Understanding Customer Behavior with RFM Analysis and Python

Introduction

In today's competitive retail landscape, understanding customer behavior is crucial to a business's success. One powerful technique for analyzing customer behavior is RFM (Recency, Frequency, Monetary) analysis. In this tutorial, we will explore how to perform RFM analysis using Python on a dataset containing behavior data from an online store. By the end of this tutorial, you will be able to segment customers, visualize their behavior, and gain valuable insights to improve your marketing strategies.

Prerequisites

```
* Basic understanding of Python
* Familiarity with Pandas and Matplotlib libraries
```

Step 1: Import necessary libraries and load the dataset

First, we will import the necessary libraries, Pandas, Matplotlib, and Seaborn, read and concatenate our datasets into a DataFrame.

Entrée [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import squarify
```

Entrée [2]:

```
sns.set_theme(style="whitegrid")
```

Entrée [3]:

```
dec_df = pd.read_csv('archive-customers-of-cosmetics-shop/2019-Dec.csv')
nov_df = pd.read_csv('archive-customers-of-cosmetics-shop/2019-Nov.csv')
oct_df = pd.read_csv('archive-customers-of-cosmetics-shop/2019-Oct.csv')
feb_df = pd.read_csv('archive-customers-of-cosmetics-shop/2020-Feb.csv')
jan_df = pd.read_csv('archive-customers-of-cosmetics-shop/2020-Jan.csv')
list_df=[dec_df, nov_df, oct_df, feb_df, jan_df]
crm_df=pd.concat(list_df, axis=0)
```

Step 2: Preprocess and explore the data

Next, convert the 'event_time' column to a datetime object and perform basic exploratory data analysis by printing the shape, data types, missing values, and statistical description of the dataset.

Entrée [4]:

```
crm_df['event_time'] = pd.to_datetime(crm_df['event_time'])
```

Entrée [5]:

```
crm_df.head()
```

Out[5]:

	event_time	event_type	product_id	category_id	category_code	bra
0	2019-12-01 00:00:00+00:00	remove_from_cart	5712790	1487580005268456287	NaN	f.c
1	2019-12-01 00:00:00+00:00	view	5764655	1487580005411062629	NaN	c
2	2019-12-01 00:00:02+00:00	cart	4958	1487580009471148064	NaN	run
3	2019-12-01 00:00:05+00:00	view	5848413	1487580007675986893	NaN	freedec
4	2019-12-01 00:00:07+00:00	view	5824148	1487580005511725929	NaN	Na

Entrée [6]:

```
crm_df.shape
```

Out[6]:

```
(20692840, 9)
```

Entrée [7]:

```
# crm_df.info()
```

Entrée [8]:

```
# crm_df.isnull().sum()
```

Entrée [9]:

```
# crm_df.describe()
```

Step 3: Perform RFM analysis

Now, we will calculate Recency, Frequency, and Monetary Value metrics for each user based on their purchases. Group the dataset by 'user_id' and aggregate the relevant columns.

Entrée [10]:

```
snapshot_date = crm_df["event_time"].max() + pd.Timedelta(days=1)
crm_df = crm_df[crm_df["event_type"] == "purchase"].groupby("user_id").agg({
    "event_time": lambda x: (snapshot_date - x.max()).days,
    "product_id": "count",
    "price": "sum"
})

crm_df.rename(columns={
    "event_time": "recency",
    "product_id": "frequency",
    "price": "monetary_value"
}, inplace=True)
```

Entrée [11]:

```
crm_df.head()
```

Out[11]:

	recency	frequency	monetary_value
user_id			
9794320	97	4	12.68
10079204	116	2	25.81
10280338	11	86	177.83
12055855	72	4	16.54
12936739	44	2	29.89

Step 4: Create RFM segments

Segment customers based on their RFM metrics by assigning quartile values to each of the R, F, and M columns.

Entrée [12]:

```
rfm_table = crm_df.copy()
quartiles = rfm_table.quantile(q=[0.25, 0.5, 0.75])
rfm_table["R"] = pd.qcut(rfm_table["recency"], q=4, labels=list(reversed(range(1, 5))))
rfm_table["F"] = pd.qcut(rfm_table["frequency"], q=4, labels=range(1, 5))
rfm_table["M"] = pd.qcut(rfm_table["monetary_value"], q=4, labels=range(1, 5))
rfm_table["RFM_Segment"] = rfm_table["R"].astype(str) + rfm_table["F"].astype(str) + rfm
rfm_table["RFM_Score"] = rfm_table[["R", "F", "M"]].sum(axis=1)
```

Entrée [13]:

```
rfm_table
```

Out[13]:

	recency	frequency	monetary_value	R	F	M	RFM_Segment	RFM_Score
user_id								
9794320	97	4	12.68	2	2	1	221	5
10079204	116	2	25.81	1	1	2	112	4
10280338	11	86	177.83	4	4	4	444	12
12055855	72	4	16.54	2	2	2	222	6
12936739	44	2	29.89	3	1	2	312	6
...	...	...	...	...	...	...	...	...
622065819	1	4	20.48	4	2	2	422	8
622066161	1	4	12.95	4	2	1	421	7
622067983	1	10	66.48	4	3	4	434	11
622069477	1	1	0.95	4	1	1	411	6
622073202	1	1	3.81	4	1	1	411	6

110518 rows × 8 columns

Entrée [14]:

```
def rfm_segment_label(r, f, m):
    if r == 4 and f == 4 and m == 4:
        return 'Champions'
    elif r >= 3 and f >= 3 and m >= 3:
        return 'Loyal Customers'
    elif r >= 2 and f >= 2 and m >= 2:
        return 'Potential Loyalists'
    elif r >= 2 and f <= 2 and m >= 2:
        return 'Promising Customers'
    elif r <= 2 and f >= 2 and m >= 2:
        return 'At Risk Customers'
    else:
        return 'Lost Customers'

# Apply the modified custom function to relabel RFM segments
rfm_table["RFM_Segment_Label"] = rfm_table.apply(lambda x: rfm_segment_label(x["R"], x["
rfm_table.head()
```

Out[14]:

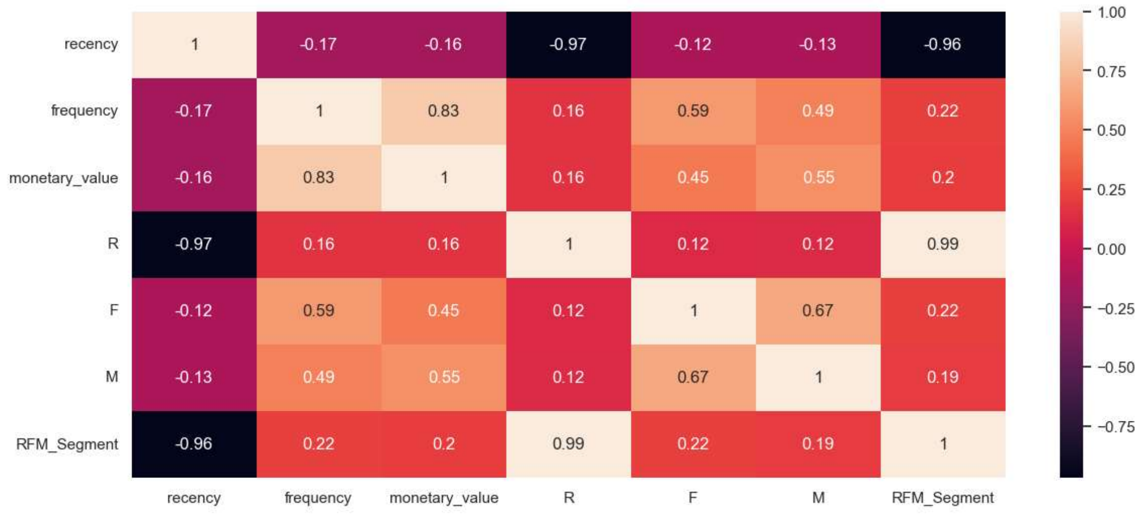
	recency	frequency	monetary_value	R	F	M	RFM_Segment	RFM_Score	RFM_S
user_id									
9794320	97	4	12.68	2	2	1	221	5	l
10079204	116	2	25.81	1	1	2	112	4	l
10280338	11	86	177.83	4	4	4	444	12	
12055855	72	4	16.54	2	2	2	222	6	Po
12936739	44	2	29.89	3	1	2	312	6	Promis

Step 5: Visualize RFM analysis

Visualize the RFM analysis using heatmaps and bubble charts to better understand the relationships between Recency, Frequency, and Monetary Value.

Entrée [15]:

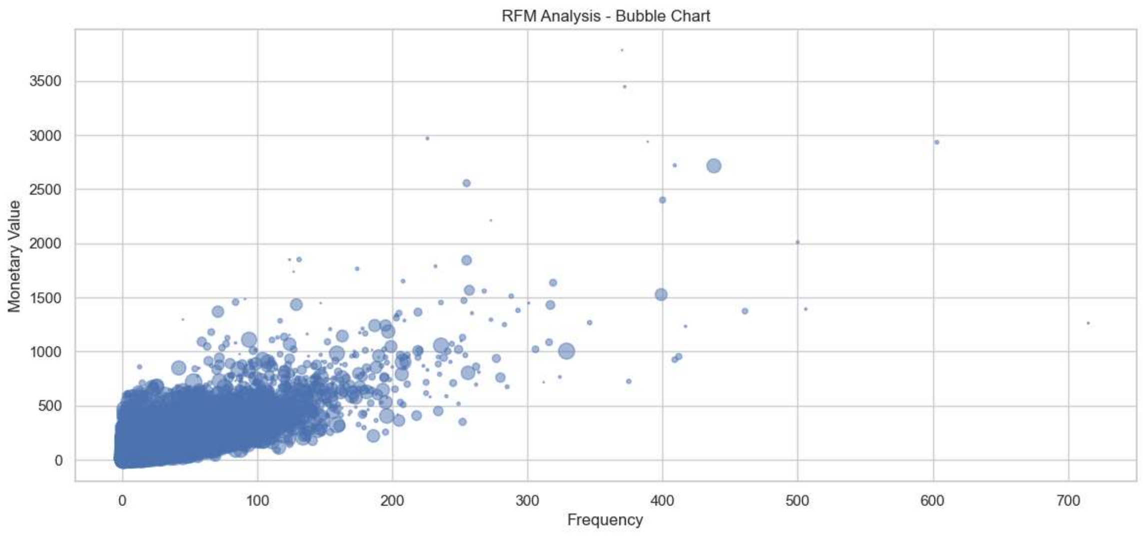
```
# Heatmap
heatmap_df = rfm_table.iloc[:, :-2]
plt.figure(figsize=(14, 6))
sns.heatmap(heatmap_df.corr(), annot=True)
plt.show()
```



	recency	frequency	monetary_value	R	F	M	RFM_Segment
recency	1	-0.17	-0.16	-0.97	-0.12	-0.13	-0.96
frequency	-0.17	1	0.83	0.16	0.59	0.49	0.22
monetary_value	-0.16	0.83	1	0.16	0.45	0.55	0.2
R	-0.97	0.16	0.16	1	0.12	0.12	0.99
F	-0.12	0.59	0.45	0.12	1	0.67	0.22
M	-0.13	0.49	0.55	0.12	0.67	1	0.19
RFM_Segment	-0.96	0.22	0.2	0.99	0.22	0.19	1

Entrée [16]:

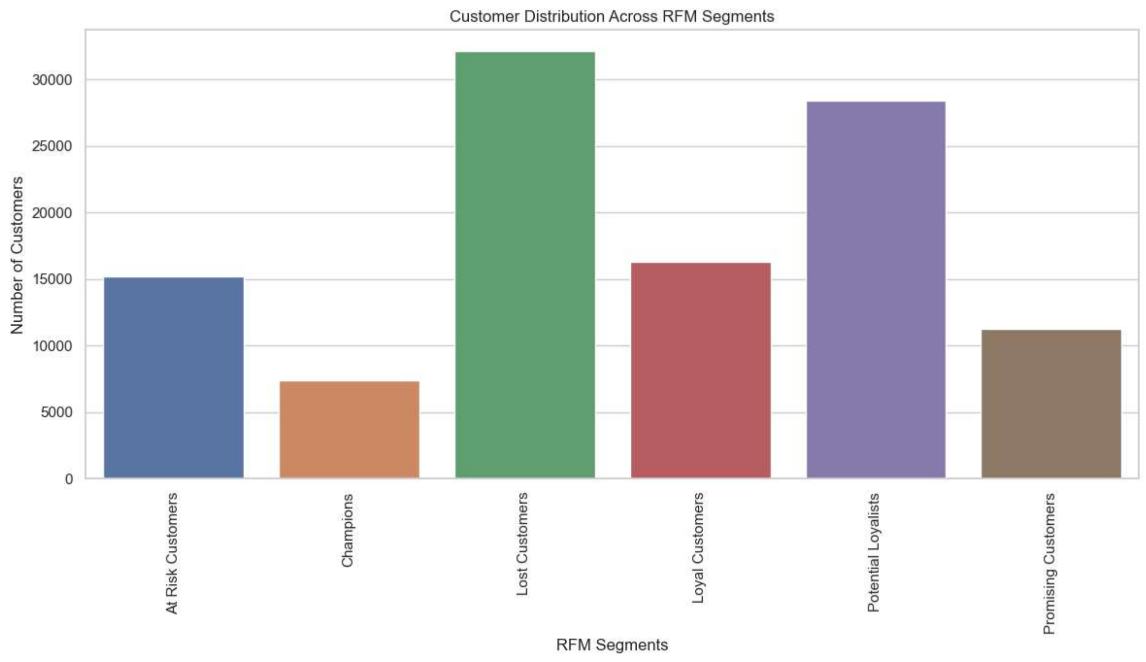
```
# Bubble chart
plt.figure(figsize=(14, 6))
plt.scatter(rfm_table["frequency"], rfm_table["monetary_value"], s=rfm_table["recency"],
plt.xlabel("Frequency")
plt.ylabel("Monetary Value")
plt.title("RFM Analysis - Bubble Chart")
plt.show()
```



Entrée [17]:

```
# Count the number of customers in each RFM segment
segment_counts = rfm_table["RFM_Segment_Label"].value_counts().sort_index()

# Plot customer segments as a bar chart
plt.figure(figsize=(14, 6))
sns.barplot(x=segment_counts.index, y=segment_counts.values)
plt.xlabel("RFM Segments")
plt.ylabel("Number of Customers")
plt.title("Customer Distribution Across RFM Segments")
plt.xticks(rotation=90)
plt.show()
```

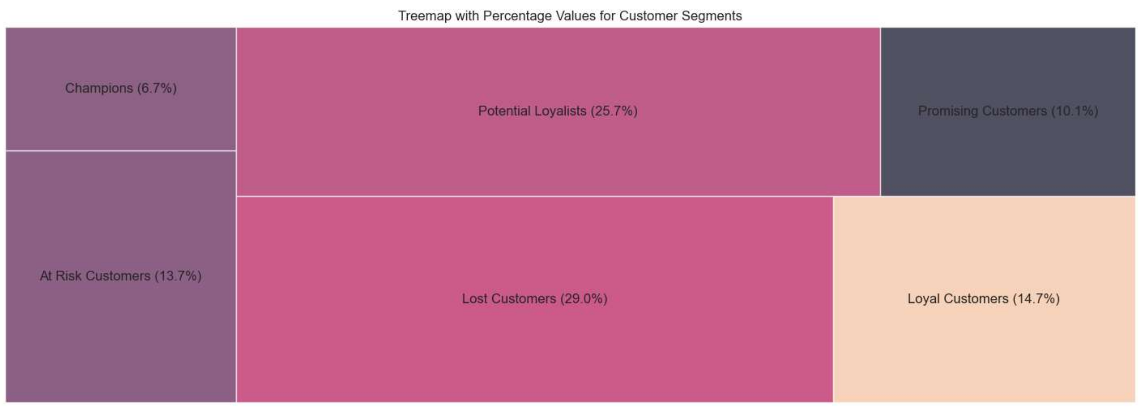


Entrée [18]:

```
# Count the number of customers in each RFM segment label
segment_label_counts = rfm_table["RFM_Segment_Label"].value_counts()

# Prepare the data for plotting
total_customers = segment_counts.sum()
segment_percentages = segment_counts / total_customers * 100
sizes = segment_percentages.values
labels = [f"{segment} ({percentage:.1f}%)" for segment, percentage in zip(segment_percen

# Plot the treemap using squarify
plt.figure(figsize=(18, 6))
squarify.plot(sizes=sizes, label=labels, alpha=0.7)
plt.axis('off')
plt.title("Treemap with Percentage Values for Customer Segments")
plt.show()
```



Here is a description of each label of RFM segmentation:

```
1- Champions: These are your best customers who have made the most recent purch
ases, purchased the highest quantity, and spent the most. They are highly engag
ed, loyal, and likely to make repeat purchases. Focus on keeping them satisfied
and engaged with personalized offers, exclusive deals, and premium services.

2- Loyal Customers: These customers have high recency, frequency, and monetary
scores (equal to or greater than 3). They are regular shoppers and have spent a
significant amount on your products or services. Strengthen your relationship w
ith them through loyalty programs, special discounts, and tailored communicatio
n to maintain their loyalty and encourage referrals.

3- Potential Loyalists: These customers have moderate recency, frequency, and m
onetary scores (equal to or greater than 2). They have the potential to become
loyal customers if nurtured properly. Offer personalized recommendations, incen
tives, and targeted marketing campaigns to increase their engagement and move t
hem into the loyal customer segment
```

Conclusion

In this tutorial, we demonstrated how to perform RFM analysis using Python on a dataset from an online store. By segmenting customers based on their Recency, Frequency, and Monetary Value, you can gain valuable insights into customer behavior. This information can be used to improve marketing strategies, personalize customer experiences, and ultimately drive business growth.