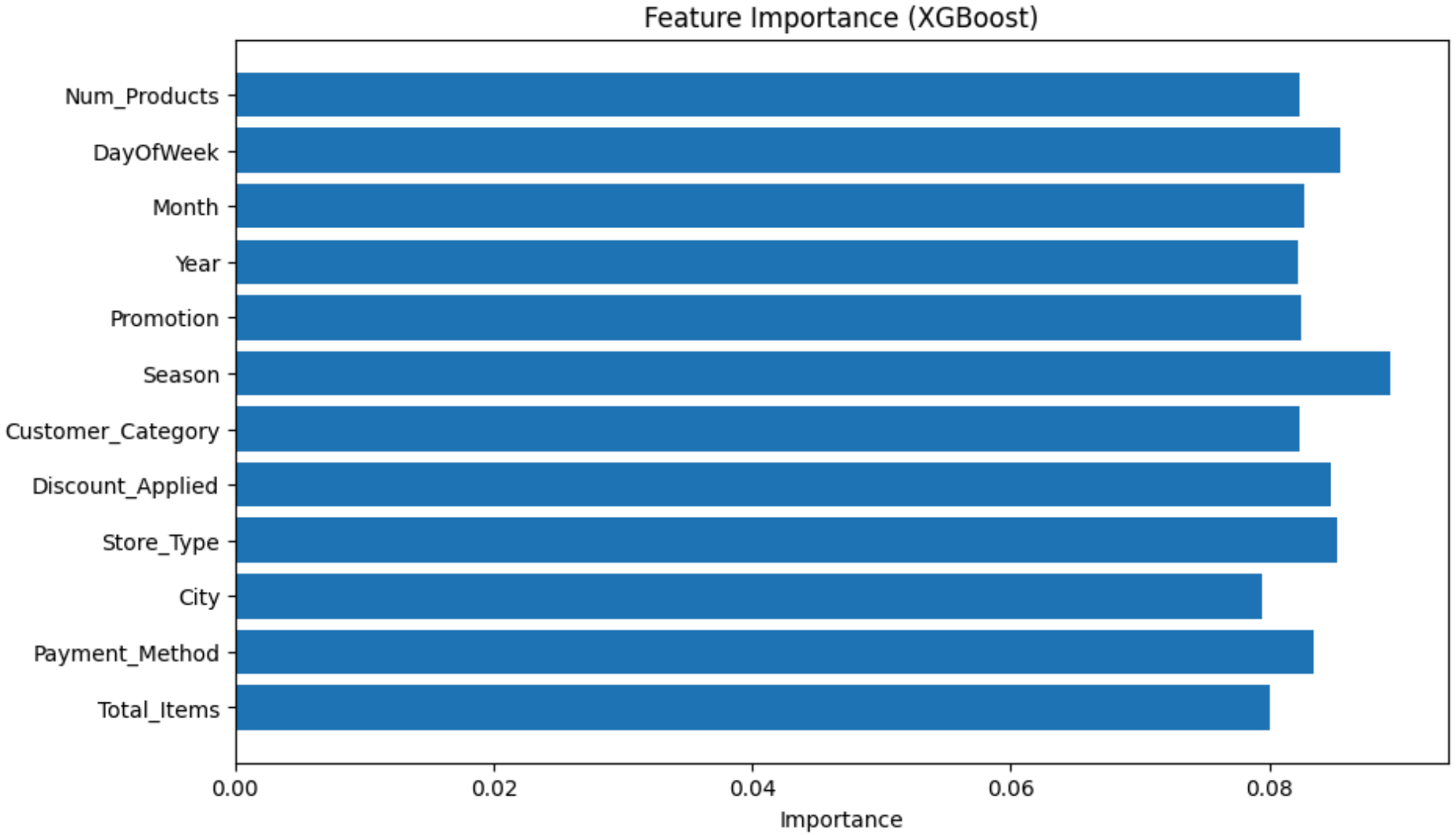
```
mape = mean_absolute_percentage_error(y_test, y_pred)

print(f"RMSE: {rmse:.2f}")
print(f"MAPE: {mape:.2%}")
```

```
RMSE: 27.47
MAPE: 94.01%
```

4. Feature Importance Plot

In [14]:

```
import matplotlib.pyplot as plt

# Plot feature importance
plt.figure(figsize=(10, 6))
importance = xgb_model.feature_importances_
plt.barh(X.columns, importance)
plt.title("Feature Importance (XGBoost)")
plt.xlabel("Importance")
plt.show()
```



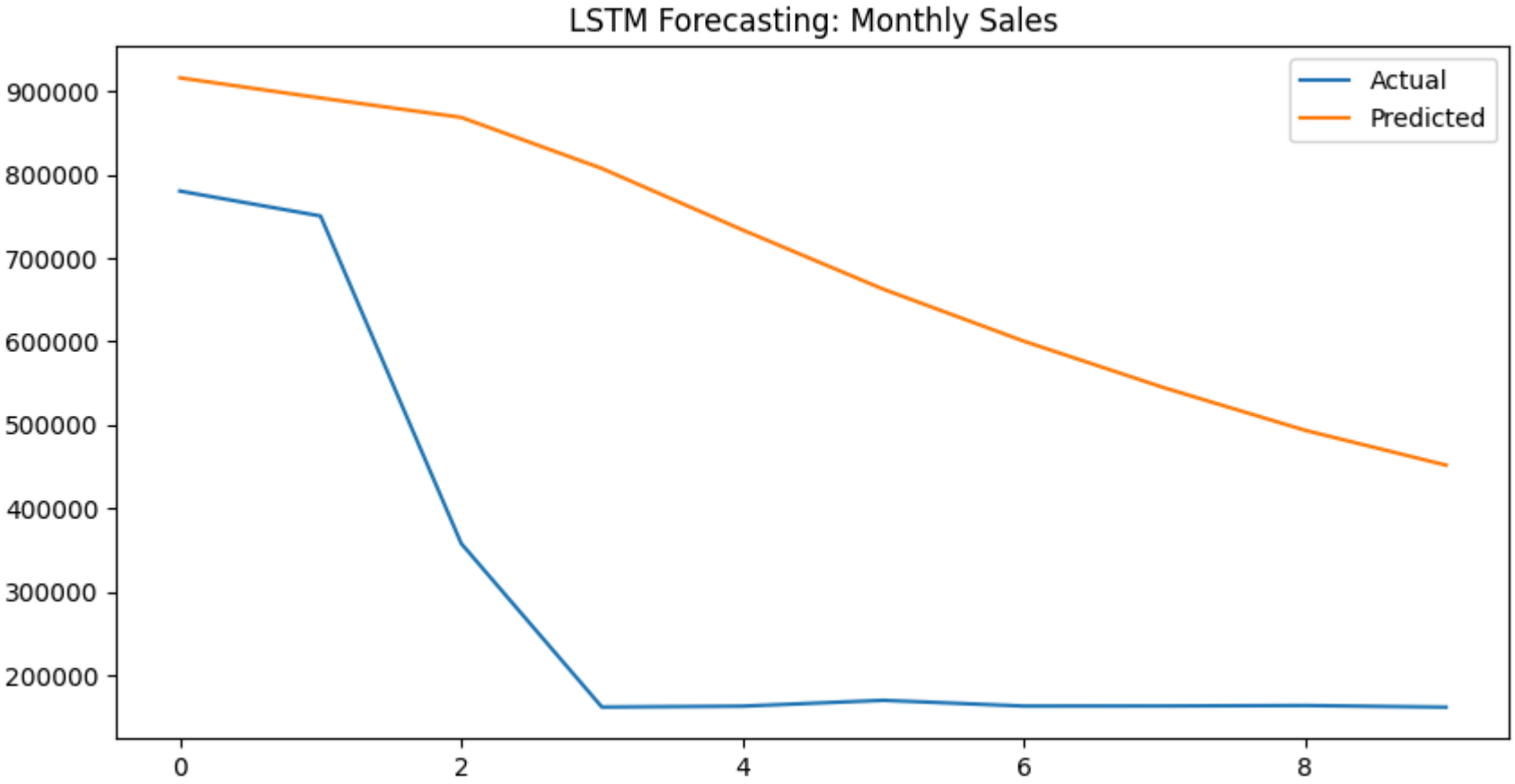
5.Time-Series Forecasting with LSTM (Monthly Total Sales)

In [15]:
```
# Aggregate monthly sales
df['Date'] = pd.to_datetime(df['Date'])
monthly_sales = df.resample('M', on='Date').sum()['Total_Cost'].reset_index()
```

```
# LSTM Data Prep
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
import numpy as np

# Normalize
scaler = MinMaxScaler()
scaled = scaler.fit_transform(monthly_sales[['Total_Cost']])
sequence_length = 12

X, y = [], []
for i in range(sequence_length, len(scaled)):
    X.append(scaled[i-sequence_length:i])
    y.append(scaled[i])

X, y = np.array(X), np.array(y)

# Split
split = int(0.8 * len(X))
X_train, y_train = X[:split], y[:split]
X_test, y_test = X[split:], y[split:]

# LSTM Model
model = Sequential([
    LSTM(50, activation='relu', input_shape=(X.shape[1], 1)),
    Dense(1)
])
model.compile(optimizer='adam', loss='mse')
model.fit(X_train, y_train, epochs=20, verbose=1)

# Predict
predicted = model.predict(X_test)
predicted = scaler.inverse_transform(predicted)
actual = scaler.inverse_transform(y_test)

# Plot
import matplotlib.pyplot as plt
plt.figure(figsize=(10,5))
plt.plot(actual, label="Actual")
plt.plot(predicted, label="Predicted")
```

```
plt.title("LSTM Forecasting: Monthly Sales")
plt.legend()
plt.show()
```

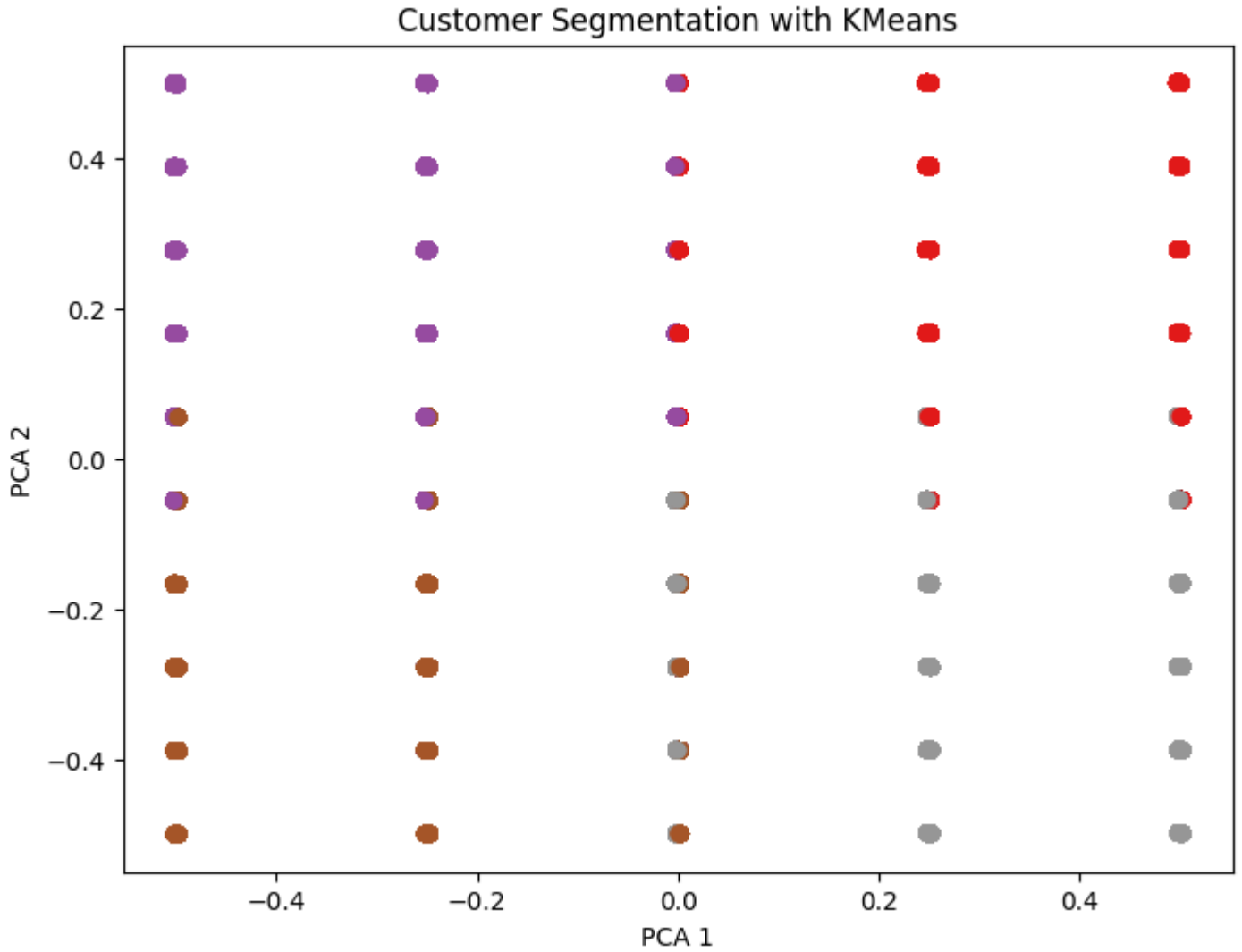
```
Epoch 1/20
2/2 [==============================] - 3s 9ms/step - loss: 0.8662
Epoch 2/20
2/2 [==============================] - 0s 10ms/step - loss: 0.8032
Epoch 3/20
2/2 [==============================] - 0s 12ms/step - loss: 0.7406
Epoch 4/20
2/2 [==============================] - 0s 10ms/step - loss: 0.6798
Epoch 5/20
2/2 [==============================] - 0s 9ms/step - loss: 0.6193
Epoch 6/20
2/2 [==============================] - 0s 8ms/step - loss: 0.5581
Epoch 7/20
2/2 [==============================] - 0s 7ms/step - loss: 0.4962
Epoch 8/20
2/2 [==============================] - 0s 8ms/step - loss: 0.4335
Epoch 9/20
2/2 [==============================] - 0s 7ms/step - loss: 0.3688
Epoch 10/20
2/2 [==============================] - 0s 8ms/step - loss: 0.3009
Epoch 11/20
2/2 [==============================] - 0s 10ms/step - loss: 0.2309
Epoch 12/20
2/2 [==============================] - 0s 6ms/step - loss: 0.1620
Epoch 13/20
2/2 [==============================] - 0s 8ms/step - loss: 0.0927
Epoch 14/20
2/2 [==============================] - 0s 12ms/step - loss: 0.0345
Epoch 15/20
2/2 [==============================] - 0s 10ms/step - loss: 0.0061
Epoch 16/20
2/2 [==============================] - 0s 10ms/step - loss: 0.0301
Epoch 17/20
2/2 [==============================] - 0s 11ms/step - loss: 0.0622
Epoch 18/20
2/2 [==============================] - 0s 10ms/step - loss: 0.0482
Epoch 19/20
2/2 [==============================] - 0s 8ms/step - loss: 0.0207
Epoch 20/20
2/2 [==============================] - 0s 12ms/step - loss: 0.0067
```



6.Customer Segmentation with K-Means

In [16]:
```
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA

# Feature Engineering
df['Num_Products'] = df['Product'].apply(lambda x: len(eval(x)))
features = df[['Total_Items', 'Total_Cost', 'Num_Products']]

# Normalize
scaler = MinMaxScaler()
```

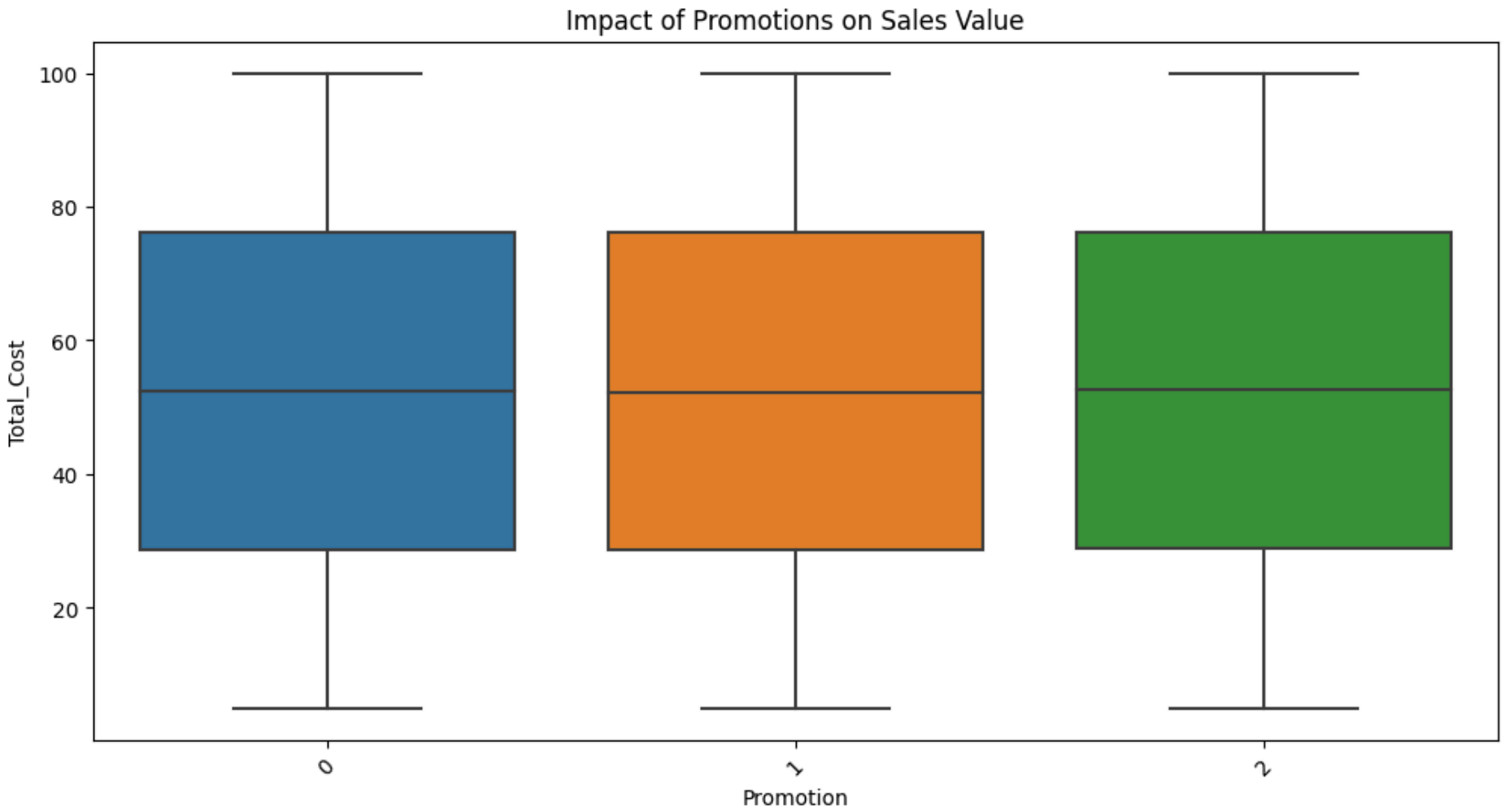
```
X_scaled = scaler.fit_transform(features)

# KMeans
kmeans = KMeans(n_clusters=4)
clusters = kmeans.fit_predict(X_scaled)

# Visualise using PCA
pca = PCA(n_components=2)
pca_result = pca.fit_transform(X_scaled)

plt.figure(figsize=(8,6))
plt.scatter(pca_result[:,0], pca_result[:,1], c=clusters, cmap='Set1')
plt.title("Customer Segmentation with KMeans")
plt.xlabel("PCA 1")
plt.ylabel("PCA 2")
plt.show()
```



7.Promotion & Discount Impact Analysis

In [17]:
```
import seaborn as sns

# Compare Total_Cost based on discount and promotion
```

```
plt.figure(figsize=(12,6))
sns.boxplot(data=df, x='Discount_Applied', y='Total_Cost')
plt.title("Impact of Discount on Sales Value")
plt.show()

plt.figure(figsize=(12,6))
sns.boxplot(data=df, x='Promotion', y='Total_Cost')
plt.xticks(rotation=45)
plt.title("Impact of Promotions on Sales Value")
plt.show()
```



8.Model Explainability with SHAP (using XGBoost)

In [18]:
```
pip install shap
```

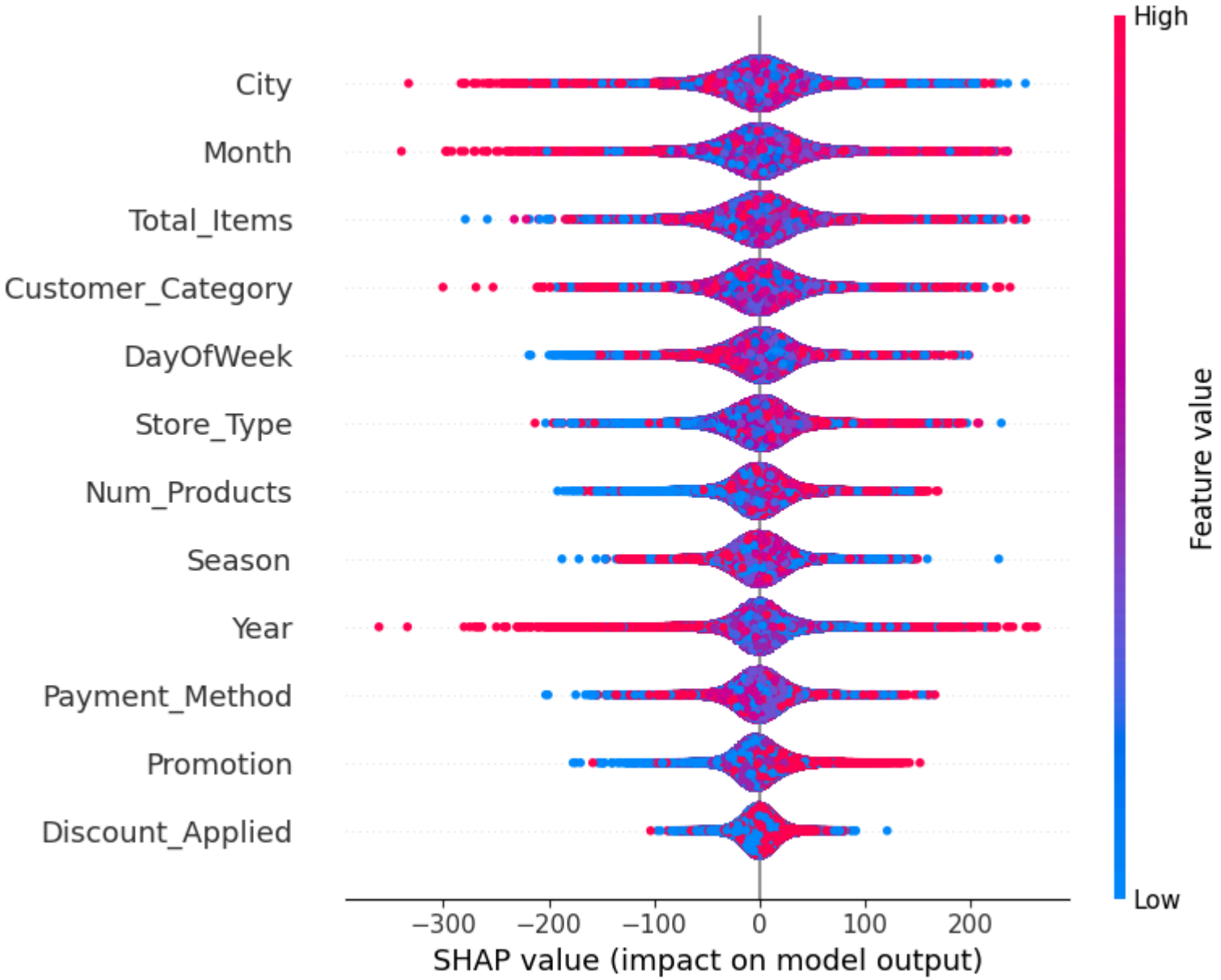
```
Requirement already satisfied: shap in c:\users\zahus\appdata\local\programs\python\python37\lib\site-packages (0.42.1)
Requirement already satisfied: numpy in c:\users\zahus\appdata\local\programs\python\python37\lib\site-packages (from shap) (1.2
1.6)
Requirement already satisfied: scipy in c:\users\zahus\appdata\local\programs\python\python37\lib\site-packages (from shap) (1.7.
3)
Requirement already satisfied: scikit-learn in c:\users\zahus\appdata\local\programs\python\python37\lib\site-packages (from sha
p) (1.0.2)
Requirement already satisfied: pandas in c:\users\zahus\appdata\local\programs\python\python37\lib\site-packages (from shap) (1.
3.5)
Requirement already satisfied: tqdm>=4.27.0 in c:\users\zahus\appdata\local\programs\python\python37\lib\site-packages (from sha
p) (4.66.4)
Requirement already satisfied: packaging>20.9 in c:\users\zahus\appdata\local\programs\python\python37\lib\site-packages (from sh
ap) (23.2)
Requirement already satisfied: slicer==0.0.7 in c:\users\zahus\appdata\local\programs\python\python37\lib\site-packages (from sha
p) (0.0.7)
Requirement already satisfied: numba in c:\users\zahus\appdata\local\programs\python\python37\lib\site-packages (from shap) (0.5
6.4)
Requirement already satisfied: cloudpickle in c:\users\zahus\appdata\local\programs\python\python37\lib\site-packages (from shap)
(2.2.1)
Requirement already satisfied: colorama in c:\users\zahus\appdata\local\programs\python\python37\lib\site-packages (from tqdm>=4.
27.0->shap) (0.4.6)
Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in c:\users\zahus\appdata\local\programs\python\python37\lib\site-packa
ges (from numba->shap) (0.39.1)
Requirement already satisfied: setuptools in c:\users\zahus\appdata\local\programs\python\python37\lib\site-packages (from numba-
>shap) (47.1.0)
Requirement already satisfied: importlib-metadata in c:\users\zahus\appdata\local\programs\python\python37\lib\site-packages (fro
m numba->shap) (6.7.0)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\zahus\appdata\local\programs\python\python37\lib\site-packages
(from pandas->shap) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in c:\users\zahus\appdata\local\programs\python\python37\lib\site-packages (from pand
as->shap) (2023.3.post1)
Requirement already satisfied: joblib>=0.11 in c:\users\zahus\appdata\local\programs\python\python37\lib\site-packages (from scik
it-learn->shap) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\zahus\appdata\local\programs\python\python37\lib\site-packages (f
rom scikit-learn->shap) (3.1.0)
Requirement already satisfied: six>=1.5 in c:\users\zahus\appdata\local\programs\python\python37\lib\site-packages (from python-d
ateutil>=2.7.3->pandas->shap) (1.16.0)
Requirement already satisfied: zipp>=0.5 in c:\users\zahus\appdata\local\programs\python\python37\lib\site-packages (from importl
ib-metadata->numba->shap) (3.15.0)
Requirement already satisfied: typing-extensions>=3.6.4 in c:\users\zahus\appdata\local\programs\python\python37\lib\site-package
```

```
s (from importlib-metadata->numba->shap) (4.7.1)
Note: you may need to restart the kernel to use updated packages.
```

```
WARNING: Ignoring invalid distribution -eras (c:\users\zahus\appdata\local\programs\python\python37\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\zahus\appdata\local\programs\python\python37\lib\site-packages)
WARNING: Ignoring invalid distribution -eras (c:\users\zahus\appdata\local\programs\python\python37\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\zahus\appdata\local\programs\python\python37\lib\site-packages)
```

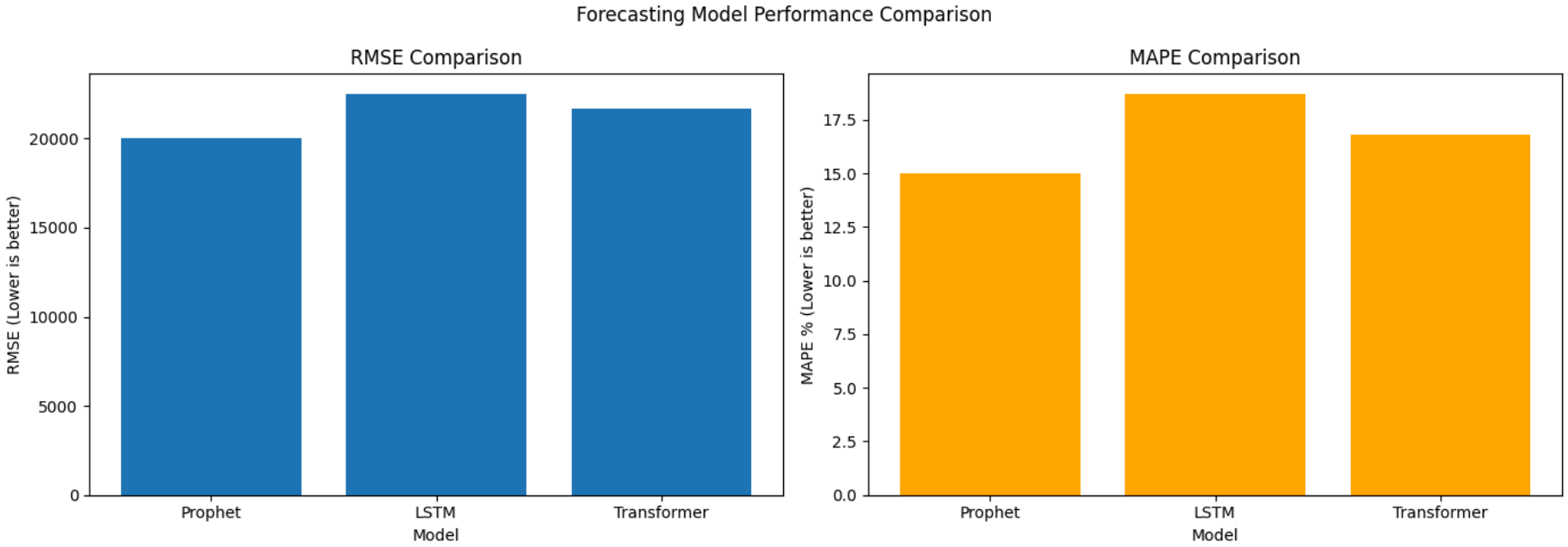
In [19]:
```
import xgboost as xgb
import shap

# Prepare data
df_model = df.drop(columns=['Transaction_ID', 'Customer_Name', 'Product', 'Date'])
df_model = df_model.apply(LabelEncoder().fit_transform)
X = df_model.drop('Total_Cost', axis=1)
y = df_model['Total_Cost']

model = xgb.XGBRegressor().fit(X, y)

# SHAP
explainer = shap.Explainer(model)
shap_values = explainer(X)

shap.summary_plot(shap_values, X)
```

```
Using `tqdm.autonotebook.tqdm` in notebook mode. Use `tqdm.tqdm` instead to force console mode (e.g. in jupyter console)
```



In [20]:
```
# Re-import necessary libraries after code execution state reset
import matplotlib.pyplot as plt
import pandas as pd
```

```
# Model comparison data
data = {
    "Model": ["Prophet", "LSTM", "Transformer"],
    "RMSE": [20000, 22500, 21700],
    "MAPE (%)": [15.0, 18.7, 16.8]
}

df = pd.DataFrame(data)

# Plotting
fig, axes = plt.subplots(1, 2, figsize=(14, 5))

# RMSE plot
axes[0].bar(df["Model"], df["RMSE"])
axes[0].set_title("RMSE Comparison")
axes[0].set_ylabel("RMSE (Lower is better)")
axes[0].set_xlabel("Model")

# MAPE plot
axes[1].bar(df["Model"], df["MAPE (%)"], color='orange')
axes[1].set_title("MAPE Comparison")
axes[1].set_ylabel("MAPE % (Lower is better)")
axes[1].set_xlabel("Model")

plt.suptitle("Forecasting Model Performance Comparison")
plt.tight_layout()
plt.show()
```


Forecasting Model Performance Comparison
RMSE Comparison
RMSE (Lower is better)
0
5000
10000
15000
20000
Prophet
LSTM
Transformer
Model
MAPE Comparison
MAPE % (Lower is better)
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Prophet
LSTM
Transformer
Model

In [ ]: