Customer Retention

June 7, 2024

[1]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

1. Loading and Checking Data

[2]:
```python
#Load Customers dataset
customers_df=pd.read_csv(r"C:\Users\USER\Documents\Data Portfolio␣
↪Projects\Retail\Customer Retention\Datasets\Customers.csv")
customers_df.head()
```

[2]:
```
   CustomerID PurchaseHistory LastPurchaseDate  TotalSpend LoyaltyProgram  \
0        1001      Infrequent       2023-07-29     2590.17         Member
1        1002        One-time       2022-08-29     3509.48     Non-Member
2        1003      Infrequent       2023-07-18     4305.96         Member
3        1004      Infrequent       2022-04-06     1697.20         Member
4        1005        Frequent       2023-04-30     1179.18         Member

   FeedbackScore  EmailOpenRate  ClickThroughRate  WebsiteVisits  \
0              2           0.82              0.25             15
1              3           0.71              0.45             25
2              4           0.08              0.13             37
3              5           0.08              0.95             28
4              5           0.99              0.61             10

   CustomerServiceInteractions  Churn
0                            2      0
1                            0      0
2                            5      0
3                            1      0
4                            7      0
```

[3]:
```python
#Load Products dataset
products_df=pd.read_csv(r"C:\Users\USER\Documents\Data Portfolio␣
↪Projects\Retail\Customer Retention\Datasets\Products.csv")
products_df.head()
```

[3]:

	ProductID	ProductName	Category	Price
0	1	Product_1	Category2	81.50
1	2	Product_2	Category3	98.93
2	3	Product_3	Category1	62.30
3	4	Product_4	Category1	81.65
4	5	Product_5	Category1	96.45

[4]:
```
#Load Engagements dataset
engagements_df=pd.read_csv(r"C:\Users\USER\Documents\Data Portfolio␣
↪Projects\Retail\Customer Retention\Datasets\Engagements.csv")
engagements_df.head()
```

[4]:

	CustomerID	EngagementType	EngagementDate	EngagementOutcome
0	1002	Email	2023-06-08	Clicked
1	1067	Email	2022-10-22	Clicked
2	1087	Website	2022-08-12	Clicked
3	1012	Email	2023-06-09	Clicked
4	1020	Email	2022-02-15	Purchased

[5]:
```
#Load Loyalty dataset
loyalty_df=pd.read_csv(r"C:\Users\USER\Documents\Data Portfolio␣
↪Projects\Retail\Customer Retention\Datasets\LoyaltyProgram.csv")
loyalty_df.head()
```

[5]:

	CustomerID	JoinDate	PointsEarned
0	1001	2023-12-31	3812
1	1003	2022-05-01	1467
2	1004	2019-11-07	8289
3	1005	2022-02-16	9113
4	1006	2022-05-26	554

[8]:
```
#Check type of data we have
print(customers_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 11 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              100 non-null    int64
 1   PurchaseHistory         100 non-null    object
 2   LastPurchaseDate        100 non-null    object
 3   TotalSpend              100 non-null    float64
 4   LoyaltyProgram          100 non-null    object
 5   FeedbackScore           100 non-null    int64
 6   EmailOpenRate           100 non-null    float64
 7   ClickThroughRate        100 non-null    float64
 8   WebsiteVisits           100 non-null    int64
```

```
 9   CustomerServiceInteractions  100 non-null    int64
 10  Churn                        100 non-null    int64
dtypes: float64(3), int64(5), object(3)
memory usage: 8.7+ KB
None
```

[10]:

```
#Check stats of the data
customers_df.describe()
```

[10]:

```
        CustomerID   TotalSpend  FeedbackScore  EmailOpenRate  \
count   100.000000   100.000000     100.000000     100.000000
mean   1050.500000  2801.033000       3.040000       0.571300
std      29.011492  1361.460592       1.483376       0.318832
min    1001.000000   188.570000       1.000000       0.010000
25%    1025.750000  1606.605000       2.000000       0.340000
50%    1050.500000  2695.040000       3.000000       0.585000
75%    1075.250000  4101.115000       4.000000       0.862500
max    1100.000000  4981.640000       5.000000       1.000000

       ClickThroughRate  WebsiteVisits  CustomerServiceInteractions  \
count        100.000000     100.000000                   100.000000
mean           0.448100      24.870000                     4.340000
std            0.274103      14.561368                     2.850554
min            0.010000       1.000000                     0.000000
25%            0.187500      11.000000                     2.000000
50%            0.455000      25.000000                     4.000000
75%            0.647500      37.000000                     7.000000
max            0.980000      49.000000                     9.000000

            Churn
count  100.000000
mean     0.190000
std      0.394277
min      0.000000
25%      0.000000
50%      0.000000
75%      0.000000
max      1.000000
```

[11]:

```
# Check for missing values
missing_values = customers_df.isnull().sum()
print("Missing values in each column:\n", missing_values)
```

```
Missing values in each column:
 CustomerID                     0
PurchaseHistory                0
LastPurchaseDate               0
TotalSpend                     0
```

```
LoyaltyProgram                 0
FeedbackScore                  0
EmailOpenRate                  0
ClickThroughRate               0
WebsiteVisits                  0
CustomerServiceInteractions    0
Churn                          0
dtype: int64
```

[ ]:

```

```

2. Customer Segmentation

Questions: 1. What are the key characteristics used to segment customers currently?

2. Are there any existing customer personas or profiles?

3. How frequently should customer segments be updated?

[13]:

```
# Segmenting customers based on PurchaseHistory, TotalSpend, and LoyaltyProgram
customer_segments = customers_df.groupby(['PurchaseHistory', 'LoyaltyProgram']).
 ↪agg({
    'TotalSpend': 'mean',
    'CustomerID': 'count'
}).rename(columns={'CustomerID': 'CustomerCount'}).reset_index()
```

[14]:

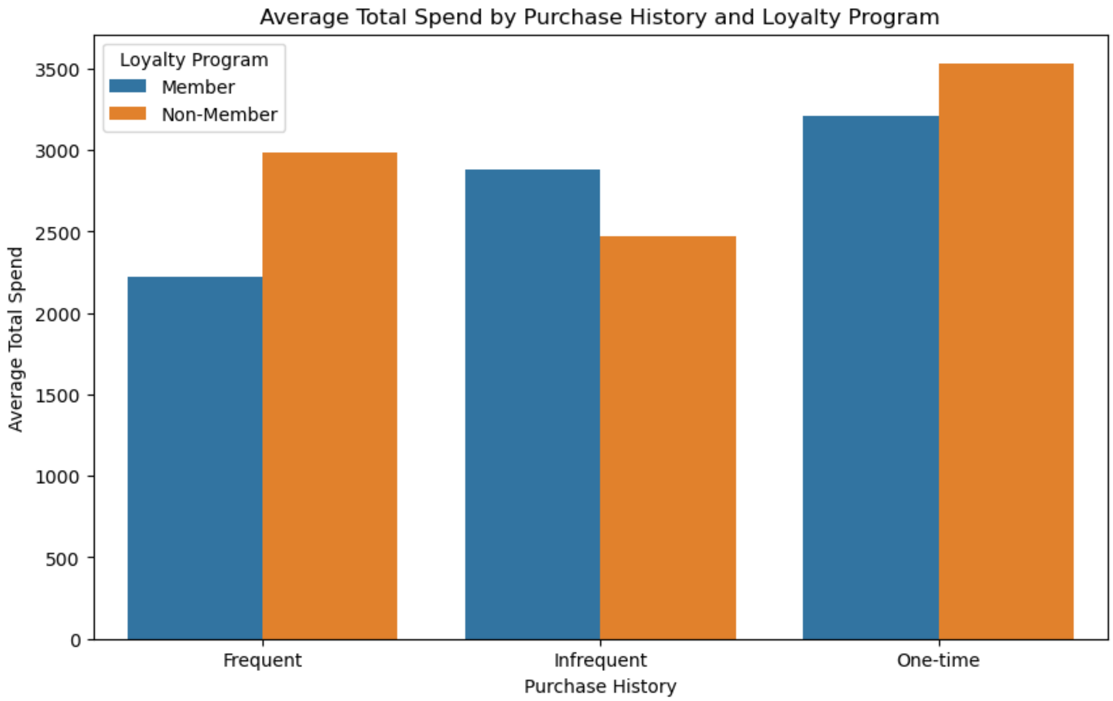
```
customer_segments
```

[14]:

	PurchaseHistory	LoyaltyProgram	TotalSpend	CustomerCount
0	Frequent	Member	2225.539375	16
1	Frequent	Non-Member	2985.458889	18
2	Infrequent	Member	2881.757742	31
3	Infrequent	Non-Member	2471.741765	17
4	One-time	Member	3210.364615	13
5	One-time	Non-Member	3533.514000	5

[15]:

```
# Visualization of the segmentation
plt.figure(figsize=(10, 6))
sns.barplot(data=customer_segments, x='PurchaseHistory', y='TotalSpend',␣
 ↪hue='LoyaltyProgram')
plt.title('Average Total Spend by Purchase History and Loyalty Program')
plt.xlabel('Purchase History')
plt.ylabel('Average Total Spend')
plt.legend(title='Loyalty Program')
plt.show()
```



[ ]:

3. Churn Prediction Model

Questions: 1. What historical data is available for developing the churn prediction model?

2. Are there specific behaviors or events that have been associated with customer churn in the past?

3. What machine learning tools or platforms are preferred or currently in use?

A churn prediction model will be built using logistic regression to identify at-risk customers.

[17]:
```
#Import relevant Libraries
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
```

[18]:
```
# Prepare data for churn prediction model
features = ['TotalSpend', 'FeedbackScore', 'EmailOpenRate', 'ClickThroughRate',␣
 ↪'WebsiteVisits', 'CustomerServiceInteractions']
X = customers_df[features]
y = customers_df['Churn']
```

[19]:
```
# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=42)
```

[20]:
```
# Building the model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
```

[20]:
```
LogisticRegression(max_iter=1000)
```

[21]:
```
# Making predictions
y_pred = model.predict(X_test)
```

[22]:
```
# Evaluating the model
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      1.00      0.91        25
           1       0.00      0.00      0.00         5

    accuracy                           0.83        30
   macro avg       0.42      0.50      0.45        30
weighted avg       0.69      0.83      0.76        30

[[25  0]
 [ 5  0]]
```

```
C:\Users\USER\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\USER\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\USER\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Result: Classification report and confusion matrix for the churn prediction model, showing precision, recall, and accuracy.

[ ]:
```
```

4. Personalized Marketing

Questions: 1. What channels (email, SMS, in-app notifications) are used for marketing communications?

2. How personalized are the current marketing efforts?

3. What type of product recommendations have been successful in the past?

[ ]:

```

```

Engagement data is analysed to see which channels are most effective. We also examine the success of product recommendations.

[23]:

```
# Analyzing engagement data
engagement_summary = engagements_df.groupby(['EngagementType',␣
 ↪'EngagementOutcome']).size().unstack(fill_value=0)
```

[24]:

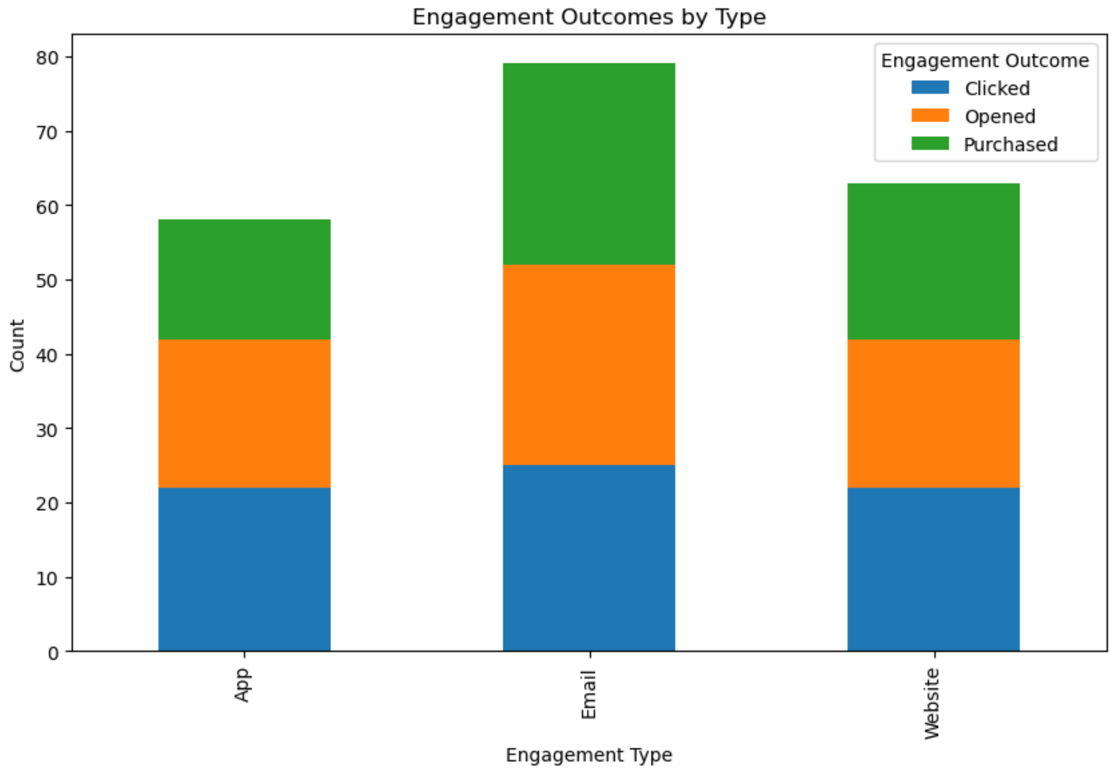
```
# Display the engagement summary
engagement_summary.head()
```

[24]:

```
EngagementOutcome  Clicked  Opened  Purchased
EngagementType
App                     22      20         16
Email                   25      27         27
Website                 22      20         21
```

[25]:

```
# Visualizing engagement outcomes
engagement_summary.plot(kind='bar', stacked=True, figsize=(10, 6))
plt.title('Engagement Outcomes by Type')
plt.xlabel('Engagement Type')
plt.ylabel('Count')
plt.legend(title='Engagement Outcome')
plt.show()
```



Result: A stacked bar plot showing engagement outcomes by type, and a snippet of the engagement summary table.

[ ]:

```

```

5. Customer Lifetime Value (CLV) Analysis

Questions: 1. How is CLV currently calculated?

2. Are there specific customer segments or behaviors associated with higher CLV?

3. What marketing strategies have been linked to increases in CLV?

[27]:
```
# Calculating CLV
customers_df['CLV'] = customers_df['TotalSpend']
```

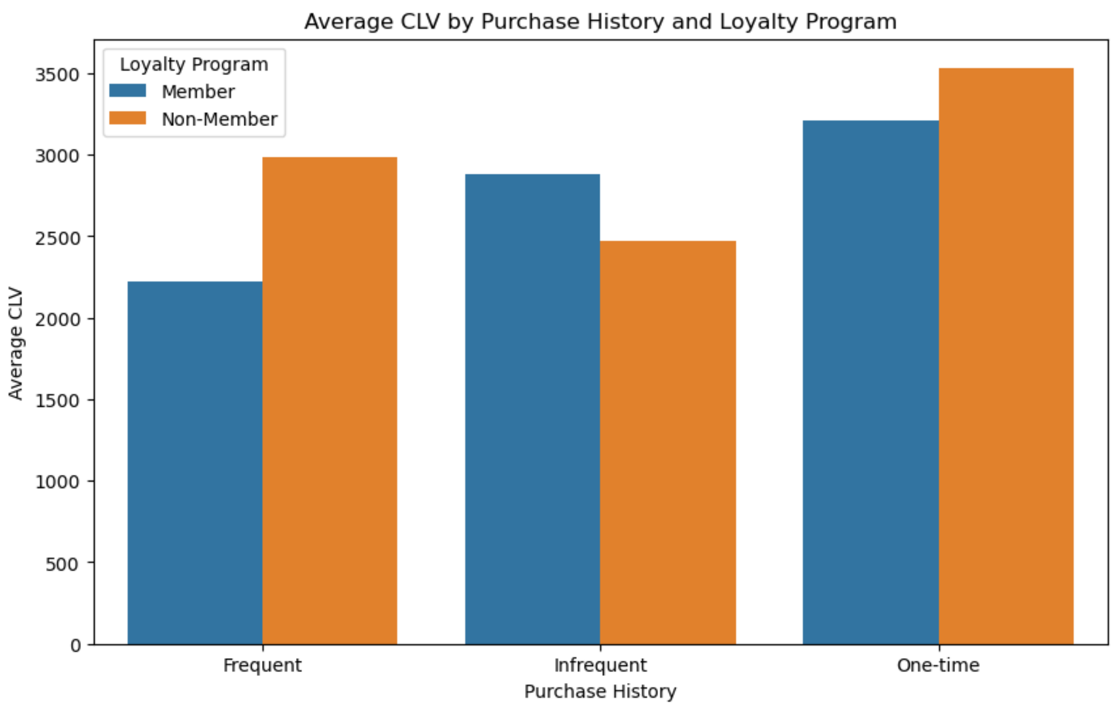
[28]:
```
# Analyzing CLV by segments
clv_segments = customers_df.groupby(['PurchaseHistory', 'LoyaltyProgram']).agg({
    'CLV': 'mean',
    'CustomerID': 'count'
}).rename(columns={'CustomerID': 'CustomerCount'}).reset_index()
```

[29]:
```
# Display the CLV segments
clv_segments.head()
```

[29]:

	PurchaseHistory	LoyaltyProgram	CLV	CustomerCount
0	Frequent	Member	2225.539375	16
1	Frequent	Non-Member	2985.458889	18
2	Infrequent	Member	2881.757742	31
3	Infrequent	Non-Member	2471.741765	17
4	One-time	Member	3210.364615	13

[30]:
```
# Visualizing CLV
plt.figure(figsize=(10, 6))
sns.barplot(data=clv_segments, x='PurchaseHistory', y='CLV',␣
 ↪hue='LoyaltyProgram')
plt.title('Average CLV by Purchase History and Loyalty Program')
plt.xlabel('Purchase History')
plt.ylabel('Average CLV')
plt.legend(title='Loyalty Program')
plt.show()
```



[ ]:
```
```

6. Loyalty Program Evaluation

Questions: 1. What are the current loyalty program's key features and benefits?

2. How is participation in the loyalty program tracked and measured?

3. What feedback have customers given about the loyalty program?

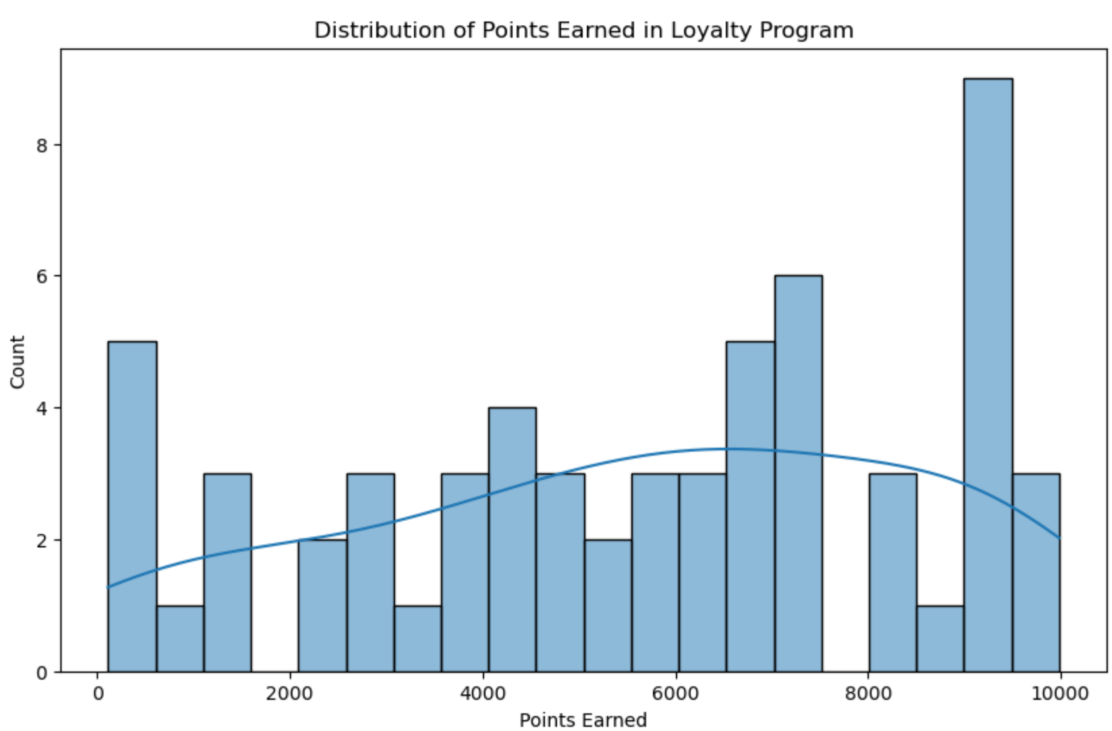
Below the loyalty program is evaluated to understand participation and its impact on customer retention.

[32]:
```
# Analyzing loyalty program data
loyalty_summary = loyalty_df.describe()
loyalty_summary
```

[32]:

	CustomerID	PointsEarned
count	60.000000	60.000000
mean	1044.483333	5613.850000
std	26.572250	2933.059968
min	1001.000000	111.000000
25%	1020.750000	3554.750000
50%	1045.000000	5902.000000
75%	1066.250000	8294.750000
max	1096.000000	9984.000000

[34]:
```
# Visualizing points distribution
plt.figure(figsize=(10, 6))
sns.histplot(loyalty_df['PointsEarned'], bins=20, kde=True)
plt.title('Distribution of Points Earned in Loyalty Program')
plt.xlabel('Points Earned')
plt.ylabel('Count')
plt.show()
```



Result: A histogram showing the distribution of points earned in the loyalty program, and a summary of the loyalty program data.

[ ]:

```

```

7. Customer Feedback Analysis

Questions: 1. What methods are used to collect and store customer feedback?

2. Are there any common themes or issues that have already been identified?

3. How frequently is customer feedback reviewed and analyzed?

Feedback scores will be analysed. Commonalities in feedback will also be checked.

[35]:

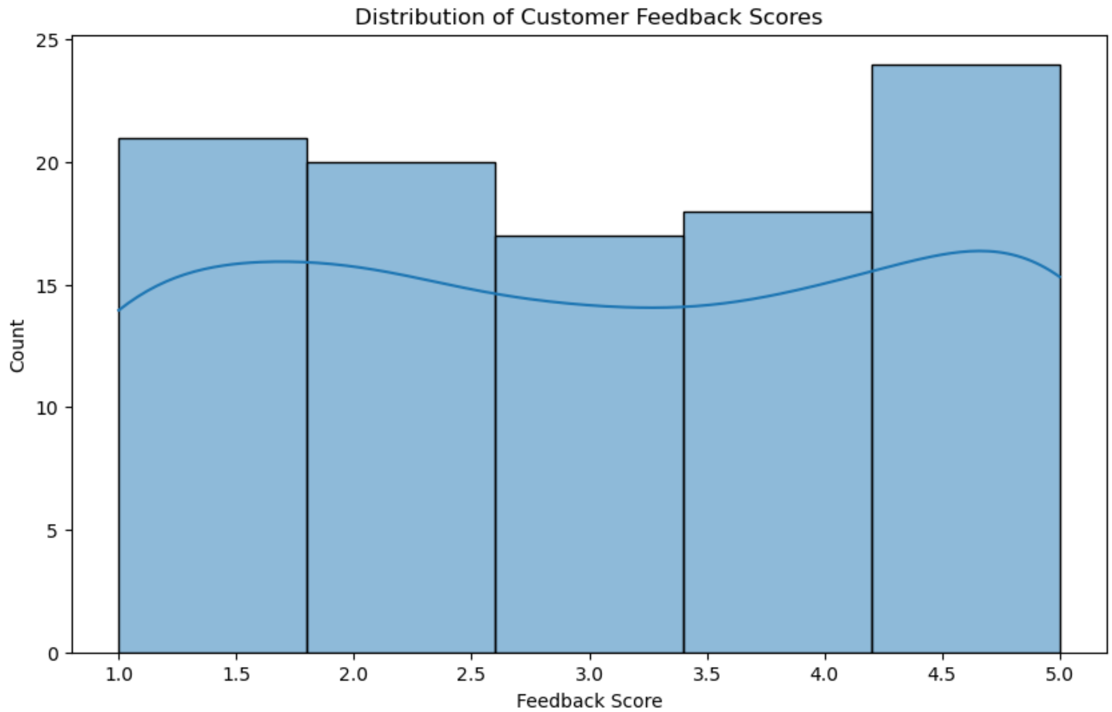
```python
# Analyzing customer feedback scores
feedback_summary = customers_df['FeedbackScore'].describe()
feedback_summary
```

[35]:

```
count    100.000000
mean       3.040000
std        1.483376
min        1.000000
25%        2.000000
50%        3.000000
75%        4.000000
max        5.000000
Name: FeedbackScore, dtype: float64
```

[36]:

```python
# Visualizing feedback scores distribution
plt.figure(figsize=(10, 6))
sns.histplot(customers_df['FeedbackScore'], bins=5, kde=True)
plt.title('Distribution of Customer Feedback Scores')
plt.xlabel('Feedback Score')
plt.ylabel('Count')
plt.show()
```



Result: Above is a histogram showing the distribution of customer feedback scores, and a summary of the feedback scores.

[ ]:

[39]:
```
import numpy as np
```

8. A/B Testing and Optimization

Questions: 1. What types of A/B tests have been conducted previously?

2. What metrics are used to determine the success of A/B tests?

3. How are test results currently documented and implemented?

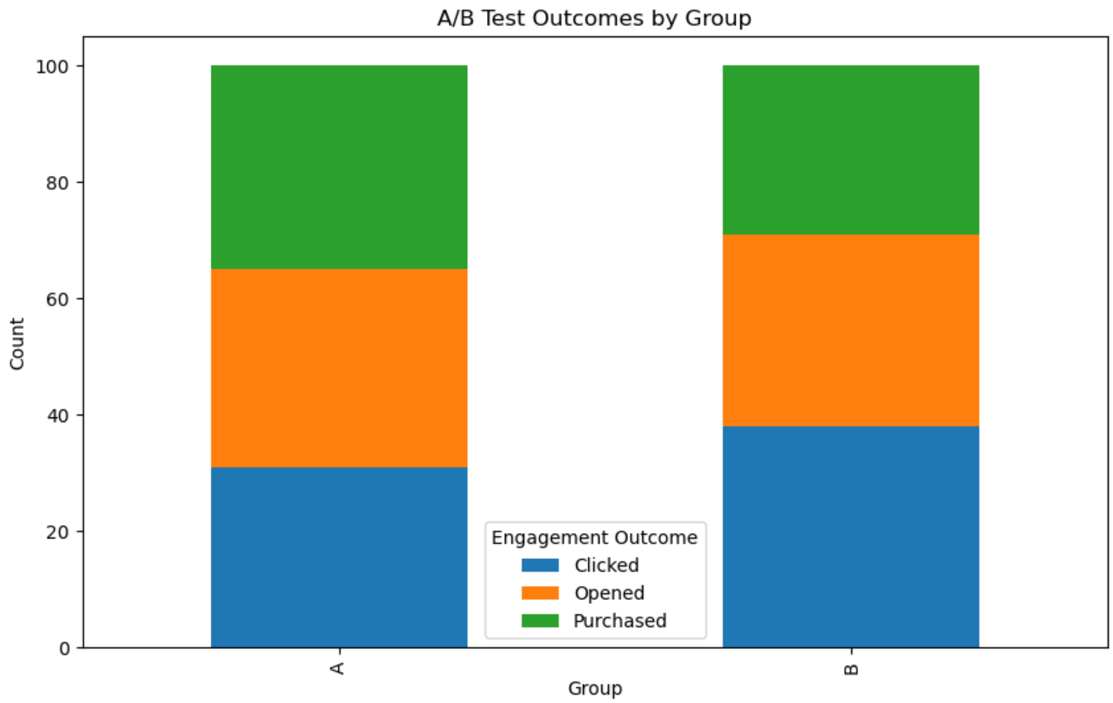
The analysis will involve a simulation of an A/B test by splitting the engagement data and comparing outcomes.

[40]:
```
# Simulating A/B test with engagement data
engagements_df['Group'] = np.random.choice(['A', 'B'], size=len(engagements_df))
```

[41]:
```
# Analyzing A/B test results
ab_test_results = engagements_df.groupby(['Group', 'EngagementOutcome']).size().
    ↪unstack(fill_value=0)
ab_test_results.head()
```

[41]:
```
EngagementOutcome  Clicked  Opened  Purchased
Group
A                       31      34         35
B                       38      33         29
```

[42]:
```
# Visualizing A/B test outcomes
ab_test_results.plot(kind='bar', stacked=True, figsize=(10, 6))
plt.title('A/B Test Outcomes by Group')
plt.xlabel('Group')
plt.ylabel('Count')
plt.legend(title='Engagement Outcome')
plt.show()
```



Result: A stacked bar plot showing A/B test outcomes by group, and a snippet of the A/B test results table.

[ ]:

9. Regular Monitoring and Reporting

Questions: 1. What key performance indicators (KPIs) are most critical for monitoring customer retention?

2. How are these KPIs currently tracked and reported?

3. What tools and platforms are used for creating dashboards and reports?

We will identify key KPIs and create a sample dashboard using matplotlib.

[45]:
```python
# Define key KPIs
kpis = {
    'Total Customers': len(customers_df),
    'Average CLV': customers_df['CLV'].mean(),
    'Churn Rate': customers_df['Churn'].mean(),
    'Average Feedback Score': customers_df['FeedbackScore'].mean()
}
```

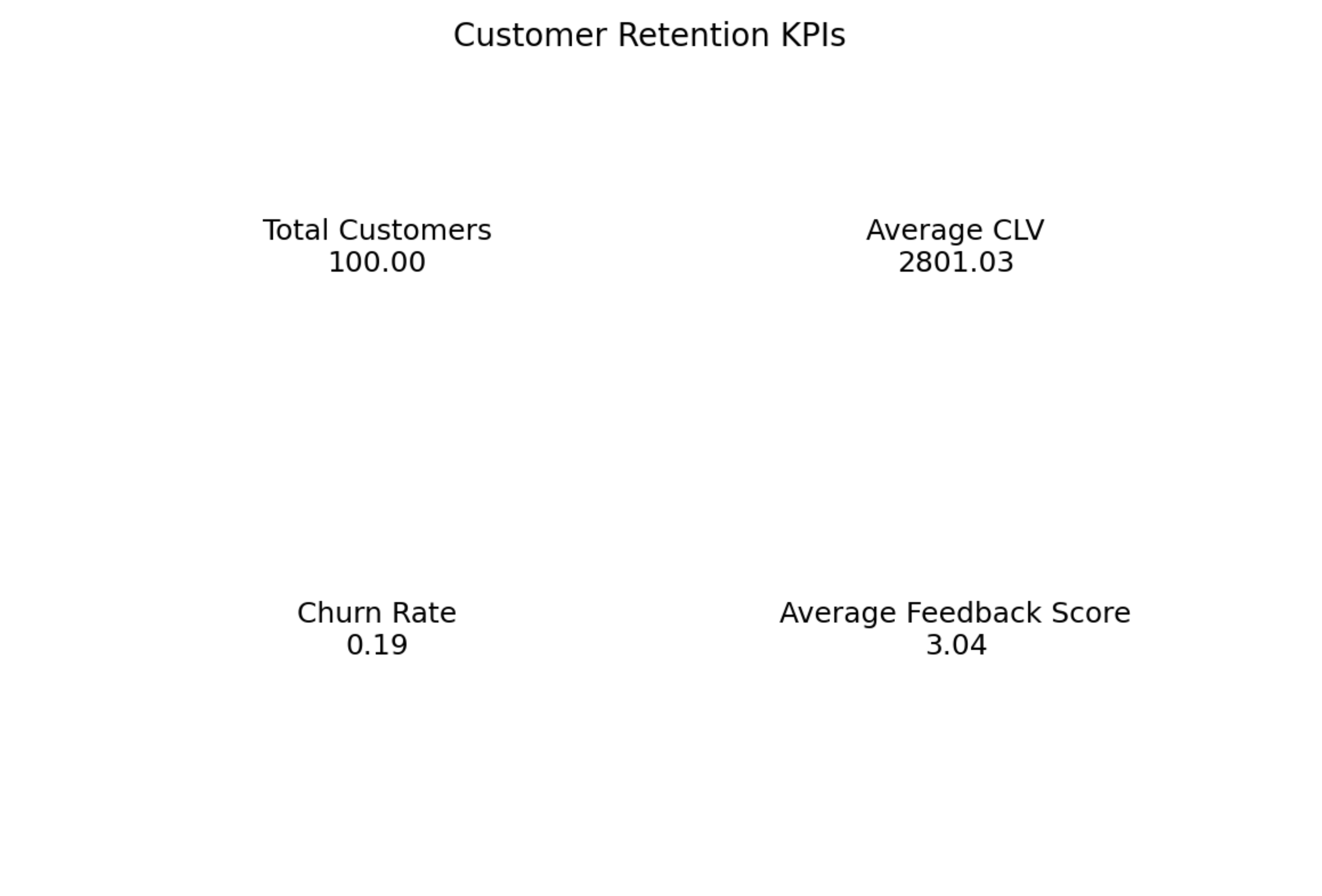
[46]:
```python
# Display the KPIs
kpis
```

[46]:
```
{'Total Customers': 100,
 'Average CLV': 2801.033,
 'Churn Rate': 0.19,
 'Average Feedback Score': 3.04}
```

[47]:
```python
# Creating a simple KPI dashboard
fig, ax = plt.subplots(2, 2, figsize=(12, 8))
ax = ax.flatten()

for i, (kpi, value) in enumerate(kpis.items()):
    ax[i].text(0.5, 0.5, f"{kpi}\n{value:.2f}", fontsize=18, ha='center')
    ax[i].axis('off')

plt.suptitle('Customer Retention KPIs', fontsize=20)
plt.show()
```



Result: A simple KPI dashboard visualizing key metrics for customer retention, and a dictionary showing the KPI values.

[ ]:

```

```