Cohort Analysis Explained

What is Cohort Analysis?

Cohort analysis is a subset of behavioral analytics that breaks users into related groups for analysis. These groups, or 'cohorts', usually share common characteristics or experiences within a defined time-span. In eCommerce and business intelligence, cohort analysis is useful for tracking customer retention, purchase behavior over time, and lifecycle value.

Notebook Cell Explanations

Cell 1: Markdown (Introduction)

This cell introduces the topic of cohort analysis and mentions the dataset used, which is from the UCI Machine Learning Repository. It briefly explains that the data is useful for supervised and unsupervised learning tasks.

Cell 2: Loading the Dataset

This code cell uses pandas to read the CSV file from an online source. The data is transactional, coming from an online retail store. The dataset includes fields such as InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, and Country.

Code snippet:

```
import pandas as pd
df =
pd.read_csv('https://coding.co.ke/datasets/OnlineRetail.csv')
df
```

Cell 3: Dataset Description

This markdown cell describes the dataset in detail. It contains all transactions from 01/12/2010 to 09/12/2011 for a UK-based online retailer that sells unique gifts, often in bulk to wholesalers.

Cell 4: Code

Code snippet:

```
import pandas as pd
df =
pd.read_csv('https://coding.co.ke/datasets/OnlineRetail.csv',dtype={'CustomerID':
str,'InvoiceNo': str},parse_dates=['InvoiceDate'],
               infer_datetime_format=True)

df
```

Cell 5: Markdown

We can also inspect the DataFrame using df.info() to see if there are missing values. As for the analysis, due to the fact that we need to have the customer IDs, we drop all the rows without them.

Cell 6: Code

Code snippet:

```
df.dropna(subset=['CustomerID'], inplace=True)
```

Cell 7: Markdown

In total, there are around 9 thousand purchases with a negative quantity. We remove them from the dataset. This introduces a kind of bias, as we include the initial orders and remove the return — this way the initial order is taken into account even though in theory it was not realized and did not generate revenue.

Then, we calculate an aggregate metric indicating how many orders were placed by each customer.

Cell 8: Code

Code snippet:

```
import numpy as np
n_orders = df.groupby(['CustomerID'])['InvoiceNo'].nunique()
n_orders
```

Cell 9: Markdown

We can get the percentage,
Using the code above, we can state that 65.57% of customers ordered more than once. This is already a valuable piece of information, as is seems that the

customers are placing multiple orders. This means that there will be at least some retention. Given that the dataset has no sign-up/joined date, it would be problematic if the majority of the users only placed one order, but we will get back to it later.

Cell 10: Code

Code snippet:

```
mult_orders_perc = np.sum(n_orders > 1) / df['CustomerID'].nunique()
print(f'{100 * mult_orders_perc:.2f}% of customers ordered more than once.')
```

Cell 11: Markdown

We create a seaborn Plot to show number of numbers by numbe of orders. Additionally, we look at the distribution of the number of orders per customer. For that, we can reuse the previously aggregated data (n_orders) and plot the data on a histogram.

Cell 12: Code

Code snippet:

```
import seaborn as sns
ax = sns.distplot(n_orders, kde=False, hist=True)
ax.set(title='Distribution of number of orders per customer',
    xlabel='# of orders',
    ylabel='# of customers');
```

Cell 13: Markdown

As the first step, we keep only the relevant columns and drop duplicated values — one order (indicated by InvoiceNo) can contain multiple items (indicated by StockCode).

Cell 14: Code

Code snippet:

```
df = df[['CustomerID', 'InvoiceNo', 'InvoiceDate']].drop_duplicates()
df
```

Cell 15: Code

Code snippet:

```
df['order_month'] = df['InvoiceDate'].dt.to_period('M')
df
```

Cell 16: Markdown

As the second step, we create the cohort and order_month variables. The first one indicates the monthly cohort based on the first purchase date (calculated per customer). The latter one is the truncated month of the purchase date.

Cell 17: Code

Code snippet:

```
df['cohort'] = df.groupby('CustomerID')['InvoiceDate'] \
                .transform('min') \
                .dt.to_period('M')
df.tail(50)
```

Cell 18: Markdown

Then, we aggregate the data per cohort and order_month and count the number of unique customers in each group. Additionally, we add the period_number, which indicates the number of periods between the cohort month and the month of the purchase.

Cell 19: Code

Code snippet:

```
from operator import attrgetter
df_cohort = df.groupby(['cohort', 'order_month']) \
            .agg(n_customers=('CustomerID', 'nunique')) \
            .reset_index(drop=False)
df_cohort['period_number'] = (df_cohort.order_month -
df_cohort.cohort).apply(attrgetter('n'))
```

Cell 20: Markdown

The next step is to pivot the df_cohort table in a way that each row contains information about a given cohort and each column contains values for a certain period.

Cell 21: Code

Code snippet:

```
cohort_pivot = df_cohort.pivot_table(index = 'cohort',
                                     columns = 'period_number',
                                     values = 'n_customers')

cohort_pivot
```

Cell 22: Markdown

To obtain the retention matrix, we need to divide the values each row by the row's first value, which is actually the cohort size — all customers who made their first purchase in the given month.

Cell 23: Code

Code snippet:

```
cohort_size = cohort_pivot.iloc[:,0]
retention_matrix = cohort_pivot.divide(cohort_size, axis = 0)
retention_matrix
```

Cell 24: Markdown

Lastly, we plot the retention matrix as a heatmap. Additionally, we wanted to include extra information regarding the cohort size. That is why we in fact created two heatmaps, where the one indicating the cohort size is using a white only colormap.

Cell 25: Code

Code snippet:

```
import matplotlib.pyplot as plt
from matplotlib import colors as mcolors
```

```
with sns.axes_style("white"):
    fig, ax = plt.subplots(1, 2, figsize=(12, 8), sharey=True,
gridspec_kw={'width_ratios': [1, 11]})

    # retention matrix
    sns.heatmap(retention_matrix,
                mask=retention_matrix.isnull(),
                annot=True,
                fmt='.0%',
                cmap='RdYlGn',
                ax=ax[1])
    ax[1].set_title('Monthly Cohorts: User Retention', fontsize=16)
    ax[1].set(xlabel='# of periods',
              ylabel='')

    # cohort size
    cohort_size_df = pd.DataFrame(cohort_size).rename(columns={0: 'cohort_size'})
    white_cmap = mcolors.ListedColormap(['white'])
    sns.heatmap(cohort_size_df,
                annot=True,
                cbar=False,
                fmt='g',
                cmap=white_cmap,
                ax=ax[0])

    fig.tight_layout()
```

Cell 26: Markdown

In the image, we can see that there is a sharp drop-off in the second month (indexed as 1) already, on average around 80% of customers do not make any purchase in the second month. The first cohort (2010–12) seems to be an exception and performs surprisingly well as compared to the other ones. A year after the first purchase, there is a 50% retention. This might be a cohort of dedicated customers, who first joined the platform based on some already-existing connections with the retailer. However, from data alone, that is very hard to accurately explain.

Throughout the matrix, we can see fluctuations in retention over time. This might be caused by the characteristics of the business, where clients do periodic purchases, followed by periods of inactivity.