**Customer Lifetime Value**

In this tutorial, learn how to calculate Customer Lifetime Value in Python.

<https://www.datacamp.com/community/tutorials/customer-life-time-value>

Italian economist Vilfredo Pareto states that 80% of the effect comes from 20% of the causes, this is known as 80/20 rule or Pareto principle. Similarly, 80% of companies business comes from 20% customers. Companies need to identify those top customers and maintain the relationship with them to ensure continuous revenue. In order to maintain a long-term relationship with customers, companies need to schedule loyalty schemes such as the discount, offers, coupons, bonus point, and gifts.

Targeting a new customer is more costly than retaining existing customers because you don’t need to spend resources, time, and work hard to acquire new customers. You just have to keep the existing customers happy. Business analyst's accurately calculate customer acquisition cost using CLTV(Customer Lifetime Value). CLTV indicates the total revenue from the customer during the entire relationship. CLTV helps companies to focus on those potential customers who can bring in the more revenue in the future.

In this tutorial, you are going to cover the following topics:

* Introduction
* Customer Lifetime value(CLTV)
* Related Work of CLTV
* CLTV Formulas
* Implementing CLTV in Python
* Prediction model for CLTV
* Pros and Cons
* Conclusion

**Customer Lifetime Value(CLTV)**

"Customer Lifetime Value is a monetary value that represents the amount of revenue or profit a customer will give the company over the period of the relationship" ([Source](https://towardsdatascience.com/predictive-customer-analytics-part-iii-aeb996beafba)). CLTV demonstrates the implications of acquiring long-term customers compare to short-term customers. Customer lifetime value (CLV) can help you to answers the most important questions about sales to every company:

* How to Identify the most profitable customers?
* How can a company offer the best product and make the most money?
* How to segment profitable customers?
* How much budget need to spend to acquire customers?

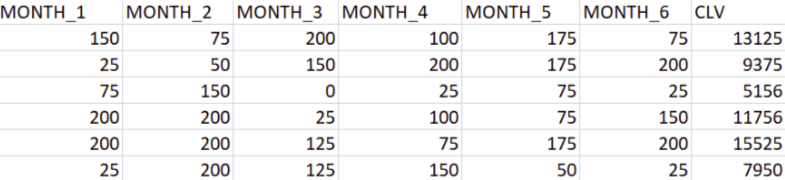
**Calculate Customer Lifetime Value**

There are lots of approaches available for calculating CLTV. Everyone has his/her own view on it. For computing CLTV we need historical data of customers but you will unable to calculate for new customers. To solve this problem Business Analyst develops machine learning models to predict the CLTV of newly customers. Let's explore some approaches for CLTV Calculation:

1) You can compute it by adding profit/revenue from customers in a given cycle. For Example, If the customer is associated with you for the last 3 years, you can sum all the profit in this 3 years. You can average the profit yearly or half-yearly or monthly, but in this approach, you cannot able to build a predictive model for new customers.

2) Build a regression model for existing customers. Take recent six-month data as independent variables and total revenue over three years as a dependent variable and build a regression model on this data.





3) CLTV can also implement using RFM(Recency, Frequency, Monetary) values. For more details, you can refer to [my tutorial](https://www.datacamp.com/community/tutorials/introduction-customer-segmentation-python).

4) **Using the following equation:** CLTV = ((Average Order Value x Purchase Frequency)/Churn Rate) x Profit margin.

Customer Value = Average Order Value \* Purchase Frequency

* **Average Order Value(AOV):** The Average Order value is the ratio of your total revenue and the total number of orders. AOV represents the mean amount of revenue that the customer spends on an order.
* Average Order Value = Total Revenue / Total Number of Orders
* **Purchase Frequency(PF):** Purchase Frequency is the ratio of the total number of orders and the total number of customer. It represents the average number of orders placed by each customer.
* Purchase Frequency = Total Number of Orders / Total Number of Customers
* **Churn Rate:** Churn Rate is the percentage of customers who have not ordered again.
* **Customer Lifetime:** Customer Lifetime is the period of time that the customer has been continuously ordering.
* Customer Lifetime=1/Churn Rate
* **Repeat Rate:** Repeat rate can be defined as the ratio of the number of customers with more than one order to the number of unique customers. Example: If you have 10 customers in a month out of who 4 come back, your repeat rate is 40%.
* Churn Rate= 1-Repeat Rate

**CLTV Implementation in Python(Using Formula)**

**Importing Required Library**

#import modules

import pandas as pd # for dataframes

import matplotlib.pyplot as plt # for plotting graphs

import seaborn as sns # for plotting graphs

import datetime as dt

import numpy as np

**Loading Dataset**

Let's first load the required Online Retail dataset using the pandas read CSV function. You can download the data from [here](http://archive.ics.uci.edu/ml/datasets/online+retail).

data = pd.read\_excel("Online\_Retail.xlsx")

data.head()

|  | **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 |

**Removing Duplicates**

Sometimes you get a messy dataset. You may have to deal with duplicates, which will skew your analysis. In python, pandas offer function drop\_duplicates(), which drops the repeated or duplicate records.

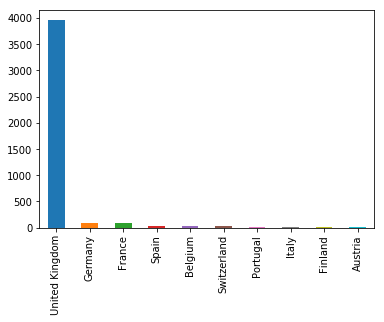
filtered\_data=data[['Country','CustomerID']].drop\_duplicates()

**Let's Jump into Data Insights**

#Top ten country's customer

filtered\_data.Country.value\_counts()[:10].plot(kind='bar')

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fe677a887f0>



In the given dataset, you can observe most of the customers are from "United Kingdom". So, you can filter data for United Kingdom customer.

uk\_data=data[data.Country=='United Kingdom']

uk\_data.info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 495478 entries, 0 to 541893

Data columns (total 8 columns):

InvoiceNo 495478 non-null object

StockCode 495478 non-null object

Description 494024 non-null object

Quantity 495478 non-null int64

InvoiceDate 495478 non-null datetime64[ns]

UnitPrice 495478 non-null float64

CustomerID 361878 non-null float64

Country 495478 non-null object

dtypes: datetime64[ns](1), float64(2), int64(1), object(4)

memory usage: 34.0+ MB

The describe() function in pandas is convenient in getting various summary statistics. This function returns the count, mean, standard deviation, minimum and maximum values and the quantiles of the data.

uk\_data.describe()

|  | **Quantity** | **UnitPrice** | **CustomerID** |
| --- | --- | --- | --- |
| **count** | 495478.000000 | 495478.000000 | 361878.000000 |
| **mean** | 8.605486 | 4.532422 | 15547.871368 |
| **std** | 227.588756 | 99.315438 | 1594.402590 |
| **min** | -80995.000000 | -11062.060000 | 12346.000000 |
| **25%** | 1.000000 | 1.250000 | 14194.000000 |
| **50%** | 3.000000 | 2.100000 | 15514.000000 |
| **75%** | 10.000000 | 4.130000 | 16931.000000 |
| **max** | 80995.000000 | 38970.000000 | 18287.000000 |

Here, you can observe some of the customers have ordered in a negative quantity, which is not possible. So, you need to filter Quantity greater than zero.

uk\_data = uk\_data[(uk\_data['Quantity']>0)]

uk\_data.info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 486286 entries, 0 to 541893

Data columns (total 8 columns):

InvoiceNo 486286 non-null object

StockCode 486286 non-null object

Description 485694 non-null object

Quantity 486286 non-null int64

InvoiceDate 486286 non-null datetime64[ns]

UnitPrice 486286 non-null float64

CustomerID 354345 non-null float64

Country 486286 non-null object

dtypes: datetime64[ns](1), float64(2), int64(1), object(4)

memory usage: 33.4+ MB

**Filter required Columns**

Here, you can filter the necessary columns for calculating CLTV. You only need her five columns CustomerID, InvoiceDate, InvoiceNo, Quantity, and UnitPrice.

* **CustomerID** will uniquely define your customers.
* **InvoiceDate** help you calculate numbers of days customer stayed with your product.
* **InvoiceNo** helps you to count the number of time transaction performed(frequency).
* **Quantity** is purchased item units in each transaction
* **UnitPrice** of each unit purchased by the customer will help you to calculate the total purchased amount.

uk\_data=uk\_data[['CustomerID','InvoiceDate','InvoiceNo','Quantity','UnitPrice']]

#Calulate total purchase

uk\_data['TotalPurchase'] = uk\_data['Quantity'] \* uk\_data['UnitPrice']

Here, you are going to perform the following operations:

* Calculate the number of days between the present date and the date of last purchase from each customer.
* Calculate the number of orders for each customer.
* Calculate sum of purchase price for each customer.

uk\_data\_group=uk\_data.groupby('CustomerID').agg({'InvoiceDate': lambda date: (date.max() - date.min()).days,

'InvoiceNo': lambda num: len(num),

'Quantity': lambda quant: quant.sum(),

'TotalPurchase': lambda price: price.sum()})

uk\_data\_group.head()

|  | **InvoiceDate** | **InvoiceNo** | **Quantity** | **TotalPurchase** |
| --- | --- | --- | --- | --- |
| **CustomerID** |  |  |  |  |
| **12346.0** | 0 | 1 | 74215 | 77183.60 |
| **12747.0** | 366 | 103 | 1275 | 4196.01 |
| **12748.0** | 372 | 4596 | 25748 | 33719.73 |
| **12749.0** | 209 | 199 | 1471 | 4090.88 |
| **12820.0** | 323 | 59 | 722 | 942.34 |

**Rename the column**

# Change the name of columns

uk\_data\_group.columns=['num\_days','num\_transactions','num\_units','spent\_money']

uk\_data\_group.head()

|  | **num\_days** | **num\_transactions** | **num\_units** | **spent\_money** |
| --- | --- | --- | --- | --- |
| **CustomerID** |  |  |  |  |
| **12346.0** | 0 | 1 | 74215 | 77183.60 |
| **12747.0** | 366 | 103 | 1275 | 4196.01 |
| **12748.0** | 372 | 4596 | 25748 | 33719.73 |
| **12749.0** | 209 | 199 | 1471 | 4090.88 |
| **12820.0** | 323 | 59 | 722 | 942.34 |

**Calculate CLTV using following formula:**

CLTV = ((Average Order Value x Purchase Frequency)/Churn Rate) x Profit margin.

Customer Value = Average Order Value \* Purchase Frequency

**1. Calculate Average Order Value**

# Average Order Value

uk\_data\_group['avg\_order\_value']=uk\_data\_group['spent\_money']/uk\_data\_group['num\_transactions']

uk\_data\_group.head()

|  | **num\_days** | **num\_transactions** | **num\_units** | **spent\_money** | **avg\_order\_value** |
| --- | --- | --- | --- | --- | --- |
| **CustomerID** |  |  |  |  |  |
| **12346.0** | 0 | 1 | 74215 | 77183.60 | 77183.600000 |
| **12747.0** | 366 | 103 | 1275 | 4196.01 | 40.737961 |
| **12748.0** | 372 | 4596 | 25748 | 33719.73 | 7.336756 |
| **12749.0** | 209 | 199 | 1471 | 4090.88 | 20.557186 |
| **12820.0** | 323 | 59 | 722 | 942.34 | 15.971864 |

**2. Calculate Purchase Frequency**

purchase\_frequency=sum(uk\_data\_group['num\_transactions'])/uk\_data\_group.shape[0]

**3. Calculate Repeat Rate and Churn Rate**

# Repeat Rate

repeat\_rate=uk\_data\_group[uk\_data\_group.num\_transactions > 1].shape[0]/uk\_data\_group.shape[0]

#Churn Rate

churn\_rate=1-repeat\_rate

purchase\_frequency,repeat\_rate,churn\_rate

(90.37107880642694, 0.9818923743942872, 0.018107625605712774)

**4. Calculate Profit Margin**

Profit margin is the commonly used profitability ratio. It represents how much percentage of total sales has earned as the gain. Let's assume our business has approx 5% profit on the total sale.

# Profit Margin

uk\_data\_group['profit\_margin']=uk\_data\_group['spent\_money']\*0.05

uk\_data\_group.head()

|  | **num\_days** | **num\_transactions** | **num\_units** | **spent\_money** | **avg\_order\_value** | **profit\_margin** |
| --- | --- | --- | --- | --- | --- | --- |
| **CustomerID** |  |  |  |  |  |  |
| **12346.0** | 0 | 1 | 74215 | 77183.60 | 77183.600000 | 3859.1800 |
| **12747.0** | 366 | 103 | 1275 | 4196.01 | 40.737961 | 209.8005 |
| **12748.0** | 372 | 4596 | 25748 | 33719.73 | 7.336756 | 1685.9865 |
| **12749.0** | 209 | 199 | 1471 | 4090.88 | 20.557186 | 204.5440 |
| **12820.0** | 323 | 59 | 722 | 942.34 | 15.971864 | 47.1170 |

**5. Calcualte Customer Lifetime Value**

# Customer Value

uk\_data\_group['CLV']=(uk\_data\_group['avg\_order\_value']\*purchase\_frequency)/churn\_rate

#Customer Lifetime Value

uk\_data\_group['cust\_lifetime\_value']=uk\_data\_group['CLV']\*uk\_data\_group['profit\_margin']

uk\_data\_group.head()

|  | **num\_days** | **num\_transactions** | **num\_units** | **spent\_money** | **avg\_order\_value** | **profit\_margin** | **CLV** | **cust\_lifetime\_value** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CustomerID** |  |  |  |  |  |  |  |  |
| **12346.0** | 0 | 1 | 74215 | 77183.60 | 77183.600000 | 3859.1800 | 3.852060e+08 | 1.486579e+12 |
| **12747.0** | 366 | 103 | 1275 | 4196.01 | 40.737961 | 209.8005 | 2.033140e+05 | 4.265538e+07 |
| **12748.0** | 372 | 4596 | 25748 | 33719.73 | 7.336756 | 1685.9865 | 3.661610e+04 | 6.173424e+07 |
| **12749.0** | 209 | 199 | 1471 | 4090.88 | 20.557186 | 204.5440 | 1.025963e+05 | 2.098545e+07 |
| **12820.0** | 323 | 59 | 722 | 942.34 | 15.971864 | 47.1170 | 7.971198e+04 | 3.755789e+06 |

**Prediction Model for CLTV**

Let's build the CLTV prediction model.

Here, you are going to predict CLTV using Linear Regression Model.

Let's first use the data loaded and filtered above.

uk\_data.head()

|  | **CustomerID** | **InvoiceDate** | **InvoiceNo** | **Quantity** | **UnitPrice** | **TotalPurchase** | **month\_yr** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 17850.0 | 2010-12-01 08:26:00 | 536365 | 6 | 2.55 | 15.30 | Dec-2010 |
| **1** | 17850.0 | 2010-12-01 08:26:00 | 536365 | 6 | 3.39 | 20.34 | Dec-2010 |
| **2** | 17850.0 | 2010-12-01 08:26:00 | 536365 | 8 | 2.75 | 22.00 | Dec-2010 |
| **3** | 17850.0 | 2010-12-01 08:26:00 | 536365 | 6 | 3.39 | 20.34 | Dec-2010 |
| **4** | 17850.0 | 2010-12-01 08:26:00 | 536365 | 6 | 3.39 | 20.34 | Dec-2010 |

Extract month and year from InvoiceDate.

uk\_data['month\_yr'] = uk\_data['InvoiceDate'].apply(lambda x: x.strftime('%b-%Y'))

uk\_data.head()

|  | **CustomerID** | **InvoiceDate** | **InvoiceNo** | **Quantity** | **UnitPrice** | **TotalPurchase** | **month\_yr** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 17850.0 | 2010-12-01 08:26:00 | 536365 | 6 | 2.55 | 15.30 | Dec-2010 |
| **1** | 17850.0 | 2010-12-01 08:26:00 | 536365 | 6 | 3.39 | 20.34 | Dec-2010 |
| **2** | 17850.0 | 2010-12-01 08:26:00 | 536365 | 8 | 2.75 | 22.00 | Dec-2010 |
| **3** | 17850.0 | 2010-12-01 08:26:00 | 536365 | 6 | 3.39 | 20.34 | Dec-2010 |
| **4** | 17850.0 | 2010-12-01 08:26:00 | 536365 | 6 | 3.39 | 20.34 | Dec-2010 |

The pivot table takes the columns as input, and groups the entries into a two-dimensional table in such a way that provides a multidimensional summarization of the data.

sale=uk\_data.pivot\_table(index=['CustomerID'],columns=['month\_yr'],values='TotalPurchase',aggfunc='sum',fill\_value=0).reset\_index()

sale.head()

| **month\_yr** | **CustomerID** | **Apr-2011** | **Aug-2011** | **Dec-2010** | **Dec-2011** | **Feb-2011** | **Jan-2011** | **Jul-2011** | **Jun-2011** | **Mar-2011** | **May-2011** | **Nov-2011** | **Oct-2011** | **Sep-2011** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 12346.0 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 77183.60 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| **1** | 12747.0 | 0.00 | 301.70 | 706.27 | 438.50 | 0.00 | 303.04 | 0.00 | 376.30 | 310.78 | 771.31 | 312.73 | 675.38 | 0.00 |
| **2** | 12748.0 | 1100.37 | 898.24 | 4228.13 | 1070.27 | 389.64 | 418.77 | 1113.27 | 2006.26 | 1179.37 | 2234.50 | 10639.23 | 2292.84 | 6148.84 |
| **3** | 12749.0 | 0.00 | 1896.13 | 0.00 | 763.06 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 859.10 | 572.59 | 0.00 | 0.00 |
| **4** | 12820.0 | 0.00 | 0.00 | 0.00 | 210.35 | 0.00 | 170.46 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 343.76 | 217.77 |

Let's sum all the months sales.

sale['CLV']=sale.iloc[:,2:].sum(axis=1)

sale.head()

| **month\_yr** | **CustomerID** | **Apr-2011** | **Aug-2011** | **Dec-2010** | **Dec-2011** | **Feb-2011** | **Jan-2011** | **Jul-2011** | **Jun-2011** | **Mar-2011** | **May-2011** | **Nov-2011** | **Oct-2011** | **Sep-2011** | **CLV** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 12346.0 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 77183.60 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 77183.60 |
| **1** | 12747.0 | 0.00 | 301.70 | 706.27 | 438.50 | 0.00 | 303.04 | 0.00 | 376.30 | 310.78 | 771.31 | 312.73 | 675.38 | 0.00 | 4196.01 |
| **2** | 12748.0 | 1100.37 | 898.24 | 4228.13 | 1070.27 | 389.64 | 418.77 | 1113.27 | 2006.26 | 1179.37 | 2234.50 | 10639.23 | 2292.84 | 6148.84 | 32619.36 |
| **3** | 12749.0 | 0.00 | 1896.13 | 0.00 | 763.06 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 859.10 | 572.59 | 0.00 | 0.00 | 4090.88 |
| **4** | 12820.0 | 0.00 | 0.00 | 0.00 | 210.35 | 0.00 | 170.46 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 343.76 | 217.77 | 942.34 |

**Selecting Feature**

Here, you need to divide the given columns into two types of variables dependent(or target variable) and independent variable(or feature variables). Select latest 6 month as independent variable.

X=sale[['Dec-2011','Nov-2011', 'Oct-2011','Sep-2011','Aug-2011','Jul-2011']]

y=sale[['CLV']]

**Splitting Data**

To understand model performance, dividing the dataset into a training set and a test set is a good strategy.

Let's split dataset by using function train\_test\_split(). You need to pass 3 parameters features, target, and test\_set size. Additionally, you can use random\_state as a seed value to maintain reproducibility, which means whenever you split the data will not affect the results. Also, if random\_state is None, then random number generator uses np.random for selecting records randomly. It means If you don't set a seed, it is different each time.

#split training set and test set

from sklearn.cross\_validation import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y,random\_state=0)

**Model Development**

First, import the Linear Regression module and create a Linear Regression object. Then, fit your model on the train set using fit() function and perform prediction on the test set using predict() function.

# import model

from sklearn.linear\_model import LinearRegression

# instantiate

linreg = LinearRegression()

# fit the model to the training data (learn the coefficients)

linreg.fit(X\_train, y\_train)

# make predictions on the testing set

y\_pred = linreg.predict(X\_test)

# print the intercept and coefficients

print(linreg.intercept\_)

print(linreg.coef\_)

[208.50969617]

[[0.99880551 0.80381254 1.60226829 1.67433228 1.52860813 2.87959449]]

**How Well Does the Model Fit the data?**

In order to evaluate the overall fit of the linear model, we use the R-squared value. R-squared is the proportion of variance explained by the model. Value of R-squared lies between 0 and 1. Higher value or R-squared is considered better because it indicates the larger variance explained by the model.

from sklearn import metrics

# compute the R Square for model

print("R-Square:",metrics.r2\_score(y\_test, y\_pred))

R-Square: 0.9666074402817512

This model has a higher R-squared (0.96). This model provides a better fit to the data.

**Model Evaluation**

For regression problems following evaluation metrics used ([Ritchie Ng](https://www.ritchieng.com/machine-learning-linear-regression/)):

* **Mean Absolute Error (MAE)** is the mean of the absolute value of the errors.
* **Mean Squared Error (MSE)** is the mean of the squared errors.
* **Root Mean Squared Error (RMSE)** is the square root of the mean of the squared errors.

# calculate MAE using scikit-learn

print("MAE:",metrics.mean\_absolute\_error(y\_test,y\_pred))

#calculate mean squared error

print("MSE",metrics.mean\_squared\_error(y\_test, y\_pred))

# compute the RMSE of our predictions

print("RMSE:",np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))

MAE: 595.0282284701234

MSE 2114139.8898678957

RMSE: 1454.0082151995896

RMSE is more popular than MSE and MAE because RMSE is interpretable with y because of the same units.

**Pros and Cons of CLTV**

CLTV helps you to design an effective business plan and also provide a chance to scale your business. CLTV draw meaningful customer segments these segment can help you to identify needs of the different-different segment.

Customer Lifetime Value is a tool, not a strategy. CLTV can figure out most profitable customers, but how you are going to make a profit from them, it depends on your strategy. Generally, CLTV models are confused and misused. Obsession with CLTV may create blinders. Companies only focus on finding the best customer group and focusing on them and repeat the business, but it’s also important to give attention to other customers.

**Conclusion**

Congratulations, you have made it to the end of this tutorial!

In this tutorial, you have covered a lot of details about Customer Lifetime Value. You have learned what customer lifetime value is, approaches for calculating CLTV, implementation of CLTV from scratch in python, a prediction model for CLTV, and Pros and Cons of CLTV. Also, you covered some basic concepts of pandas such as groupby and pivot table for summarizing selected columns and rows of data.

Hopefully, you can now utilize CLTV concept to analyze your own datasets. Thanks for reading this tutorial!

If you would like to learn more about analyzing customer data in Python, take DataCamp's [Customer Analytics & A/B Testing in Python](https://www.datacamp.com/courses/customer-analytics-ab-testing-in-python) course.

**COMMENTS**

[**Jiaxiang Li**](https://www.datacamp.com/profile/lijiaxiang)

Good intro of  LTV. I follow a code and have a question, is there any way to give different churn rate for customers. Because I think it is large different between different customers.

**2**

**2**

REPLY[23/12/2018 07:57 AM](https://www.datacamp.com/community/tutorials/customer-life-time-value#comment-3386)

[**Avinash Navlani**](https://www.datacamp.com/profile/avinashnvln8)

Churn means when a customer leaves the service of one company and moves to another.  Churn rate definition is different for different context or domain or business model. For the subscription model, the churn rate can be the number of customers canceling the subscription. Another definition could be the number of customers, who don't continue the company's services.

**3**

**3**

REPLY[27/12/2018 11:39 PM](https://www.datacamp.com/community/tutorials/customer-life-time-value#comment-3436)

[**Bartosz Schneider**](https://www.datacamp.com/profile/bartoszschneider)

Dropping the negative order quantities can in reality be not such a good idea as returned or cancelled orders can appear as negative quantities, gross values etc. in your data. In these cases, summing those values on customerID + orderID solves this problem for you without dropping data. That way you can even incorporate return rates into your CLV model, because the data is not lost.

**8**

**8**

REPLY[07/01/2019 07:16 AM](https://www.datacamp.com/community/tutorials/customer-life-time-value#comment-3595)

[**jmercier**](https://www.datacamp.com/profile/jmercier)

Would you have any suggestion on how to proceed if the data we have has to do with contractual purchases such as insurance policies. Also, in such a data set, each client\_ID may be associated to more than one policy (e.g. car, life and home.) Say the columns are customer\_ID, product\_category (car, home or life), date\_of\_account\_opening, date\_account\_closing, and Premium.

How would one calculate the CLTV in that case?

**2**

**2**

REPLY[21/03/2019 06:18 PM](https://www.datacamp.com/community/tutorials/customer-life-time-value#comment-4477)

[**chenguttuvanlakshmanan**](https://www.datacamp.com/profile/chenguttuvanlakshmanan)

Hi Could please tell me in the prediction model , why you are using the total sales value instead of CLTV calculated from the previous step , i am trying for one of my project