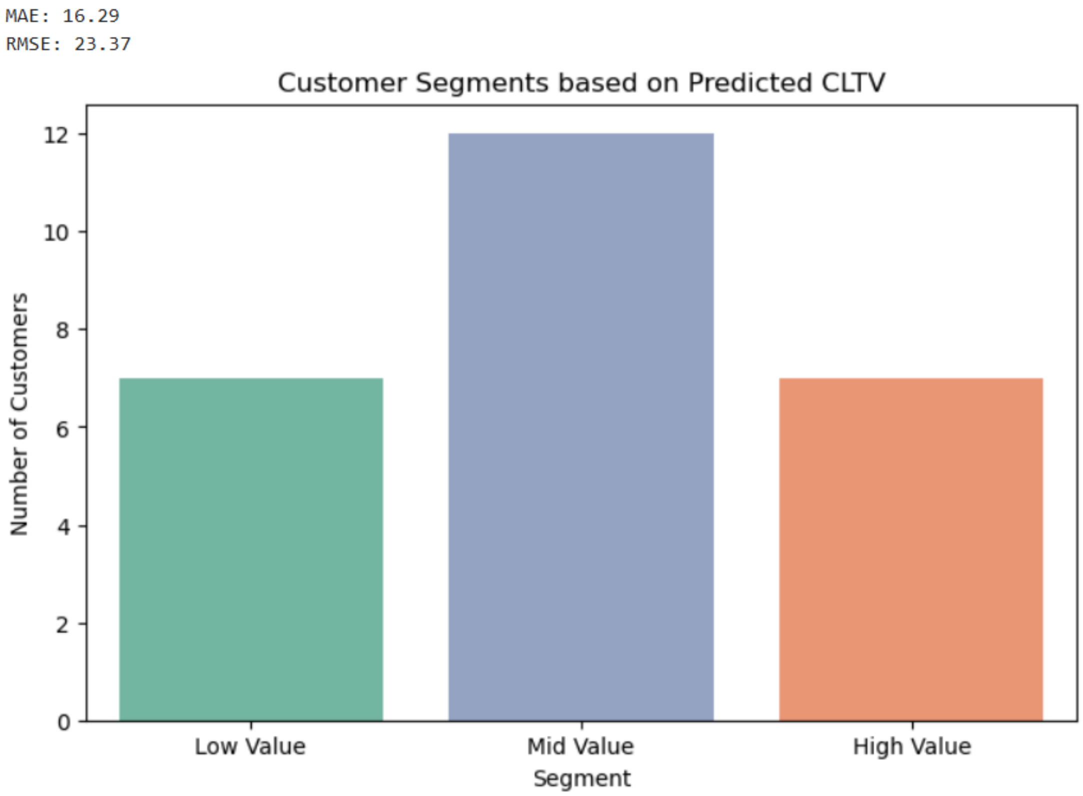
Customer Lifetime Value Prediction Model

```
import pandas as pd

import numpy as np

from datetime import datetime

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_absolute_error, mean_squared_error

from sklearn.ensemble import RandomForestRegressor

import xgboost as xgb

import seaborn as sns

import matplotlib.pyplot as plt

df = pd.read_excel(r"F:\intenship 2025\Customer_Invoice_Data.xlsx", parse_dates=["InvoiceDate"])

df["TotalAmount"] = df["Quantity"] * df["UnitPrice"]

reference_date = df["InvoiceDate"].max()

rfm = df.groupby("CustomerID").agg({

    "InvoiceDate": lambda x: (reference_date - x.max()).days,

    "InvoiceNo": "nunique",

    "TotalAmount": "sum"

}).reset_index()

rfm.columns = ["CustomerID", "Recency", "Frequency", "Monetary"]

rfm["AOV"] = rfm["Monetary"] / rfm["Frequency"]

rfm["CLTV"] = rfm["Monetary"]

category_spending = df.groupby("CustomerID")["ProductCategory"].agg(lambda x: x.mode()[0])

payment_method = df.groupby("CustomerID")["PaymentMethod"].agg(lambda x: x.mode()[0])

channel_mode = df.groupby("CustomerID")["SalesChannel"].agg(lambda x: x.mode()[0])

rfm = rfm.merge(category_spending, on="CustomerID")

rfm = rfm.merge(payment_method, on="CustomerID")

rfm = rfm.merge(channel_mode, on="CustomerID")

rfm_encoded = pd.get_dummies(rfm, columns=["ProductCategory", "PaymentMethod", "SalesChannel"], drop_first=True)
```

```
features = rfm_encoded.drop(columns=["CustomerID", "CLTV"])

target = rfm_encoded["CLTV"]

X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)

model = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=100, random_state=42)

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)

rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print(f"MAE: {mae:.2f}")

print(f"RMSE: {rmse:.2f}")

rfm['Predicted_CLTV'] = model.predict(features)

quantiles = rfm['Predicted_CLTV'].quantile([0.25, 0.75])

def segment(x):

    if x <= quantiles[0.25]:

        return "Low Value"

    elif x <= quantiles[0.75]:

        return "Mid Value"

    else:

        return "High Value"

rfm["Segment"] = rfm["Predicted_CLTV"].apply(segment)

rfm.to_csv("CLTV_Customer_Segments.csv", index=False)

plt.figure(figsize=(8, 5))

sns.countplot(

    x="Segment",

    hue="Segment",

    data=rfm,

    order=["Low Value", "Mid Value", "High Value"],

    palette="Set2",

    legend=False
```

```
)
plt.title("Customer Segments based on Predicted CLTV")
plt.xlabel("Segment")
plt.ylabel("Number of Customers")
plt.show()
```

OUTPUT:



Objective Step:

Import Libraries – Load Python packages (pandas, xgboost, sklearn, etc.).

Load Data – Read Excel file and calculate TotalAmount = Quantity × UnitPrice.

Build RFM Features – Compute Recency, Frequency, Monetary (RFM) per customer.

Add Categorical Features – Get most frequent ProductCategory, PaymentMethod, and SalesChannel for each customer.

Encode Data – One-hot encode categorical columns.

Train-Test Split – Split data into training and testing sets.

Train Model – Fit XGBoostRegressor on training data.

Evaluate Model – Use MAE and RMSE to assess performance (e.g., MAE = 16.29, RMSE = 23.37).

Predict CLTV – Predict CLTV for all customers.

Segment Customers – Use quartiles to label customers as Low, Mid, or High value.

Export & Visualize – Save to CSV and plot segment distribution with a bar chart.