Performance Indicators in Retail Networks

April 14, 2023

1 Performance Indicators in Retail Networks Analysis

[1]:

```
# Python Language Version
from platform import python_version
print('Python Language Version Used In This Jupyter Notebook:',␣
  ↪python_version())
```

```
Python Language Version Used In This Jupyter Notebook: 3.7.16
```

1.1 Defining the Problem

Our job is to calculate, analyze and interpret 8 key performance indicators based on the data provided. The data are fictitious, but represent values that can be considered real.

The indicators were defined by the company's strategic planning area, which needs to monitor the evolution of sales and the effectiveness of Marketing campaigns over time.

Here are the 8 indicators that will be part of our review:

- Indicator 1 - Monthly Revenue
- Indicator 2 - Monthly Percentage Growth Rate
- Indicator 3 - Active Customers Per Month in a Country (Brazil)
- Indicator 4 - Total Items Purchased Per Month in a Country (Brazil)
- Indicator 5 - Average Monthly Revenue in a Country (Brazil)
- Indicator 6 - Difference in Billing Over Time Between New and Old Clients
- Indicator 7 - Rate of New Clients
- Indicator 8 - Monthly Customer Retention Rate

1.2 Loading Packages

[2]:

```
# Imports
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib
import plotly
import matplotlib.pyplot as plt
import plotly.offline as pyoff
import plotly.graph_objs as go
from datetime import datetime, timedelta
```

```
%matplotlib inline
pyoff.init_notebook_mode()
```

1.3 Loading the Data

[3]:
```
# Loading the Data
data = pd.read_csv("data/dataset.csv", header = 0, encoding = 'unicode_escape')
```

[4]:
```
# Visualize the data
data.head()
```

[4]:
```
  BillNum ProductCode                          ProductName  Quantity  \
0  536365       21730    GLASS STAR FROSTED T-LIGHT HOLDER       6.0
1  536365      85123A   WHITE HANGING HEART T-LIGHT HOLDER       6.0
2  536365       71053                  WHITE METAL LANTERN       6.0
3  536365      84406B       CREAM CUPID HEARTS COAT HANGER       8.0
4  536365      84029G  KNITTED UNION FLAG HOT WATER BOTTLE       6.0

         SaleDate  UnitaryValue  CustomerID Country
0  12/1/2010 8:26          4.25     17850.0  Brasil
1  12/1/2010 8:26          2.55     17850.0  Brasil
2  12/1/2010 8:26          3.39     17850.0  Brasil
3  12/1/2010 8:26          2.75     17850.0  Brasil
4  12/1/2010 8:26          3.39     17850.0  Brasil
```

[5]:
```
# Shape
data.shape
```

[5]:
```
(541800, 8)
```

[6]:
```
# Data types
data.dtypes
```

[6]:
```
BillNum          object
ProductCode      object
ProductName      object
Quantity        float64
SaleDate         object
UnitaryValue    float64
CustomerID      float64
Country          object
dtype: object
```

[7]:
```
# Describe
data.describe()
```

```
[7]:            Quantity   UnitaryValue     CustomerID
     count  535765.000000  535765.000000  403078.000000
     mean        9.587418       4.637011   15287.751909
     std       153.307728      97.312120    1713.884183
     min    -74215.000000  -11062.060000   12346.000000
     25%         1.000000       1.250000   13953.000000
     50%         3.000000       2.080000   15152.000000
     75%        10.000000       4.130000   16791.000000
     max     74215.000000   38970.000000   18287.000000
```

```
[8]: # Checking for null values
     data.isna().sum()
```

```
[8]: BillNum             0
     ProductCode      6035
     ProductName      7489
     Quantity         6035
     SaleDate         6035
     UnitaryValue     6035
     CustomerID     138722
     Country          6035
     dtype: int64
```

```
[9]: # Convert date column to date type
     data.SaleDate = pd.to_datetime(data.SaleDate)
```

```
[10]: # Data types
      data.dtypes
```

```
[10]: BillNum                 object
      ProductCode             object
      ProductName             object
      Quantity               float64
      SaleDate        datetime64[ns]
      UnitaryValue           float64
      CustomerID             float64
      Country                 object
      dtype: object
```

```
[11]: # Date range of the period in which the sales occurred, now with the correct␣
       ↪data type
      print('Min Date:', data['SaleDate'].min())
      print('Max Date:', data['SaleDate'].max())
```

```
     Min Date: 2010-12-01 08:26:00
     Max Date: 2011-12-09 12:50:00
```

[12]:

```
# Countries to which sales took place
data['Country'].unique()
```

[12]:

```
array(['Brasil', 'Uruguai', nan, 'Australia', 'Holanda', 'Alemanha',
       'Noruega', 'Irlanda', 'Espanha', 'Poland', 'Portugal', 'Italy',
       'Belgium', 'Lithuania', 'Japan', 'Iceland', 'Channel Islands',
       'Dinamarca', 'Cyprus', 'Sweden', 'Austria', 'Israel', 'Finland',
       'Bahrain', 'Greece', 'Hong Kong', 'Cingapura', 'Iraque', 'Equador',
       'Saudi Arabia', 'Czech Republic', 'Canada', 'China', 'Inglaterra',
       'USA', 'Chile', 'Malta', 'Paraguai'], dtype=object)
```

1.3.1 Indicator 1 - Monthly Revenue

Billing = Quantity * Unit_Value

[13]:

```
# Extract monthly revenue
data['YearMonth'] = data['SaleDate'].map(lambda date: 100 * date.year + date.
 ↪month)
```

[14]:

```
# Visualize the data
data.head()
```

[14]:

```
  BillNum ProductCode                          ProductName  Quantity  \
0  536365       21730    GLASS STAR FROSTED T-LIGHT HOLDER       6.0
1  536365      85123A   WHITE HANGING HEART T-LIGHT HOLDER       6.0
2  536365       71053                  WHITE METAL LANTERN       6.0
3  536365      84406B       CREAM CUPID HEARTS COAT HANGER       8.0
4  536365      84029G  KNITTED UNION FLAG HOT WATER BOTTLE       6.0

             SaleDate  UnitaryValue  CustomerID Country  YearMonth
0 2010-12-01 08:26:00          4.25     17850.0  Brasil   201012.0
1 2010-12-01 08:26:00          2.55     17850.0  Brasil   201012.0
2 2010-12-01 08:26:00          3.39     17850.0  Brasil   201012.0
3 2010-12-01 08:26:00          2.75     17850.0  Brasil   201012.0
4 2010-12-01 08:26:00          3.39     17850.0  Brasil   201012.0
```

[15]:

```
# Calculate revenue
data["Revenue"] = data["Quantity"] * data["UnitaryValue"]
```

[16]:

```
# Visualize the data
data.head()
```

[16]:

```
  BillNum ProductCode                          ProductName  Quantity  \
0  536365       21730    GLASS STAR FROSTED T-LIGHT HOLDER       6.0
1  536365      85123A   WHITE HANGING HEART T-LIGHT HOLDER       6.0
2  536365       71053                  WHITE METAL LANTERN       6.0
3  536365      84406B       CREAM CUPID HEARTS COAT HANGER       8.0
4  536365      84029G  KNITTED UNION FLAG HOT WATER BOTTLE       6.0
```

```
             SaleDate  UnitaryValue  CustomerID Country  YearMonth  Revenue
0 2010-12-01 08:26:00          4.25     17850.0  Brasil   201012.0    25.50
1 2010-12-01 08:26:00          2.55     17850.0  Brasil   201012.0    15.30
2 2010-12-01 08:26:00          3.39     17850.0  Brasil   201012.0    20.34
3 2010-12-01 08:26:00          2.75     17850.0  Brasil   201012.0    22.00
4 2010-12-01 08:26:00          3.39     17850.0  Brasil   201012.0    20.34
```

```
[17]: # Group revenue by month/year
      df_revenue = data.groupby(['YearMonth']).agg({'Revenue': sum}).reset_index()
```

```
[18]: # Data table
      df_revenue
```

```
[18]:     YearMonth      Revenue
      0    201012.0   742758.820
      1    201101.0   553674.540
      2    201102.0   492636.260
      3    201103.0   678093.000
      4    201104.0   488332.991
      5    201105.0   717480.910
      6    201106.0   686094.980
      7    201107.0   676697.211
      8    201108.0   678906.680
      9    201109.0  1013901.152
      10   201110.0  1066055.680
      11   201111.0  1455571.020
      12   201112.0   428657.300
```

1.3.2 Visualization of Indicator 1

```
[19]: # Plot

      # Definition of data in the plot
      plot_data = [go.Scatter(x = df_revenue['YearMonth'],
                              y = df_revenue['Revenue'],)]

      # Layout
      plot_layout = go.Layout(xaxis = {"type": "category"},
                              title = 'Monthly Revenue')

      # Figure Plot
      fig = go.Figure(data = plot_data, layout = plot_layout)
      pyoff.iplot(fig)
```

1.3.3 Indicator 2 - Monthly Percentage Growth Rate

Percent Monthly Growth Rate = Monthly Revenue / Previous Monthly Revenue * 100

[20]:

```
# We use the pct_change() function to calculate the monthly percentage change
df_revenue['MonthlyGrowth'] = df_revenue['Revenue'].pct_change()
```

[21]:

```
# Data table
df_revenue
```

[21]:

```
    YearMonth      Revenue  MonthlyGrowth
0    201012.0   742758.820            NaN
1    201101.0   553674.540      -0.254570
2    201102.0   492636.260      -0.110242
3    201103.0   678093.000       0.376458
4    201104.0   488332.991      -0.279844
5    201105.0   717480.910       0.469245
6    201106.0   686094.980      -0.043745
7    201107.0   676697.211      -0.013697
8    201108.0   678906.680       0.003265
9    201109.0  1013901.152       0.493432
10   201110.0  1066055.680       0.051439
11   201111.0  1455571.020       0.365380
12   201112.0   428657.300      -0.705506
```

1.3.4 Visualization of Indicator 2

[22]:

```
# Plot

# Definition of data in the plot (we filter month 12 of 2011 because we don't␣
 ↪have enough data)
plot_data = [go.Scatter(x = df_revenue.query("YearMonth < 201112")['YearMonth'],
                        y = df_revenue.query("YearMonth <␣
 ↪201112")['MonthlyGrowth'],)]

# Layout
plot_layout = go.Layout(xaxis = {"type": "category"},
                        title = 'Monthly Percentage Growth Rate')

# Figure Plot
fig = go.Figure(data = plot_data, layout = plot_layout)
pyoff.iplot(fig)
```

1.3.5 Indicator 3 - Active Customers Per Month in a Country (Brazil)

Active customers are those who have made at least one purchase each month.

[23]:

```
# Create a dataframe only with data from Brazil
brazil_data = data.query("Country=='Brasil'").reset_index(drop = True)
```

[24]:
```
# Active users are those who have made at least one purchase
df_month_active = brazil_data.groupby('YearMonth')['CustomerID'].nunique().
 ↪reset_index()
```

[25]:
```
# Data
df_month_active
```

[25]:
```
    YearMonth  CustomerID
0    201012.0         870
1    201101.0         684
2    201102.0         714
3    201103.0         922
4    201104.0         817
5    201105.0         985
6    201106.0         943
7    201107.0         899
8    201108.0         866
9    201109.0        1176
10   201110.0        1285
11   201111.0        1548
12   201112.0         611
```

1.3.6 Visualization of Indicator 3

[26]:
```
# Plot

# Definition of data in the plot
plot_data = [go.Bar(x = df_month_active['YearMonth'],
                    y = df_month_active['CustomerID'],)]

# Layout
plot_layout = go.Layout(xaxis = {"type": "category"},
                        title = 'Active Customers Per Month in a Country␣
 ↪(Brazil)')

# Figure plot
fig = go.Figure(data = plot_data, layout = plot_layout)
pyoff.iplot(fig)
```

1.3.7 Indicator 4 - Total Items Purchased Per Month in a Country (Brazil)

Total items purchased per month.

[27]:
```
# Group the data to calculate the total items purchased per month in Brazil
df_month_items = brazil_data.groupby('YearMonth')['Quantity'].sum().
 ↪reset_index()
```

[28]:

```
# Data
df_month_items
```

[28]:

```
    YearMonth  Quantity
0    201012.0  294305.0
1    201101.0  235404.0
2    201102.0  221731.0
3    201103.0  276552.0
4    201104.0  254141.0
5    201105.0  303200.0
6    201106.0  255897.0
7    201107.0  319908.0
8    201108.0  316844.0
9    201109.0  454669.0
10   201110.0  466860.0
11   201111.0  637079.0
12   201112.0  196740.0
```

1.3.8 Visualization of Indicator 4

[29]:

```
# Plot

# Definition of data in the plot
plot_data = [go.Bar(x = df_month_items['YearMonth'],
                    y = df_month_items['Quantity'],)]

# Layout
plot_layout = go.Layout(xaxis = {"type": "category"},
                        title = 'Total Items Purchased Per Month in a Country␣
 ↪(Brazil)')

# Figure plot
fig = go.Figure(data = plot_data, layout = plot_layout)
pyoff.iplot(fig)
```

1.3.9 Indicator 5 - Average Monthly Sales in a Country (Brazil)

Average revenue per month in a country.

[30]:

```
# Calculate average revenue
df_average_revenue = brazil_data.groupby('YearMonth')['Revenue'].mean().
 ↪reset_index()
```

[31]:

```
# Data
df_average_revenue
```

[31]:

```
    YearMonth    Revenue
0    201012.0  16.949637
```

```
1    201101.0  13.713483
2    201102.0  16.148086
3    201103.0  16.835848
4    201104.0  15.830581
5    201105.0  17.782249
6    201106.0  16.814490
7    201107.0  15.814140
8    201108.0  17.392062
9    201109.0  19.014590
10   201110.0  16.145289
11   201111.0  16.368396
12   201112.0  16.295788
```

1.3.10 Visualization of Indicator 5

[32]:
```
# Plot

# Definition of data in the plot
plot_data = [go.Bar(x = df_average_revenue['YearMonth'],
                    y = df_average_revenue['Revenue'],)]

# Layout
plot_layout = go.Layout(xaxis = {"type": "category"},
                        title = 'Average Monthly Revenue in a Country (Brazil)')

# Figure plot
fig = go.Figure(data = plot_data, layout = plot_layout)
pyoff.iplot(fig)
```

[33]:
```
# Calculate total billing per month
df_total_revenue = brazil_data.groupby('YearMonth')['Revenue'].sum().
  ↪reset_index()
```

[34]:
```
# Data
df_total_revenue
```

[34]:
```
    YearMonth     Revenue
0    201012.0  671137.840
1    201101.0  428779.470
2    201102.0  403782.900
3    201103.0  555498.800
4    201104.0  438237.971
5    201105.0  592024.400
6    201106.0  550069.230
7    201107.0  561322.901
8    201108.0  535849.440
9    201109.0  857196.722
```

```
10   201110.0   873040.360
11   201111.0  1277258.660
12   201112.0   384303.560
```

[35]:

```
# Plot

# Definition of data in the plot
plot_data = [go.Bar(x = df_total_revenue['YearMonth'],
                    y = df_total_revenue['Revenue'],)]

# Layout
plot_layout = go.Layout(xaxis = {"type": "category"},
                        title = 'Total Monthly Revenue in a Country (Brazil)')

# Figure plot
fig = go.Figure(data = plot_data, layout = plot_layout)
pyoff.iplot(fig)
```

1.3.11 Indicator 6 - Difference in Revenue Over Time Between New and Old Customers

Let's consider new customers as those with a low volume of purchases and old customers as those with a high volume of purchases.

[36]:

```
# Let's find the lowest volume purchase date for each customer
df_min_purchase = data.groupby('CustomerID')["SaleDate"].min().reset_index()
```

[37]:

```
# Adjust the column names
df_min_purchase.columns = ['CustomerID', 'MinorPurchaseDate']
```

[38]:

```
# Let's extract the month in which the lowest volume of purchases for each␣
 ↪customer occurred
df_min_purchase['MonthMinorPurchaseMonthly'] =␣
 ↪df_min_purchase['MinorPurchaseDate'].map(lambda date: 100 * date.year + date.
 ↪month)
```

[39]:

```
# Data
df_min_purchase.head()
```

[39]:

```
   CustomerID   MinorPurchaseDate  MonthMinorPurchaseMonthly
0     12346.0 2011-01-18 10:01:00                     201101
1     12347.0 2010-12-07 14:57:00                     201012
2     12348.0 2010-12-16 19:09:00                     201012
3     12349.0 2011-11-21 09:51:00                     201111
4     12350.0 2011-02-02 16:01:00                     201102
```

[40]:

```
# Let's merge the original dataset with the purchase volume dataset
purchase_data = pd.merge(data, df_min_purchase, on = "CustomerID")
```

```
purchase_data.head()
```

[40]:
```
  BillNum ProductCode                          ProductName  Quantity  \
0  536365       21730    GLASS STAR FROSTED T-LIGHT HOLDER       6.0
1  536365      85123A   WHITE HANGING HEART T-LIGHT HOLDER       6.0
2  536365       71053                  WHITE METAL LANTERN       6.0
3  536365      84406B       CREAM CUPID HEARTS COAT HANGER       8.0
4  536365      84029G  KNITTED UNION FLAG HOT WATER BOTTLE       6.0

             SaleDate  UnitaryValue  CustomerID Country  YearMonth  Revenue  \
0 2010-12-01 08:26:00          4.25     17850.0  Brasil   201012.0    25.50
1 2010-12-01 08:26:00          2.55     17850.0  Brasil   201012.0    15.30
2 2010-12-01 08:26:00          3.39     17850.0  Brasil   201012.0    20.34
3 2010-12-01 08:26:00          2.75     17850.0  Brasil   201012.0    22.00
4 2010-12-01 08:26:00          3.39     17850.0  Brasil   201012.0    20.34

    MinorPurchaseDate  MonthMinorPurchaseMonthly
0 2010-12-01 08:26:00                     201012
1 2010-12-01 08:26:00                     201012
2 2010-12-01 08:26:00                     201012
3 2010-12-01 08:26:00                     201012
4 2010-12-01 08:26:00                     201012
```

[41]:
```
# Let's create a new user type column and fill it in as New
purchase_data['UserType'] = 'New'
```

[42]:
```
# Data
purchase_data['UserType'].value_counts()
```

[42]:
```
New    403078
Name: UserType, dtype: int64
```

[43]:
```
# Data
purchase_data.head()
```

[43]:
```
  BillNum ProductCode                          ProductName  Quantity  \
0  536365       21730    GLASS STAR FROSTED T-LIGHT HOLDER       6.0
1  536365      85123A   WHITE HANGING HEART T-LIGHT HOLDER       6.0
2  536365       71053                  WHITE METAL LANTERN       6.0
3  536365      84406B       CREAM CUPID HEARTS COAT HANGER       8.0
4  536365      84029G  KNITTED UNION FLAG HOT WATER BOTTLE       6.0

             SaleDate  UnitaryValue  CustomerID Country  YearMonth  Revenue  \
0 2010-12-01 08:26:00          4.25     17850.0  Brasil   201012.0    25.50
1 2010-12-01 08:26:00          2.55     17850.0  Brasil   201012.0    15.30
2 2010-12-01 08:26:00          3.39     17850.0  Brasil   201012.0    20.34
3 2010-12-01 08:26:00          2.75     17850.0  Brasil   201012.0    22.00
```

```
4 2010-12-01 08:26:00          3.39     17850.0  Brasil   201012.0    20.34

   MinorPurchaseDate  MonthMinorPurchaseMonthly UserType
0 2010-12-01 08:26:00                    201012      New
1 2010-12-01 08:26:00                    201012      New
2 2010-12-01 08:26:00                    201012      New
3 2010-12-01 08:26:00                    201012      New
4 2010-12-01 08:26:00                    201012      New
```

[44]:

```
# An old customer is one whose purchase volume in the month is greater than the␣
 ↪minimum volume
# If it's true, we change the UserType column to "Old" and if not, we keep it␣
 ↪as "New"
purchase_data.loc[purchase_data['YearMonth'] >␣
 ↪purchase_data['MonthMinorPurchaseMonthly'], 'UserType'] = 'Old'
```

[45]:

```
# Data
purchase_data['UserType'].value_counts()
```

[45]:

```
Old    285016
New    118062
Name: UserType, dtype: int64
```

[46]:

```
# Now we calculate billing by type of user per month
df_month_user_revenue = purchase_data.groupby(['YearMonth',␣
 ↪'UserType'])['Revenue'].sum().reset_index()
```

[47]:

```
# Removed month 12 of 2011 as we don't have enough data
df_month_user_revenue = df_month_user_revenue.query("YearMonth != 201012 and␣
 ↪YearMonth != 201112")
```

[48]:

```
# Data
df_month_user_revenue
```

[48]:

```
    YearMonth UserType     Revenue
1    201101.0      New  201769.790
2    201101.0      Old  268917.980
3    201102.0      New  148578.380
4    201102.0      Old  283505.490
5    201103.0      New  188647.400
6    201103.0      Old  387223.030
7    201104.0      New  118691.331
8    201104.0      Old  303130.130
9    201105.0      New  115388.890
10   201105.0      Old  527764.450
11   201106.0      New   91922.430
12   201106.0      Old  512031.920
```

```
13    201107.0      New    65516.061
14    201107.0      Old   505029.670
15    201108.0      New    76924.070
16    201108.0      Old   536541.270
17    201109.0      New   153000.971
18    201109.0      Old   773825.361
19    201110.0      New   154288.820
20    201110.0      Old   816700.360
21    201111.0      New   133540.980
22    201111.0      Old   993843.450
```

1.3.12 Visualization of Indicator 6

[49]:

```
# Plot

# Definition of data in the plot
plot_data = [go.Scatter(x = df_month_user_revenue.query("UserType ==␣
 ↪'Old'")['YearMonth'],
                        y = df_month_user_revenue.query("UserType ==␣
 ↪'Old'")['Revenue'],
                        name = 'Old Customer'),
             go.Scatter(x = df_month_user_revenue.query("UserType ==␣
 ↪'New'")['YearMonth'],
                        y = df_month_user_revenue.query("UserType ==␣
 ↪'New'")['Revenue'],
                        name = 'New Customer')]

# Layout
plot_layout = go.Layout(xaxis = {"type": "category"},
                        title = 'Difference in Revenue Over Time Between New␣
 ↪and Old Customers')

# Figure Plot
fig = go.Figure(data = plot_data, layout = plot_layout)
pyoff.iplot(fig)
```

1.3.13 Indicator 7 - Rate of New Clients

Since we defined new and old customers in indicator 6, we can now use the data and calculate the proportion of new customers over time.

[50]:

```
# Calcula a taxa de novos clientes
df_new_customers_rate = purchase_data.query("UserType == 'New'").
 ↪groupby(['YearMonth'])['CustomerID'].nunique() / purchase_data.
 ↪query("UserType == 'Old'").groupby(['YearMonth'])['CustomerID'].nunique()
```

[51]:

```
# Adjust index and remove missing values
df_new_customers_rate = df_new_customers_rate.reset_index()
df_new_customers_rate = df_new_customers_rate.dropna()
```

[52]:

```
# Data
df_new_customers_rate
```

[52]:

```
    YearMonth  CustomerID
1    201101.0    1.162983
2    201102.0    0.909091
3    201103.0    0.756897
4    201104.0    0.498333
5    201105.0    0.348750
6    201106.0    0.287990
7    201107.0    0.238155
8    201108.0    0.205665
9    201109.0    0.297109
10   201110.0    0.328052
11   201111.0    0.230935
12   201112.0    0.064163
```

1.3.14 Visualization of Indicator 7

[53]:

```
# Plot

# Definition of data in the plot
plot_data = [go.Bar(x = df_new_customers_rate.query("YearMonth > 201101 and␣
 ↪YearMonth < 201112")['YearMonth'],
                    y = df_new_customers_rate.query("YearMonth > 201101 and␣
 ↪YearMonth < 201112")['CustomerID'],)]

# Layout
plot_layout = go.Layout(xaxis = {"type": "category"},
                        title = 'New Customer Rate')

# Figure plot
fig = go.Figure(data = plot_data, layout = plot_layout)
pyoff.iplot(fig)
```

1.3.15 Indicator 8 - Monthly Customer Retention Rate

Monthly Customer Retention Rate = Previous Month's Customers / Total Active Customers

[54]:

```
# We group the data by customer and month and add the billing
customer_purchase_data = purchase_data.groupby(['CustomerID',␣
 ↪'YearMonth'])['Revenue'].sum().reset_index()
```

[55]:
```
# Data
customer_purchase_data.head()
```

[55]:
```
   CustomerID  YearMonth  Revenue
0     12346.0   201101.0     0.00
1     12347.0   201012.0   711.79
2     12347.0   201101.0   475.39
3     12347.0   201104.0   636.25
4     12347.0   201106.0   372.32
```

[56]:
```
# Now we define the retention with a cross table
df_ret = pd.crosstab(customer_purchase_data['CustomerID'],␣
 ↪customer_purchase_data['YearMonth']).reset_index()
```

[57]:
```
# Data
df_ret.head()
```

[57]:
```
YearMonth  CustomerID  201012.0  201101.0  201102.0  201103.0  201104.0  \
0             12346.0         0         1         0         0         0
1             12347.0         1         1         0         0         1
2             12348.0         1         1         0         0         1
3             12349.0         0         0         0         0         0
4             12350.0         0         0         1         0         0

YearMonth  201105.0  201106.0  201107.0  201108.0  201109.0  201110.0  \
0                 0         0         0         0         0         0
1                 0         1         0         1         0         1
2                 0         0         0         0         1         0
3                 0         0         0         0         0         0
4                 0         0         0         0         0         0

YearMonth  201111.0  201112.0
0                 0         0
1                 0         1
2                 0         0
3                 1         0
4                 0         0
```

[58]:
```
# Extract the months
months = df_ret.columns[2:]
months
```

[58]:
```
Index([201101.0, 201102.0, 201103.0, 201104.0, 201105.0, 201106.0, 201107.0,
       201108.0, 201109.0, 201110.0, 201111.0, 201112.0],
      dtype='object', name='YearMonth')
```

[59]:

```
# This loop will calculate the retention over the months

# List to save the result
ret_list = []

# Loop
for i in range(len(months)-1):
    retention_data = {}
    current_month = months[i+1]
    last_month = months[i]
    retention_data['YearMonth'] = int(current_month)
    retention_data['TotalUser'] = df_ret[current_month].sum()
    retention_data['TotalRetention'] = df_ret[(df_ret[current_month] > 0) &␣
 ↪(df_ret[last_month] > 0)][current_month].sum()
    ret_list.append(retention_data)

ret_list
```

[59]:

```
[{'YearMonth': 201102, 'TotalUser': 798, 'TotalRetention': 299},
 {'YearMonth': 201103, 'TotalUser': 1019, 'TotalRetention': 345},
 {'YearMonth': 201104, 'TotalUser': 899, 'TotalRetention': 346},
 {'YearMonth': 201105, 'TotalUser': 1079, 'TotalRetention': 399},
 {'YearMonth': 201106, 'TotalUser': 1051, 'TotalRetention': 464},
 {'YearMonth': 201107, 'TotalUser': 993, 'TotalRetention': 415},
 {'YearMonth': 201108, 'TotalUser': 979, 'TotalRetention': 433},
 {'YearMonth': 201109, 'TotalUser': 1301, 'TotalRetention': 465},
 {'YearMonth': 201110, 'TotalUser': 1425, 'TotalRetention': 552},
 {'YearMonth': 201111, 'TotalUser': 1711, 'TotalRetention': 690},
 {'YearMonth': 201112, 'TotalUser': 680, 'TotalRetention': 439}]
```

[60]:

```
# Dados
df_ret_final = pd.DataFrame(ret_list)
df_ret_final.head()
```

[60]:

```
   YearMonth  TotalUser  TotalRetention
0     201102        798             299
1     201103       1019             345
2     201104        899             346
3     201105       1079             399
4     201106       1051             464
```

Now we calculate the ratio to find the indicator.

[61]:

```
# Calculate the indicator
df_ret_final['RetentionRate'] = df_ret_final['TotalRetention'] /␣
 ↪df_ret_final['TotalUser']
df_ret_final
```

```
[61]:     YearMonth  TotalUser  TotalRetention  RetentionRate
      0      201102        798             299       0.374687
      1      201103       1019             345       0.338567
      2      201104        899             346       0.384872
      3      201105       1079             399       0.369787
      4      201106       1051             464       0.441484
      5      201107        993             415       0.417925
      6      201108        979             433       0.442288
      7      201109       1301             465       0.357417
      8      201110       1425             552       0.387368
      9      201111       1711             690       0.403273
      10     201112        680             439       0.645588
```

1.3.16 Visualization of Indicator 8

```
[62]: # Plot

      # Definition of data in the plot
      plot_data = [go.Scatter(x = df_ret_final.query("YearMonth <␣
        ↪201112")['YearMonth'],
                              y = df_ret_final.query("YearMonth <␣
        ↪201112")['RetentionRate'],
                              name = "taxa")]

      # Layout
      plot_layout = go.Layout(xaxis = {"type": "category"},
                              title = 'Monthly Customer Retention Rate')

      # Figure plot
      fig = go.Figure(data = plot_data, layout = plot_layout)
      pyoff.iplot(fig)
```

2 End