**Modeling customer churn for an e-commerce company with Python**

*From <*[*https://towardsdatascience.com/modeling-customer-churn-for-an-e-commerce-business-with-python-874315e688bf*](https://towardsdatascience.com/modeling-customer-churn-for-an-e-commerce-business-with-python-874315e688bf)*>*

t’s more cost effective to retain existing customers than to acquire new ones, which is why it’s important to track customers at high risk of turnover (churn) and target them with retention strategies.

In this project, I’ll build a customer churn model based off of data from Olist, a Brazilian e-commerce site.

I’ll use that to identify high risk customers and inform retention strategies and marketing experiments.

There’s one complication with e-commerce. While it’s straightforward to measure churn for a contractual (subscription-based) business, churns aren’t explicitly observed in non-contractual businesses (e-commerce). In these scenarios, probabilistic models come in handy for estimating time of customer death. The probabilistic model that I’ll use is the BG/NBD model from the Lifetimes package.

Loading the data

import pandas as pd

import numpy as np

import datetime as dt

import seaborn as sns

import matplotlib.pyplot as pltfrom lifetimes.utils import \*

from lifetimes import BetaGeoFitter,GammaGammaFitter

from lifetimes.plotting import plot\_probability\_alive\_matrix, plot\_frequency\_recency\_matrix, plot\_period\_transactions, plot\_cumulative\_transactions,plot\_incremental\_transactionsrom lifetimes.generate\_data import beta\_geometric\_nbd\_modelfrom lifetimes.plotting import plot\_calibration\_purchases\_vs\_holdout\_purchases, plot\_period\_transactions,plot\_history\_aliveorders = pd.read\_csv(‘brazilian-ecommerce/olist\_orders\_dataset.csv’)

items = pd.read\_csv(‘brazilian-ecommerce/olist\_order\_items\_dataset.csv’)

cust = pd.read\_csv(‘brazilian-ecommerce/olist\_customers\_dataset.csv’)

The lifetimes package relies on recency-frequency-monetary (RFM) analysis to model churn and customer lifetime value (CLV). To make our models, we’ll need a a dataframe that consists of recency, frequency, and monetary columns. The definitions of each are below.

* Recency: time between initial purchase and most recent (last) purchase
* Frequency: number of repeat purchases made by a customer (total purchases — 1)
* Monetary: total spent on purchases

Customer ID information will come from cust. Order date will come from orders. Price will come from items.

Building the dataset

print(cust.columns)

There are two columns used to identify customers. customer\_id is a customer ID token that is generated for every order. If the same customer makes multiple orders, he has multiple customer\_id identifiers. What we want to use for this analysis is customer\_unique\_id, which is unique to each purchaser and can be used to track their purchases over time.

Here is the distribution of purchases made by customers.

cust.groupby('customer\_unique\_id').size().value\_counts()

The majority of customers made only a single purchase.

orders = pd.merge(orders,cust[['customer\_id','customer\_unique\_id']],on='customer\_id')orders.columns

In the items dataset, each item in an order gets its separate row. The price column refers cumulative order purchase rather than individual item price. Since I only need order price, I'll keep the first item from each order.

print(items.columns)

items.drop\_duplicates('order\_id',keep='first',inplace=True)

Next, I’ll join orders with items to append price information.

transaction\_data = pd.merge(orders,items,'inner','order\_id')transaction\_data = transaction\_data[['customer\_unique\_id','order\_purchase\_timestamp','price']]## convert timestamp to date; only need the daytransaction\_data['date'] = pd.to\_datetime(transaction\_data['order\_purchase\_timestamp']).dt.date

transaction\_data = transaction\_data.drop('order\_purchase\_timestamp',axis=1)transaction\_data.head()

Now that I have my transaction data, I want to convert this into dataframe with the RFM variables that I mentioned in the introduction. The Lifetimes package has a function for converting transaction data into an RFM DataFrame.

summary = summary\_data\_from\_transaction\_data(transaction\_data,'customer\_unique\_id','date',monetary\_value\_col='price',)

summary.describe()

Understanding the RFM DataFrame

summary.head()

The summary function converted customer transactions into an aggregated table. Many of the customers have frequency, recency, and monetary = 0, like customer 0000366f3b9a7992bf8c76cfdf3221e2. That's because Lifetimes only considers customers who have made repeat purchases into account.

Using days as time periods (can also be defined as weeks or months), variables are defined like so for the Lifetimes model:

* frequency: # of days in which a customer made a repeat purchase
* T: customer's age in days
* recency: customer's age in days at time of most recent purchase
* monetary\_value: mean of a customer's purchases, excluding the 1st purchase

frequency excludes the customer's first purchase because that is considered the day the customer is born. Afterwards, you can begin to question whether or not that customer is alive.

summary[summary['frequency']>0].head()

transaction\_data[transaction\_data['customer\_unique\_id']=='004288347e5e88a27ded2bb23747066c']

Note how customer 004288347e5e88a27ded2bb23747066c made two purchases with Olist but his frequency is 1 and monetary\_value is $87.90 based on how frequency and monetary\_value are defined.

Visualizing the RFM DataFrame

We’re going to use the Beta-Geometric/NBD (BG/NBD) model for customer churn. The BG/NBD model is an adaptation of the Pareto/NBD model. Both models describe repeat purchasing patterns in businesses where customer turnover is unobserved; however, the BG/NBD is much more computationally feasible.

Assumptions of the BG/NBD model:

* A customer’s relationship has two phases: “alive” for an unobserved period of time, then “dead”
* While alive, the number of transactions made by a customer follows a Poisson distribution with transaction rate lambda
* Heterogeneity in lambda follows a gamma distribution
* After any transaction, a customer dies with probability p; the probability that a customer dies after a number of transactions follows a geometric distribution
* p follows a beta distribution
* Lambda and p vary independently across customers

For more information on the BG/NBD model, check out this [paper](http://brucehardie.com/papers/bgnbd_2004-04-20.pdf) by Peter Fader and this [post](https://medium.com/data-shopify/how-shopify-merchants-can-measure-retention-c12284bfed6f) by Cam Davidson-Pilon.

bgf = BetaGeoFitter(penalizer\_coef=0.0)

bgf.fit(summary['frequency'], summary['recency'], summary['T']);

plot\_frequency\_recency\_matrix(bgf);

plot\_probability\_alive\_matrix(bgf)

Training a model and evaluating model performance

Next we want to evaluate the model to see how well it performs in the future. I’ll split the data into a training (calibration) period and a holdout (observation) period, train the BG/NBD model and evaluate performance with four plots that Peter Fader outlines in [this talk](https://www.youtube.com/watch?v=guj2gVEEx4s) (@ 26:10). These plots are:

**1) Calibration period histogram**: does the model fit the training data?

**2) Cumulative transaction plot**: does the model predict cumulative sales well?

**3) Incremental transaction plot**: does the model capture the overall trend in transactions?

**4) Conditional expectations plot**: can the model predict the number of purchases a customer will make based on the training data?

1) Calibration period histogram

plot\_period\_transactions(bgf).set\_yscale('log');

The model is fairly representative of the real data up until four repeat transactions. There are few customers who make more purchases.

2) Cumulative transaction plot

summary\_cal\_holdout = calibration\_and\_holdout\_data(transaction\_data, 'customer\_unique\_id', 'date',calibration\_period\_end='2017-09-03', observation\_period\_end='2018-09-03' )

We can evaluate how the dataset works by plotting both of them.

bgf.fit(summary\_cal\_holdout['frequency\_cal'], summary\_cal\_holdout['recency\_cal'], summary\_cal\_holdout['T\_cal'])plot\_cumulative\_transactions(bgf, transaction\_data, 'date', 'customer\_unique\_id', 730, 365);

The red line represents the boundary between the calibration period on the left and the holdout period on the right. As you can see, the BG/NBD model does a pretty swell job at predicting cumulative transactions.

3) Incremental transaction plot

plot\_incremental\_transactions(bgf, transaction\_data, 'date', 'customer\_unique\_id', 730, 365);

This plot shows that the model does a decent job capturing general trends in the data.

4) Conditional expectations plot

plot\_calibration\_purchases\_vs\_holdout\_purchases(bgf, summary\_cal\_holdout);

The model performs well up to three calibration period purchases, but diverges from the holdout data because of the distribution of the data.

cust.groupby('customer\_unique\_id').size().value\_counts()

Less than 1% of customers have made four or more purchases, so there’s not much data for the BG/NBD model to learn about customers who have made many repeat transactions.

In practice, I would consider collecting more data if I were to proceed with modeling customer churn. But for learning purposes, it will still be a good exercise to predict churn.

Modeling churn risk

The BG/NBD model assumes that death can only occur after a repeat purchase, since the customer leaving occurs during a purchase and the first purchase is reserved to signal a customer’s birth.

Because of this, customers with only one transactions will have a 100% probability of being alive, which is suspect. To account for this limitation, we’ll only predict churn risk on customers who have made at least one repeat transaction.

df = summary[summary['frequency']>0]df['prob\_alive'] = bgf.conditional\_probability\_alive(df['frequency'],df['recency'],df['T'])sns.distplot(df['prob\_alive']);

From here, we can visualize customers based on the probability that they’re “alive”. Using domain knowledge we might be able to set a threshold for customers who probably have already churned, and also identify customers who are at risk for churning, but haven’t yet disappeared.

Next, I would like to set a decision threshold for customer churn. This is an opportunity to inject personal expertise or talk with domain experts. Assume I speak with the sales and marketing managers, and we agree to consider a customer with <10% chance of being alive to have churned.

df['churn'] = ['churned' if p < .1 else 'not churned' for p in df['prob\_alive']]sns.countplot(df['churn']);

A little over 92% of customers have churned, meaning that there’s a lot of opportunity for improvement regarding retention.

We can assume that the customers who have churned are already lost. But what is interesting in a business setting is the customers who are at high risk for churn, but haven’t churned yet. Later on, it might still be a good idea to apply different treatments to the churned group.

If I can identify them, maybe I can encourage the marketing team to target them with promotions.

sns.distplot(df[df['churn']=='not churned']['prob\_alive']).set\_title('Probability alive, not churned');

It seems reasonable to bucket customers with 80% or more churn risk to be considered high risk for churn.

df['churn'][(df['prob\_alive']>=.1) & (df['prob\_alive']<.2)] = "high risk"

df['churn'].value\_counts()

Now that I have these churn groupings, I can move forward and apply special treatments to these groups. Ideally there would be more data and a bigger population of high-risk customers.

Conclusion

We’ve modeled churn risk in a non-contractual setting, and now have three customer segments — not churned, high risk, and churned. This could feed into a dashboard to give stakeholders a glimpse of “at-risk” customers. It also provides three different groups that we can run specific actions. Some ideas:

1) Reach out to churned customers to figure out why they left.

2) Send different types of targeted emails and special offers to the high risk group. If the sample size of high risk customers is large enough, you could split off a few small treatment groups and compare how their retention and CLV change with different promotional or customer relationship strategies.

3) Determine the the highest value customers in the non-churn group, and serve them additional benefits to ensure that they remain loyal customers.

[Collin Ching](https://towardsdatascience.com/@collindching?source=user_profile---------0-----------------------) in [Towards Data Science](https://towardsdatascience.com/?source=user_profile---------0-----------------------)

*From <*[*https://towardsdatascience.com/@collindching*](https://towardsdatascience.com/@collindching)*>*

https://github.com/collindching/Olist-Customer-Churn/blob/master/Olist%20churn%20risk%20model.ipynb

*From <*[*https://towardsdatascience.com/modeling-customer-churn-for-an-e-commerce-business-with-python-874315e688bf*](https://towardsdatascience.com/modeling-customer-churn-for-an-e-commerce-business-with-python-874315e688bf)*>*

BG-NBD Model for Customer Base Analysishete

*From <*[*http://benalexkeen.com/bg-nbd-model-for-customer-base-analysis-in-python/*](http://benalexkeen.com/bg-nbd-model-for-customer-base-analysis-in-python/)*>*

### Useful resources

* [What's a Customer Worth?](https://towardsdatascience.com/whats-a-customer-worth-8daf183f8a4f) good starter tutorial on customer transactions and lifetime value with Lifetimes
* [Lifetimes documentation](https://lifetimes.readthedocs.io/en/latest/Quickstart.html)
* [Kaggle Customer Lifetime Value](https://www.kaggle.com/dhimananubhav/customer-lifetime-value/comments) same goal, same data, different model and analysis
* [Peter Fader CLV presentation](https://www.youtube.com/watch?v=guj2gVEEx4s) Peter Fader explains CLV... his overview of model performance graphs in the presentation is helpful
* [Robert Medri's Etsy CLV presentation](http://cdn.oreillystatic.com/en/assets/1/event/85/Case%20Study_%20What_s%20a%20Customer%20Worth_%20Presentation.pdf): slides titled "Acting on CLV" explain next action steps after you've modeled churn and CLV

<https://lifetimes.readthedocs.io/en/latest/>

**Measuring users is hard. Lifetimes makes it easy.**

**Introduction**

Lifetimes can be used to analyze your users based on a few assumption:

1. Users interact with you when they are “alive”.
2. Users under study may “die” after some period of time.

I’ve quoted “alive” and “die” as these are the most abstract terms: feel free to use your own definition of “alive” and “die” (they are used similarly to “birth” and “death” in survival analysis). Whenever we have individuals repeating occurrences, we can use Lifetimes to help understand user behaviour.

**Applications**

If this is too abstract, consider these applications:

* Predicting how often a visitor will return to your website. (Alive = visiting. Die = decided the website wasn’t for them)
* Understanding how frequently a patient may return to a hospital. (Alive = visiting. Die = maybe the patient moved to a new city, or became deceased.)
* Predicting individuals who have churned from an app using only their usage history. (Alive = logins. Die = removed the app)
* Predicting repeat purchases from a customer. (Alive = actively purchasing. Die = became disinterested with your product)
* Predicting the lifetime value of your customers

**Specific Application: Customer Lifetime Value**

As emphasized by P. Fader and B. Hardie, understanding and acting on customer lifetime value (CLV) is the most important part of your business’s sales efforts. [And (apparently) everyone is doing it wrong](https://www.youtube.com/watch?v=guj2gVEEx4s). *Lifetimes* is a Python library to calculate CLV for you.

**Installation**

pip install lifetimes

**Documentation and tutorials**

[Official documentation](http://lifetimes.readthedocs.io/en/latest/)

**Questions? Comments? Requests?**

Please create an issue in the [lifetimes repository](https://github.com/CamDavidsonPilon/lifetimes).

**More Information**

1. [Roberto Medri](http://cdn.oreillystatic.com/en/assets/1/event/85/Case%20Study_%20What_s%20a%20Customer%20Worth_%20Presentation.pdf) did a nice presentation on CLV at Etsy.
2. [Papers](http://mktg.uni-svishtov.bg/ivm/resources/Counting_Your_Customers.pdf), lots of [papers](http://brucehardie.com/notes/009/pareto_nbd_derivations_2005-11-05.pdf).
3. R implementation is called [BTYD](http://cran.r-project.org/web/packages/BTYD/vignettes/BTYD-walkthrough.pdf) (for, *Buy ’Til You Die*).

**Contents:**

* [Quickstart](https://lifetimes.readthedocs.io/en/latest/Quickstart.html)
  + [The shape of your data](https://lifetimes.readthedocs.io/en/latest/Quickstart.html#the-shape-of-your-data)
  + [Basic Frequency/Recency analysis using the BG/NBD model](https://lifetimes.readthedocs.io/en/latest/Quickstart.html#basic-frequency-recency-analysis-using-the-bg-nbd-model)
  + [Estimating customer lifetime value using the Gamma-Gamma model](https://lifetimes.readthedocs.io/en/latest/Quickstart.html#estimating-customer-lifetime-value-using-the-gamma-gamma-model)
* [Saving and loading model](https://lifetimes.readthedocs.io/en/latest/Saving%20and%20loading%20model.html)
  + [Fit model](https://lifetimes.readthedocs.io/en/latest/Saving%20and%20loading%20model.html#fit-model)
  + [Saving model](https://lifetimes.readthedocs.io/en/latest/Saving%20and%20loading%20model.html#saving-model)
  + [Loading model](https://lifetimes.readthedocs.io/en/latest/Saving%20and%20loading%20model.html#loading-model)
* [More Examples and recipes](https://lifetimes.readthedocs.io/en/latest/More%20examples%20and%20recipes.html)
  + [Example SQL statement to transform transactional data into RFM data](https://lifetimes.readthedocs.io/en/latest/More%20examples%20and%20recipes.html#example-sql-statement-to-transform-transactional-data-into-rfm-data)
  + [Create table with RFM summary matrix with holdout](https://lifetimes.readthedocs.io/en/latest/More%20examples%20and%20recipes.html#create-table-with-rfm-summary-matrix-with-holdout)
* [lifetimes package](https://lifetimes.readthedocs.io/en/latest/lifetimes.html)
  + [Subpackages](https://lifetimes.readthedocs.io/en/latest/lifetimes.html#subpackages)
  + [Submodules](https://lifetimes.readthedocs.io/en/latest/lifetimes.html#submodules)
  + [lifetimes.estimation module](https://lifetimes.readthedocs.io/en/latest/lifetimes.html#lifetimes-estimation-module)
  + [lifetimes.generate\_data module](https://lifetimes.readthedocs.io/en/latest/lifetimes.html#module-lifetimes.generate_data)
  + [lifetimes.plotting module](https://lifetimes.readthedocs.io/en/latest/lifetimes.html#module-lifetimes.plotting)
  + [lifetimes.utils module](https://lifetimes.readthedocs.io/en/latest/lifetimes.html#module-lifetimes.utils)
  + [lifetimes.version module](https://lifetimes.readthedocs.io/en/latest/lifetimes.html#module-lifetimes.version)
  + [Module contents](https://lifetimes.readthedocs.io/en/latest/lifetimes.html#module-lifetimes)
* [Changelog](https://lifetimes.readthedocs.io/en/latest/Changelog.html)
  + [0.11.1](https://lifetimes.readthedocs.io/en/latest/Changelog.html#id1)
  + [0.11.0](https://lifetimes.readthedocs.io/en/latest/Changelog.html#section-1)
  + [0.10.1](https://lifetimes.readthedocs.io/en/latest/Changelog.html#section-2)
  + [0.10.0](https://lifetimes.readthedocs.io/en/latest/Changelog.html#section-3)
  + [0.9.1](https://lifetimes.readthedocs.io/en/latest/Changelog.html#section-4)
  + [0.9.0](https://lifetimes.readthedocs.io/en/latest/Changelog.html#section-5)
  + [0.8.1](https://lifetimes.readthedocs.io/en/latest/Changelog.html#section-6)
* [High Level Overview](https://lifetimes.readthedocs.io/en/latest/High%20Level%20Overview.html)
  + [Workflow](https://lifetimes.readthedocs.io/en/latest/High%20Level%20Overview.html#workflow)
  + [Fitters](https://lifetimes.readthedocs.io/en/latest/High%20Level%20Overview.html#fitters)
  + [Graphs](https://lifetimes.readthedocs.io/en/latest/High%20Level%20Overview.html#graphs)
  + [The utils.py File](https://lifetimes.readthedocs.io/en/latest/High%20Level%20Overview.html#the-utils-py-file)

**Indices and tables**

* [Index](https://lifetimes.readthedocs.io/en/latest/genindex.html)
* [Module Index](https://lifetimes.readthedocs.io/en/latest/py-modindex.html)
* [Search Page](https://lifetimes.readthedocs.io/en/latest/search.html)

[Next](https://lifetimes.readthedocs.io/en/latest/Quickstart.html)

 Predicting customer lifetime value and future purchases

<https://www.kaggle.com/dhimananubhav/e-commerce-predicting-customer-lifetime-value/comments>