Chapter 2 - Ex4: Customer Lifetime Value

- Link tham khảo: Hand-on Data Science for Marketing

Customer Lifetime Value

Wikipedia (https://en.wikipedia.org/wiki/Customer_lifetime_value):

- "In marketing, customer lifetime value (CLV or often CLTV), lifetime customer value (LCV), or life-time value (LTV) is a prediction of the net profit attributed to the entire future relationship with a customer. The prediction model can have varying levels of sophistication and accuracy, ranging from a crude heuristic to the use of complex predictive analytics techniques.
- Customer lifetime value can also be defined as the monetary value of a customer relationship, based on the present value of the projected future cash flows from the customer relationship.citation needed (https://en.wikipedia.org/wiki/Wikipedia:Citation_needed) Customer lifetime value is an important concept in that it encourages firms to shift their focus from quarterly profits to the long-term health of their customer relationships. Customer lifetime value is an important metric because it represents an upper limit on spending to acquire new customers. For this reason it is an important element in calculating payback of advertising spent in marketing mix modeling."

"Customer Lifetime Value (CLTV) - Giá trị trọn đời của khách hàng là giá trị tiền tệ đại diện cho số tiền doanh thu hoặc lợi nhuận mà khách hàng sẽ mang lại cho công ty trong suốt thời gian của mối quan hệ". CLTV cho thấy ý nghĩa của việc có được khách hàng dài hạn so với khách hàng ngắn hạn. Giá trị trọn đời của khách hàng (CLV) có thể giúp trả lời các câu hỏi quan trọng nhất về bán hàng cho mọi công ty:

- Làm thế nào để xác định khách hàng có lợi nhuận cao nhất?
- Làm thế nào một công ty có thể cung cấp sản phẩm tốt nhất và kiếm được nhiều tiền nhất?
- Làm thế nào để phân khúc khách hàng có lợi nhuận?
- Cần bao nhiêu ngân sách để có được khách hàng?

Trong marketing, CLV là một trong những số liệu chính cần có và theo dõi.

- Nhìn chung, để có được khách hàng mới tốn kém hơn là giữ khách hàng hiện tại, vì vậy biết giá trị trọn đời và chi phí liên quan đến việc có được khách hàng mới là điều cần thiết để xây dựng chiến lược tiếp thị với positive ROI tích cực.

Ví dụ:

- Nếu CLV trung bình của khách hàng là 100 usd và và chỉ tốn 10 usd để có được một khách hàng mới, thì doanh nghiệp sẽ tạo thêm doanh thu khi có được khách hàng mới.
- Tuy nhiên, nếu chi phí 150 usd để có được một khách hàng mới và CLV trung bình của khách hàng vẫn là 100 usd, thì sẽ mất tiền cho mỗi lần có một khách hàng mới. Như vậy, nếu chi tiêu tiếp thị cho việc có thêm một khách hàng mới vượt quá CLV, ta sẽ mất tiền cho mỗi lần có

thêm một khách hàng => Trường hợp này, tốt hơn là nên làm việc với các khách hàng hiện tại.

Có nhiều cách để tính CLV

- Một cách là tìm số tiền mua trung bình của khách hàng (customer's average purchase amount), tần suất mua (purchase frequency) và tuổi thọ (lifetime span) và thực hiện một phép tính đơn giản để có được CLV.

Ví dụ:

- Số tiền mua trung bình của khách hàng là 100 usd, trung bình họ thực hiện mua hàng năm lần mỗi tháng => giá trị trung bình của khách hàng này mỗi tháng là 500 usd (**= customer's average purchase amount * purchase frequency**). Bây giờ, chúng ta cần biết tuổi thọ của khách hàng này (custumer's lifetime span).
- Một cách để ước tính tuổi thọ của khách hàng (customer's lifetime spa) là xem tỷ lệ bỏ đi trung bình hàng tháng (average monthly churn rate), là tỷ lệ phần trăm khách hàng bỏ đi hay chấm dứt mối quan hệ với doanh nghiệp. Có thể ước tính tuổi thọ của khách hàng bằng cách chia cho tỷ lệ bỏ đi (churn rate).
- Giả sử, với tỷ lệ bỏ đi là 5%, tuổi thọ ước tính của khách hàng là 20 năm. Nếu giá trị trung bình của khách hàng mỗi tháng là 500 usd, thì CLV của khách hàng này là 120.000 uds (= 500usd * 12 tháng * 20 năm)

Vì chúng ta thường không biết tuổi thọ của khách hàng (custumer's lifetime span), chúng ta sẽ cố gắng ước tính CLV trong một khoảng thời gian nhất định. Điều này có thể được thực hiện bằng cách ước tính CLV 12 tháng, 24 tháng hoặc cũng có thể là CLV 3 tháng của khách hàng.
Ngoài phương pháp đã nêu ở ví dụ trên, CLV cũng có thể được ước tính thông qua việc xây dựng các mô hình dự đoán. Sử dụng thuật toán Machine Learning và dữ liệu lịch sử mua hàng của khách hàng, chúng ta có thể xây dựng các mô hình Machine Learning dự đoán CLV của khách hàng trong một khoảng thời gian nhất định.

Trong bài tập này, cho dữ liệu OnlineRetail.xlsx download tại http://archive.ics.uci.edu/ml/datasets/online+retail (http://archive.ics.uci.edu/ml/datasets/online+retail) chứa dữ liệu lịch sử khách hàng, hãy xây dựng regression model dự đoán CLV 3 tháng tiếp theo của khách hàng.

Yêu cầu:

- Đoc dữ liệu. Tìm hiểu sơ bộ về dữ liệu.
- Tiền xử lý dữ liệu
- Phân tích dữ liệu
- Xây dựng model dự đoán
- Đánh giá model

In [1]:
```
%matplotlib inline
```

In [2]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

1. Load Data

In [3]:
```
df = pd.read_excel('Online Retail.xlsx', sheet_name='Online Retail')
```

In [4]:
```
df.shape
```

Out[4]:
```
(541909, 8)
```

In [5]:
```
df.head()
```

Out[5]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

2. Data Clean-Up

Handling negative quantity:

- Có các giao dịch với giá trị Quantity < 0, đại diện cho các đơn đặt hàng bị hủy => bỏ qua các đơn đặt hàng bị hủy.

In [6]:
```
df.loc[df['Quantity'] <= 0].shape
```

Out[6]:
```
(10624, 8)
```

In [7]:
```
df.shape
```

Out[7]:
```
(541909, 8)
```

In [8]:
```
df = df.loc[df['Quantity'] > 0]
```

In [9]:
```
df.shape
```

Out[9]:
```
(531285, 8)
```

Dropping NaN records

- Cần xóa bỏ những hồ sơ không có CustomerID. Vì chúng ta sẽ xây dựng một mô hình Machine Learning để dự đoán giá trị khách hàng 3 tháng nên cần nhóm dữ liệu theo cột CustomerID. Không có CustomerID, chúng ta không thể xây dựng mô hình.

In [10]:
```
pd.isnull(df['CustomerID']).sum()
```

Out[10]:
```
133361
```

In [11]:
```
df.shape
```

Out[11]:
```
(531285, 8)
```

In [12]:
```
df = df[pd.notnull(df['CustomerID'])]
```

In [13]:
```
df.shape
```

Out[13]:
```
(397924, 8)
```

In [14]:
```
df.head()
```

Out[14]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

Handling incomplete data

- Bộ dữ liệu chứa các giao dịch trong khoảng thời gian từ ngày 01/12/2010 => 09/12/2011. Trong đó, tháng 12/2011 chưa đủ dữ liệu. Để xây dựng mô hình đúng cho dự đoán giá trị

khách hàng 3 tháng, thì cần bỏ qua các giao dịch trong tháng cuối cùng.

In [15]:
```
print('Date Range: %s ~ %s' % (df['InvoiceDate'].min(),
                               df['InvoiceDate'].max()))
```

```
Date Range: 2010-12-01 08:26:00 ~ 2011-12-09 12:50:00
```

In [16]:
```
# Số giao dịch tính từ 01/12/2011 => sau
```

In [17]:
```
df.loc[df['InvoiceDate'] >= '2011-12-01'].shape
```

Out[17]:
```
(17304, 8)
```

In [18]:
```
df.shape
```

Out[18]:
```
(397924, 8)
```

In [19]:
```
# Số giao dịch trước 01/12/2011
```

In [20]:
```
df = df.loc[df['InvoiceDate'] < '2011-12-01']
```

In [21]:
```
df.shape
```

Out[21]:
```
(380620, 8)
```

Total Sales value

- Tạo cột 'Sales' (total purchase amount for each transaction) = 'Quantity' * 'UnitPrice'

In [22]:
```
df['Sales'] = df['Quantity'] * df['UnitPrice']
```

In [23]:
```
df.head()
```

Out[23]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Sale
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	15.3
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.3
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	22.0
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.3
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.3

Per Order Data

- Nhóm dữ liệu theo CustomerID và InvoiceNo

In [24]:
```
orders_df = df.groupby(['CustomerID', 'InvoiceNo']).agg({
    'Sales': sum,
    'InvoiceDate': max
})
```

In [25]:
```
orders_df
```

Out[25]:

CustomerID	InvoiceNo	Sales	InvoiceDate
12346.0	541431	77183.60	2011-01-18 10:01:00
	537626	711.79	2010-12-07 14:57:00
12347.0	542237	475.39	2011-01-26 14:30:00
	549222	636.25	2011-04-07 10:43:00
	556201	382.52	2011-06-09 13:01:00
...	...	...	...
18283.0	578262	313.65	2011-11-23 13:27:00
	579673	223.61	2011-11-30 12:59:00
	554065	765.28	2011-05-22 10:39:00
18287.0	570715	1001.32	2011-10-12 10:23:00
	573167	70.68	2011-10-28 09:29:00

3. Data Analysis

- Để tính toán CLV, cần biết purchase_frequency (tần suất mua hàng), purchase_duration (thời gian mua hàng), avg (tiền mua hàng trung bình), count (số lần mua hàng) của khách hàng.

In [26]:
```
def groupby_mean(x):
    return x.mean()

def groupby_count(x):
    return x.count()

def purchase_duration(x):
    return (x.max() - x.min()).days

def avg_frequency(x):
    return (x.max() - x.min()).days/x.count()

groupby_mean.__name__ = 'avg'
groupby_count.__name__ = 'count'
purchase_duration.__name__ = 'purchase_duration'
avg_frequency.__name__ = 'purchase_frequency'
```

In [27]:
```
summary_df = orders_df.reset_index().groupby('CustomerID').agg({
    'Sales': [min, max, sum, groupby_mean, groupby_count],
    'InvoiceDate': [min, max, purchase_duration, avg_frequency]
})
```

In [28]:
```
summary_df
```

Out[28]:

	Sales							
	min	max	sum	avg	count	min	max	purchase_du
CustomerID								
12346.0	77183.60	77183.60	77183.60	77183.600000	1.0	2011-01-18 10:01:00	2011-01-18 10:01:00	
12347.0	382.52	1294.32	4085.18	680.863333	6.0	2010-12-07 14:57:00	2011-10-31 12:25:00	
12348.0	227.44	892.80	1797.24	449.310000	4.0	2010-12-16 19:09:00	2011-09-25 13:13:00	
12349.0	1757.55	1757.55	1757.55	1757.550000	1.0	2011-11-21 09:51:00	2011-11-21 09:51:00	
						2011-	2011-	

In [29]:
```
summary_df.columns = ['_'.join(col).lower() for col in summary_df.columns]
```

In [30]:
```
summary_df
```

Out[30]:

CustomerID	sales_min	sales_max	sales_sum	sales_avg	sales_count	invoicedate_min	invoic
12346.0	77183.60	77183.60	77183.60	77183.600000	1.0	2011-01-18 10:01:00	
12347.0	382.52	1294.32	4085.18	680.863333	6.0	2010-12-07 14:57:00	
12348.0	227.44	892.80	1797.24	449.310000	4.0	2010-12-16 19:09:00	
12349.0	1757.55	1757.55	1757.55	1757.550000	1.0	2011-11-21 09:51:00	
12350.0	334.40	334.40	334.40	334.400000	1.0	2011-02-02 16:01:00	
...	...	...	...	...	...	...	
18280.0	180.60	180.60	180.60	180.600000	1.0	2011-03-07 09:52:00	
18281.0	80.82	80.82	80.82	80.820000	1.0	2011-06-12 10:53:00	
18282.0	100.21	100.21	100.21	100.210000	1.0	2011-08-05 13:35:00	
18283.0	1.95	313.65	1886.88	125.792000	15.0	2011-01-06 14:14:00	
18287.0	70.68	1001.32	1837.28	612.426667	3.0	2011-05-22 10:39:00	

4298 rows × 9 columns

In [31]:
```
summary_df.shape
```

Out[31]:
```
(4298, 9)
```

In [32]:
```
summary_df.describe()
```

Out[32]:

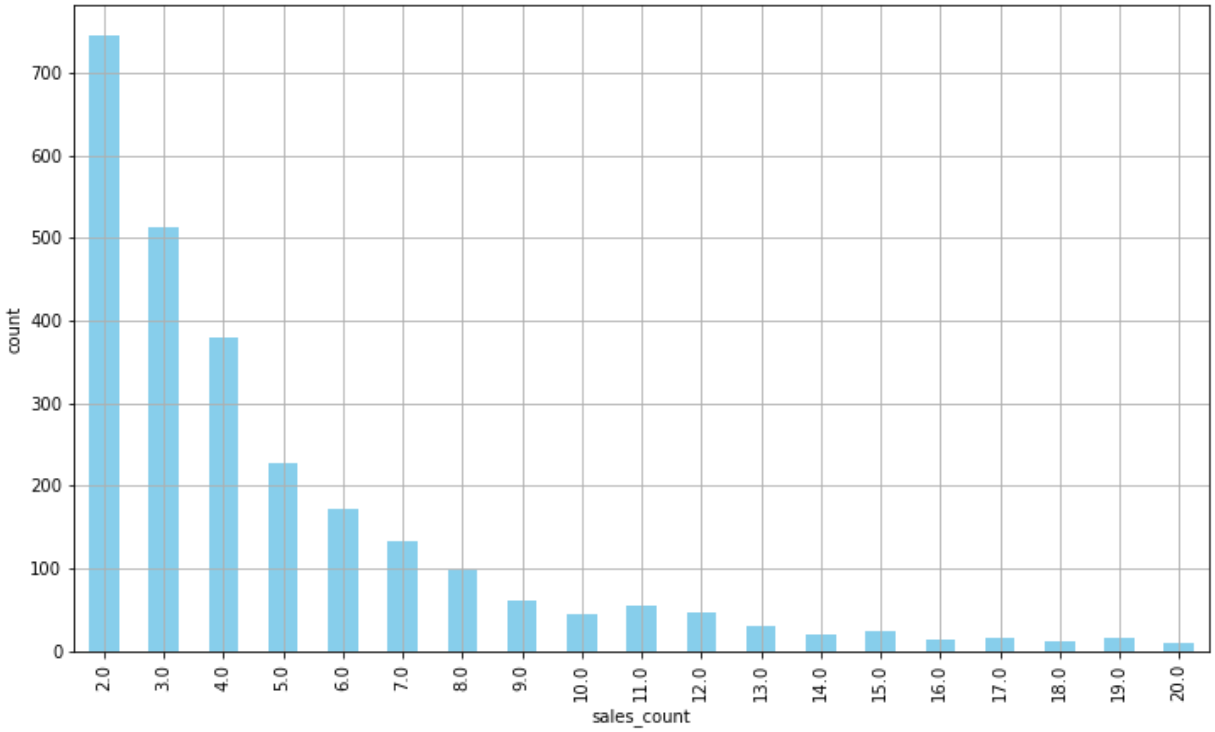
	sales_min	sales_max	sales_sum	sales_avg	sales_count	invoicedate_purchase_
count	4298.000000	4298.00000	4298.000000	4298.000000	4298.000000	429
mean	266.298816	613.89789	1952.818779	400.255621	4.131689	12
std	1219.631315	1747.66601	8354.913254	1271.187289	7.420253	12
min	0.000000	0.00000	0.000000	0.000000	1.000000	
25%	95.770000	223.36000	304.305000	178.602500	1.000000	
50%	172.175000	366.17000	657.265000	295.033958	2.000000	8
75%	310.722500	618.19250	1599.515000	431.594250	4.000000	24
max	77183.600000	77183.60000	268478.000000	77183.600000	201.000000	36

In [33]:
```
summary_df = summary_df.loc[summary_df['invoicedate_purchase_duration'] > 0]
```

In [34]:
```
summary_df.shape
```

Out[34]:
```
(2692, 9)
```

In [35]:
```
ax = summary_df.groupby('sales_count').count()['sales_avg'][:20].plot(
    kind='bar',
    color='skyblue',
    figsize=(12,7),
    grid=True
)

ax.set_ylabel('count')

plt.show()
```



- Biểu đồ cho thấy phần lớn khách hàng đã thực hiện 9 lần mua hoặc ít hơn trong lịch sử.

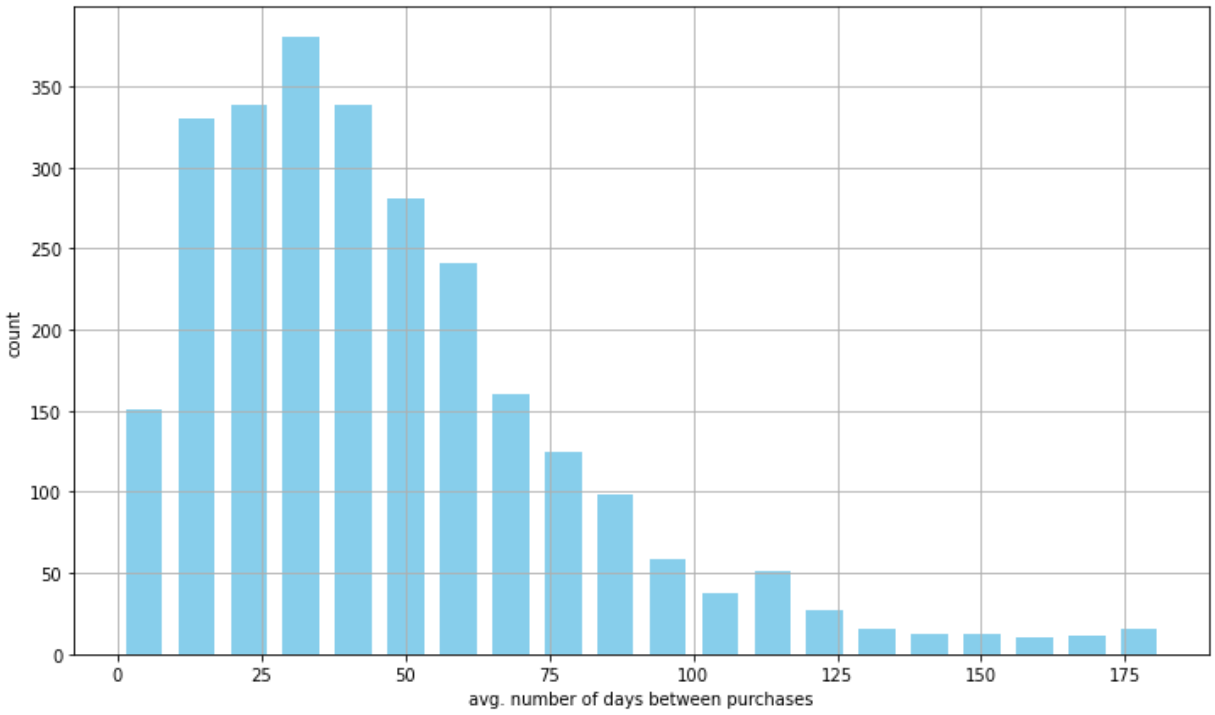
In [36]:
```
summary_df['sales_count'].describe()
```

Out[36]:
```
count    2692.000000
mean        5.971025
std         8.878128
min         2.000000
25%         2.000000
50%         4.000000
75%         6.000000
max       201.000000
Name: sales_count, dtype: float64
```

In [37]:
```
summary_df['sales_avg'].describe()
```

Out[37]:
```
count     2692.000000
mean       391.458687
std        465.584404
min          3.450000
25%        197.661000
50%        306.043333
75%        444.524000
max      14844.766667
Name: sales_avg, dtype: float64
```

In [38]:
```
ax = summary_df['invoicedate_purchase_frequency'].hist(
    bins=20,
    color='skyblue',
    rwidth=0.7,
    figsize=(12,7)
)

ax.set_xlabel('avg. number of days between purchases')
ax.set_ylabel('count')

plt.show()
```



- Biểu đồ cho chúng ta cái nhìn tổng thể về tần suất khách hàng lặp lại mua hàng trong lịch sử. Từ đó có thể thấy phần lớn khách hàng lặp lại việc mua hàng sau 20 đến 50 ngày.

In [39]:
```
summary_df['invoicedate_purchase_frequency'].describe()
```

Out[39]:
```
count    2692.000000
mean       46.999022
std        32.395004
min         0.029412
25%        23.500000
50%        40.500000
75%        62.333333
max       182.000000
Name: invoicedate_purchase_frequency, dtype: float64
```

In [40]:
```
summary_df['invoicedate_purchase_duration'].describe()
```

Out[40]:
```
count    2692.000000
mean      199.720282
std       107.816559
min         1.000000
25%       107.000000
50%       209.000000
75%       296.000000
max       364.000000
Name: invoicedate_purchase_duration, dtype: float64
```

4. Predicting 3-Month CLV

- Trong phần này, chúng ta sẽ xây dựng một mô hình dự đoán giá trị khách hàng 3 tháng. Trước tiên, cần cắt dữ liệu thành các khối trong 3 tháng và lấy dữ liệu của 3 tháng cuối làm target cho các dự đoán và phần còn lại làm feature.

4.1. Data Preparation

In [41]:
```
clv_freq = '3M'
```

- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Grouper.html (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Grouper.html)
- https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html#offset-aliases (https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html#offset-aliases)

In [42]:
```
data_df = orders_df.reset_index().groupby([
    'CustomerID',
    pd.Grouper(key='InvoiceDate', freq=clv_freq)
]).agg({
    'Sales': [sum, groupby_mean, groupby_count],
})
```

In [43]:
```
data_df.columns = ['_'.join(col).lower() for col in data_df.columns]
```

In [44]:
```
data_df = data_df.reset_index()
```

In [45]:
```
data_df.head(10)
```

Out[45]:

	CustomerID	InvoiceDate	sales_sum	sales_avg	sales_count
0	12346.0	2011-03-31	77183.60	77183.600	1.0
1	12347.0	2010-12-31	711.79	711.790	1.0
2	12347.0	2011-03-31	475.39	475.390	1.0
3	12347.0	2011-06-30	1018.77	509.385	2.0
4	12347.0	2011-09-30	584.91	584.910	1.0
5	12347.0	2011-12-31	1294.32	1294.320	1.0
6	12348.0	2010-12-31	892.80	892.800	1.0
7	12348.0	2011-03-31	227.44	227.440	1.0
8	12348.0	2011-06-30	367.00	367.000	1.0
9	12348.0	2011-09-30	310.00	310.000	1.0

- Mã hóa các giá trị ngày thành M_1, M_2, M_3, v.v., với số nhỏ hơn đại diện cho các ngày gần đây hơn. Ví dụ: ngày 2011-12-31 hiện được mã hóa thành M_1 và ngày 2011-09-30 hiện được mã hóa thành M_2.

In [46]:
```
date_month_map = {
    str(x)[:10]: 'M_%s' % (i+1) for i, x in enumerate(
        sorted(data_df.reset_index()['InvoiceDate'].unique(), reverse=True)
    )
}
```

In [47]:
```
data_df['M'] = data_df['InvoiceDate'].apply(lambda x: date_month_map[str(x)[:10]]
```

In [48]:
```
date_month_map
```

Out[48]:
```
{'2011-12-31': 'M_1',
 '2011-09-30': 'M_2',
 '2011-06-30': 'M_3',
 '2011-03-31': 'M_4',
 '2010-12-31': 'M_5'}
```

In [49]:
```
data_df.head(10)
```

Out[49]:

	CustomerID	InvoiceDate	sales_sum	sales_avg	sales_count	M
0	12346.0	2011-03-31	77183.60	77183.600	1.0	M_4
1	12347.0	2010-12-31	711.79	711.790	1.0	M_5
2	12347.0	2011-03-31	475.39	475.390	1.0	M_4
3	12347.0	2011-06-30	1018.77	509.385	2.0	M_3
4	12347.0	2011-09-30	584.91	584.910	1.0	M_2
5	12347.0	2011-12-31	1294.32	1294.320	1.0	M_1
6	12348.0	2010-12-31	892.80	892.800	1.0	M_5
7	12348.0	2011-03-31	227.44	227.440	1.0	M_4
8	12348.0	2011-06-30	367.00	367.000	1.0	M_3
9	12348.0	2011-09-30	310.00	310.000	1.0	M_2

- Building Sample Set

In [ ]:
```
```

In [50]:
```
features_df = pd.pivot_table(
    data_df.loc[data_df['M'] != 'M_1'],
    values=['sales_sum', 'sales_avg', 'sales_count'],
    columns='M',
    index='CustomerID'
)
```

In [51]:
```
features_df.columns = ['_'.join(col) for col in features_df.columns]
```

In [52]:
```
features_df.shape
```

Out[52]:
```
(3616, 12)
```

In [53]:
```
features_df.head(10)
```

Out[53]:

CustomerID	sales_avg_M_2	sales_avg_M_3	sales_avg_M_4	sales_avg_M_5	sales_count_M_2	sale
12346.0	NaN	NaN	77183.600	NaN	NaN	
12347.0	584.91	509.385	475.390	711.79	1.0	
12348.0	310.00	367.000	227.440	892.80	1.0	
12350.0	NaN	NaN	334.400	NaN	NaN	
12352.0	316.25	NaN	312.362	NaN	2.0	
12353.0	NaN	89.000	NaN	NaN	NaN	
12354.0	NaN	1079.400	NaN	NaN	NaN	
12355.0	NaN	459.400	NaN	NaN	NaN	
12356.0	NaN	481.460	2271.620	NaN	NaN	
12358.0	484.86	NaN	NaN	NaN	1.0	

In [54]:
```
features_df = features_df.fillna(0)
```

In [55]:
```
features_df.head()
```

Out[55]:

CustomerID	sales_avg_M_2	sales_avg_M_3	sales_avg_M_4	sales_avg_M_5	sales_count_M_2	sale
12346.0	0.00	0.000	77183.600	0.00	0.0	
12347.0	584.91	509.385	475.390	711.79	1.0	
12348.0	310.00	367.000	227.440	892.80	1.0	
12350.0	0.00	0.000	334.400	0.00	0.0	
12352.0	316.25	0.000	312.362	0.00	2.0	

In [56]:
```
response_df = data_df.loc[
    data_df['M'] == 'M_1',
    ['CustomerID', 'sales_sum']
]
```

In [57]:
```
response_df.columns = ['CustomerID', 'CLV_'+ clv_freq]
```

In [58]:
```
response_df.shape
```

Out[58]:
```
(2407, 2)
```

In [59]:
```
response_df.head(10)
```

Out[59]:

	CustomerID	CLV_3M
5	12347.0	1294.32
10	12349.0	1757.55
14	12352.0	311.73
20	12356.0	58.35
21	12357.0	6207.67
25	12359.0	2876.85
28	12360.0	1043.78
33	12362.0	2119.85
37	12364.0	299.06
41	12370.0	739.28

In [60]:
```
sample_set_df = features_df.merge(
    response_df,
    left_index=True,
    right_on='CustomerID',
    how='left'
)
```

In [61]:
```
sample_set_df.shape
```

Out[61]:
```
(3616, 14)
```

In [62]:
```
sample_set_df.head(10)
```

Out[62]:

	sales_avg_M_2	sales_avg_M_3	sales_avg_M_4	sales_avg_M_5	sales_count_M_2	sales_coun
NaN	0.00	0.000	77183.600	0.00	0.0	
5.0	584.91	509.385	475.390	711.79	1.0	
NaN	310.00	367.000	227.440	892.80	1.0	
NaN	0.00	0.000	334.400	0.00	0.0	
14.0	316.25	0.000	312.362	0.00	2.0	
NaN	0.00	89.000	0.000	0.00	0.0	
NaN	0.00	1079.400	0.000	0.00	0.0	
NaN	0.00	459.400	0.000	0.00	0.0	
20.0	0.00	481.460	2271.620	0.00	0.0	
NaN	484.86	0.000	0.000	0.00	1.0	

In [63]:
```
sample_set_df = sample_set_df.fillna(0)
```

In [64]:
```
sample_set_df.head()
```

Out[64]:

	sales_avg_M_2	sales_avg_M_3	sales_avg_M_4	sales_avg_M_5	sales_count_M_2	sales_coun
NaN	0.00	0.000	77183.600	0.00	0.0	
5.0	584.91	509.385	475.390	711.79	1.0	
NaN	310.00	367.000	227.440	892.80	1.0	
NaN	0.00	0.000	334.400	0.00	0.0	
14.0	316.25	0.000	312.362	0.00	2.0	

In [65]:
```
sample_set_df['CLV_'+clv_freq].describe()
```

Out[65]:
```
count     3616.000000
mean       511.558520
std       2371.743293
min          0.000000
25%          0.000000
50%          0.000000
75%        458.662500
max      68012.350000
Name: CLV_3M, dtype: float64
```

4.2. Regression Models

In [66]:
```
from sklearn.model_selection import train_test_split
```

In [67]:
```
target_var = 'CLV_'+clv_freq
all_features = [x for x in sample_set_df.columns if x not in ['CustomerID', targ
```

In [68]:
```
x_train, x_test, y_train, y_test = train_test_split(
    sample_set_df[all_features],
    sample_set_df[target_var],
    test_size=0.3
)
```

- Linear Regression Model

In [69]:
```
from sklearn.linear_model import LinearRegression

# Try these models as well
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
```

In [70]:
```
reg_fit = LinearRegression()
```

In [71]:
```
reg_fit.fit(x_train, y_train)
```

Out[71]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [72]:
```
reg_fit.intercept_
```

Out[72]:
```
79.7513574255185
```

In [73]:
```
coef = pd.DataFrame(list(zip(all_features, reg_fit.coef_)))
coef.columns = ['feature', 'coef']

coef
```

Out[73]:

	feature	coef
0	sales_avg_M_2	0.257082
1	sales_avg_M_3	-0.413658
2	sales_avg_M_4	-0.441985
3	sales_avg_M_5	-0.275130
4	sales_count_M_2	79.703381
5	sales_count_M_3	52.589856
6	sales_count_M_4	-163.307863
7	sales_count_M_5	119.970460
8	sales_sum_M_2	0.129634
9	sales_sum_M_3	0.265313
10	sales_sum_M_4	0.457662
11	sales_sum_M_5	0.497540

4.3. Evaluation

In [74]:
```
from sklearn.metrics import r2_score, median_absolute_error
```

In [75]:
```
train_preds =  reg_fit.predict(x_train)
test_preds = reg_fit.predict(x_test)
```

- R-Squared

In [76]:
```
print('In-Sample R-Squared: %0.4f' % r2_score(y_true=y_train,
                                              y_pred=train_preds))
print('Out-of-Sample R-Squared: %0.4f' % r2_score(y_true=y_test,
                                                  y_pred=test_preds))
```

```
In-Sample R-Squared: 0.7810
Out-of-Sample R-Squared: 0.6168
```

- Median Absolute Error

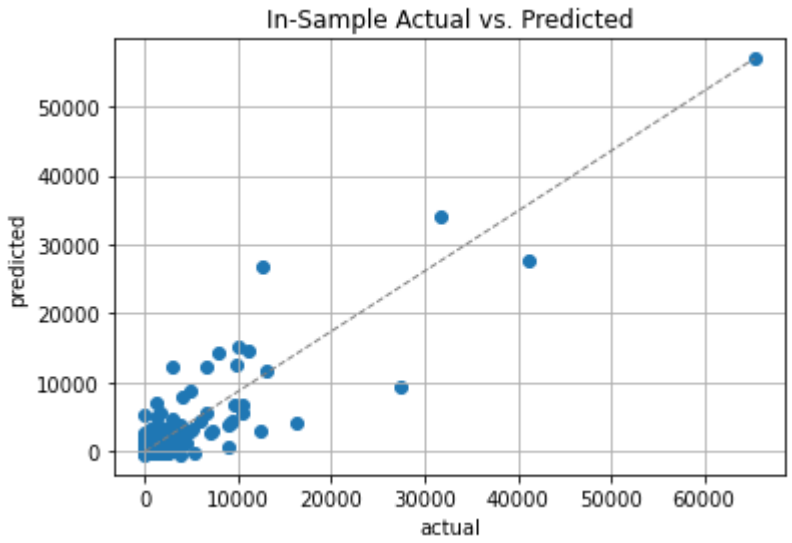
- Median Absolute Error

In [77]:
```
print('In-Sample MSE: %0.4f' % median_absolute_error(y_true=y_train,
                                                     y_pred=train_preds))
print('Out-of-Sample MSE: %0.4f' % median_absolute_error(y_true=y_test,
                                                         y_pred=test_preds))
```

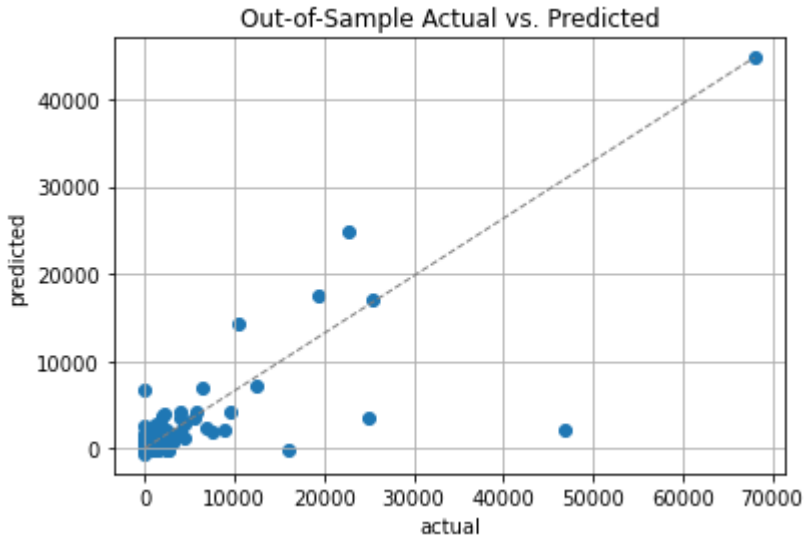
```
In-Sample MSE: 232.9884
Out-of-Sample MSE: 222.7988
```

- Scatter Plot

In [78]:
```
plt.scatter(y_train, train_preds)
plt.plot([0, max(y_train)], [0, max(train_preds)],
         color='gray',
         lw=1, linestyle='--')

plt.xlabel('actual')
plt.ylabel('predicted')
plt.title('In-Sample Actual vs. Predicted')
plt.grid()

plt.show()
```



In [79]:
```
plt.scatter(y_test, test_preds)
plt.plot([0, max(y_test)], [0, max(test_preds)],
         color='gray',
         lw=1, linestyle='--')

plt.xlabel('actual')
plt.ylabel('predicted')
plt.title('Out-of-Sample Actual vs. Predicted')
plt.grid()

plt.show()
```



In [ ]:
```
```