Abstract

Estimating future value of a customer is one of the core pillars in marketing strategy. Value can be perceived either by the net profit or revenue earned from a customer during a period of defined length in the future. A number of Econometric, Probabilistic and Machine Learning models utilize customer level transactions for this purpose.

The modeling approach explored in this project is one amongst a suite of Bayesian probabilistic models popularly known as the Buy till you die models for estimating customer value. The report discusses the theory and application of a combination of Beta Geometric Negative Binomial Distribution (BG/NBD) model and the Gamma-Gamma submodel for estimating the expected future value of customers for an e-commerce retail business. The BG/NBD model was first introduced by Fader, Hardie and Lee in 2004 for predicting expected future transactions and survival probability for customers in a non-contractual setup.

The model is trained over a calibration period of 9 months and the predictions are tested for a holdout period of 4 months. The model performance is evaluated against a simplistic baseline model based on the observed average behavior of an individual. Finally, the model is used to predict "High Future Value" customers and the lift obtained in capturing target customers is reported.

Contents

Buy till you die model for customer future value prediction

Utkarsh Singh

July 19, 2020

1 Introduction

Organizations are increasingly developing their arsenal of marketing tactics to drive customer centricity. At the centre of marketing research is Campaign Budget Optimization (CBO) for re-distributing budget to prioritize the *high potential audience*. Customer Lifetime Value (CLV) is an important metric used by marketers to identify such an audience. It is defined as the net present value of all the future cash flows generated by the customer over his/her *lifetime*. Typically, the *lifetime duration* is considered to be three years due to two reasons: (a) Product Lifecycle and, (b) 80% of profit comes in three years (Gupta and Lehman, 2006). However, this period can be defined as per marketing needs. In order to estimate CLV, the following questions need to be addressed:

- How many times is a customer expected to purchase in a given period in future?
- What will be the worth of these future purchases?
- What will be the total cost of making the sale?

For the scope of this project, the cost component is kept out of the computation due to data unavailability. CLV is estimated as the net expected revenue earned for a duration of 4 months in the future.

In a non-contractual setting, a customer can drop out at any point in time. Therefore, a single definition of churn cannot be used to predict drop-out probability, which depends on, amongst other factors, the individual's own frequency of purchase and her time of last purchase. This point is illustrated with the plot below:

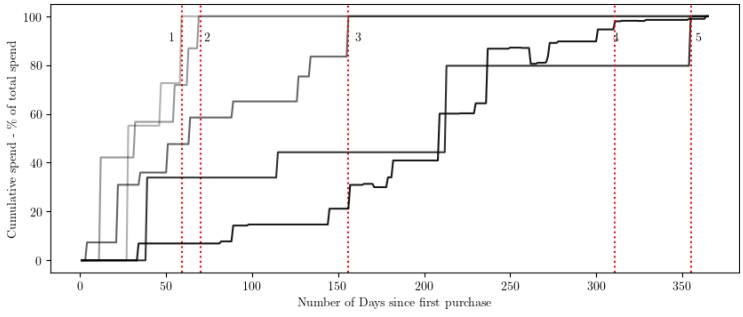


Fig1.Spend distribution and dropout

Customer 1 ceases activity within ~ 60 days from the first shop (cumulative spend reaches total spend), whereas customer 5 takes over 100 days within consecutive purchases and continues to stay alive till the 350th day. It is reasonable to assume that the probability of staying alive beyond the 350th day is much higher for customer 5 as compared to customer 1. The report discusses in detail the probabilistic models that account for these subtleties of customer buying pattern.

2 Objective

The objective of this project is to demonstrate the application of Bayesian probabilistic models to estimate future customer value. This involves predicting the expected count and value of transactions in a given period of time in the future for every customer. The models are designed for a non-contractual business, and assume that customer response is unsolicited i.e. no sales triggers are used in the calibration or holdout period. The probabilistic model discussed accounts for the fact that a customer has a certain probability of dropping out after every purchase.

3 Data

Typically e-commerce transaction-level datasets are proprietary and consequently unavailable in public domain for free. However, The UCI Machine Learning Repository has collected this dataset containing actual transactions from 2010 and 2011 for a UK-based and registered non-store online retailer. A majority of customers are wholesalers and small B2C businesses. The dataset is available for public access and can be found on UCI's website by the title "Online Retail". The data contains 20,728 transaction for 4,372 customers in data occurring between 01/12/2010 and 09/12/2011. See import code and information on the data fields here in the appendix.

See the python utilities used for the project.

4 Exploratory Analysis

For aggregating data at a customer level, a different date of transaction for a customer is considered a different "visit". All transactions made on a single date come under the same "visit". Refer to the data aggregation code. Below is a snapshot of the aggregated data:

4.1 Visits per customer

Probabilistic models can be used to model the expected number of visits in a given period for repeat buyers. 66% of the customers in the given dataset are repeat buyers. On average, repeat buyers have made >5 vists, but the variation of frequency is high across customers. The distribution for number of visits per customer can be modelled by a Poisson family distribution (Colombo and Jiang, 1999). View the summary statistics for the number of visits for repeat visitors here

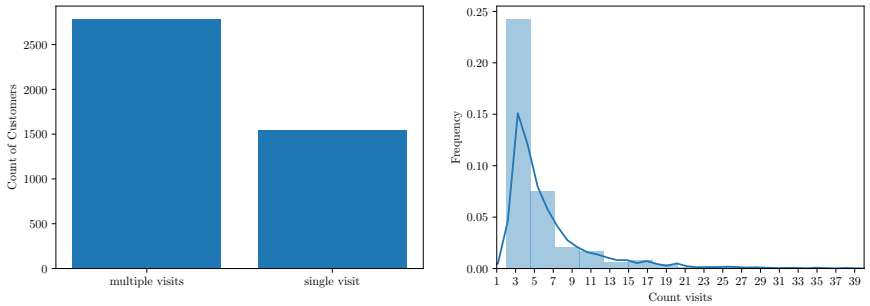


Fig 2. Distribution of visits

4.2 Average spend per visit

Average spend per visit typically is in the range of £[100,500] (Please note that most customers are B2C businesses). Since spend data is bounded by 0 on the left and tends to be right skewed, gamma distribution is used as a prior for modelling spend (Colombo and Jiang, 1999).

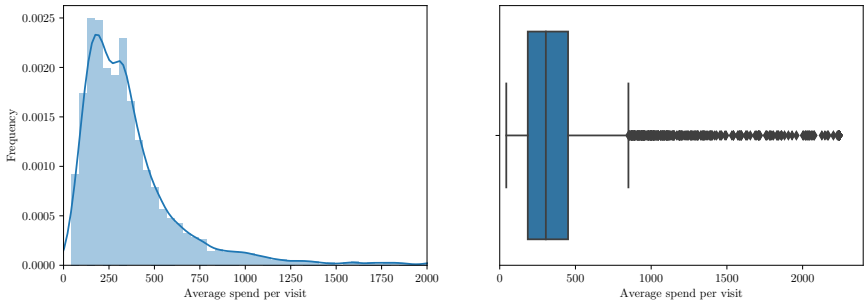


Fig 3. Distribution of Avg. spend per visit for customers

4.2.1 Analyze spend over subsequent visits

In order to analyze whether spend increases or decreases with subsequent visits, repeat buyers are first segmented based on their total visits. Within each segment, average spend is plotted with the n^{th} visit. Figure 4 shows that customers with higher frequency have a higher average spend per visit. However, there is no observable trend in the average spend as customers make subsequent visits. This behavior is observed across the aforementioned customer segments. Refer to the code for calculating average spend over subsequent visits

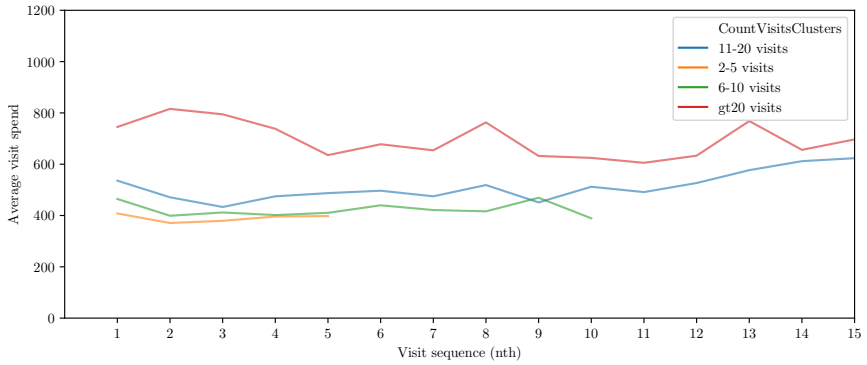


Fig 4. Average spend over subsequent visits

4.3 Visit cycle

Typically customers make a subsequent visit within 30-45 days from the previous visit. Figure 5 shows the average number of visits per month for repeat buyers across segments based on total visits. The summary statistics for visits per month (for all customers) show that 75% of the population makes under 1.2 visits per month. This figure is 1.5 for repeat buyers . Refer to the code for visit cycle summary. Therefore, a period of 1 month can be taken as a suitable unit for inter-arrival time. This will later on be used for determining the length of observation and holdout periods for the Bayesian model. Click to view the summary statistics for *visits per month*

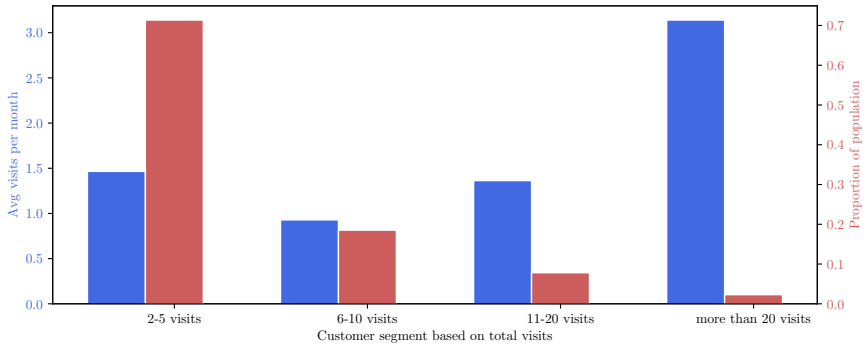


Fig 5. Average spend over subsequent visits

4.4 RFM metrics

Recency, Frequency and Monetary metrics are key inputs to predicting future purchases and drop-out probability of a customer at any point in time. Further sections of the report would discuss in detail the theory behind the use of RFM data for probabilistic models. In the context of CLV modelling, the RFM metrics are defined as follows:

- Recency: Number of time periods elapsed (days in this case) between the first and the last visit of a customer
- Frequency: Number of repeat visits
- Monetary: Average spend value per visit for repeat visits

"T" is the total observation period for a customer - from the date of first visit to the end of observation period. The table below summarizes the RFM metrics for all customers in the data. Refer to the RFM summary code.

50% of the customers have less than or equal to 1 repeat visit in the data. Summary of Recency and Frequency metrics calculated for the entire dataset is tabulated below:

	frequency	recency	T	monetary_value
mean	2.864024	130.741415	222.782899	306.944168
std	5.952745	132.210176	117.883623	2612.453380
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	112.000000	0.000000
50%	1.000000	93.000000	248.000000	176.940000
75%	3.000000	252.000000	326.000000	360.161667
max	131.000000	373.000000	373.000000	168469.600000

The Monetary and Frequency metrics do not show strong linear correlation with each other as can be seen in the plot below. Recency and Frequency display non-linear correlation. Customers with a high frequency have typically made a visit in the recent past. Please note that higher recency values indicate that the customer has made a purchase more recently (with reference to 09/12/2011 as end date for the observation period). Customers with high frequency and low recency values have a lower probability of being alive by the end of the observation period.

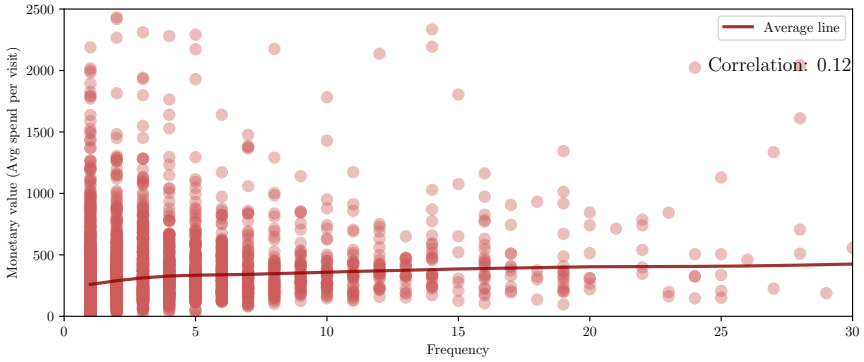


Fig 6. Monetary value vs Frequency plot

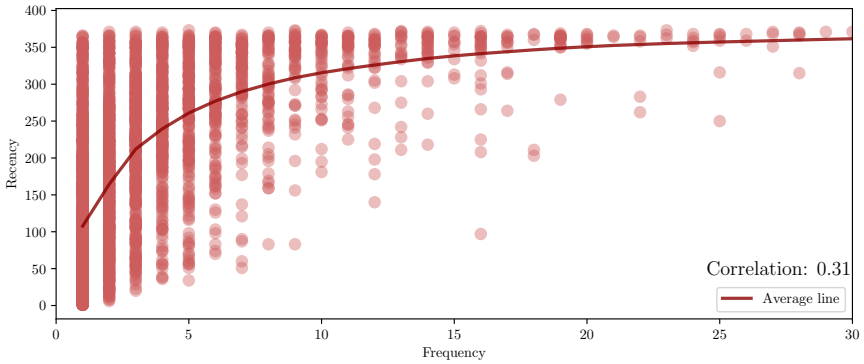


Fig 7. Recency vs Frequency plot

5 BG/NBD Model - Theory

The Beta Geometric/Negative Binomial Distribution (BG/NBD) model [2] is a Bayesian inference model introduced in 2004 by Fader, Hardie and Lee as an improvement of the Pareto/NBD model developed by Schmittlein et al. in 1987. The BG/NBD model belongs to the class of "Buy Till You Die" probabilistic models that help in projecting the future value of a customer by assessing the expected number of his/her future transactions and the probability of being "alive" beyond any transaction. Please note that the words "Visit" and "Transaction" are used interchangeably in this context and have the same meaning.

5.1 Assumptions

The BG/NBD model is based on the following assumptions [2]:

i) While active, the number of transactions made by a customer follows a Poisson process with transaction rate λ. This is equivalent to assuming that the time between transactions is distributed exponentially with transaction rate λ, i.e.,

$$f(t_j|t_{j-1};\lambda) = \lambda e^{-\lambda(t_j - t_{j-1})} \tag{1}$$

ii) Heterogeneity exists in λ, i.e. each customer has his/her own transaction rate. This intrinsic λ is unknown for a customer and is considered a latent random variable. The model assumes that λ follows a gamma prior with pdf:

$$f(\lambda|r,\alpha) = \frac{\alpha^r \lambda^{r-1} e^{-\lambda\alpha}}{\Gamma(r)}, \lambda > 0.$$

r is the shape parameter and α is the scale parameter.

iii) After any transaction, a customer becomes inactive with probability p. Therefore the point at which the customer "drops out" is distributed across transactions according to a (shifted) geometric distribution with pmf

$$P\left(inactive\, immediately\, after\, j^{th}\, transaction\right) = p(1-p)^{j-1}, j = 1,2,3,..$$

iv) Heterogeneity exists in p. Similar to λ, p is a latent random variable and follows a Beta prior with pdf:

$$f(p|a,b) = \frac{p^{a-1}(1-p)^{b-1}}{B(a,b)}, 0 \leq p \leq 1.$$

v) The transaction rate λ and the dropout probability p vary independently across customers.

To better illustrate the assumptions, figure 8 shows:

(left) poisson distributions simulated for 10 customers where each λ is randomly chosen from a Gamma distribution. (right) Geometric distribution of drop-out probability simulated for 10 customers where each p is randomly chosen from a Beta distribution

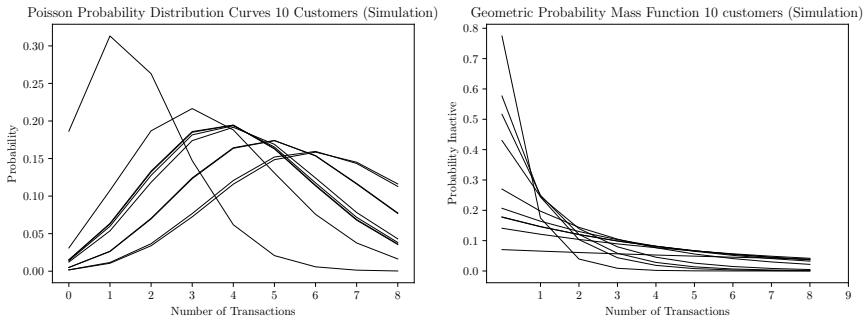
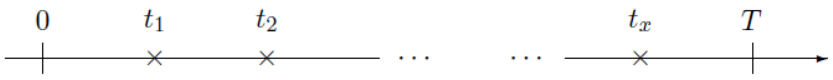


Fig 8. Illustrative simulations

5.2 Outputs

Consider a customer who had x transactions in the period (0,T] with the transactions occurring at $t_1, t_2, \ldots, t_x$ as shown below:



- The likelihood of the first transaction occurring at t_1 is a standard exponential likelihood component $\lambda e^{-\lambda t_1}$
- The likelihood of the xth transaction occurring at t_x is the probability of remaining active at t_{x-1} times the standard exponential likelihood component, which is equal to $(1-p)\lambda e^{-\lambda(t_x - t_{x-1})}$

- Finally, the likelihood of observing zero purchases in $(t_x, T]$ is the probability the customer became inactive at t_x, plus the probability he remained active but made no purchases in this interval. This is equal to $p + (1-p)e^{-\lambda(T-t_x)}$

Multiplying all the likelihoods, the overall likelihood function for a given sequence of purchases is

$$L(\lambda, p|t_1, t_2..t_x, T) = (1-p)^x \lambda^x e^{-\lambda T} + \delta_{x>0} p(1-p)^{x-1} \lambda^x e^{-\lambda t_x}$$

Therefore, the information on the timing of all x transactions is not required. A sufficient summary of the customer's purchase history is $(X = x, t_x, T)$

Eventually, by fitting the previously mentioned distributions on the historical customers data, the derived model estimates the following for each customer (see [2] for derivations):

- The probability of observing x transactions in a time period of length t as:

$$P(X(t) = x|\lambda, p) = (1-p)^x \frac{(\lambda t)^x e^{-\lambda t}}{x!} + \delta_{x>0} p(1-p)^{x-1} \left[1 - e^{-\lambda t} \sum_{j=0}^{x-1} \frac{(\lambda t)^j}{j!}\right]$$

- The expected number of transactions in a time period of length t as:

$$E(X(t)|\lambda, p) = \frac{1}{p} - \frac{e^{-\lambda p t}}{p}$$

- The probability of a customer becoming inactive at period τ as:

$$P(\tau > t) = e^{-\lambda p t}$$

The fitted distributions parameters are then used in the forward-looking customer-base analysis to find the **expected number of transactions in a future period of length t for an individual with past observed behavior defined by x, t_x, T** — where x = number of historical transactions, t_x = time of last purchase and T = Age of a customer (Time between first visit to the end of observation period). Derivation is present in the Appendix of P. Fader's Paper[2].

The four BG/NBD model parameters (r,α,a,b) can be estimated from data via the method of maximum likelihood. For a sample of N customers, the sample log-likelihood function given below can be maximized by standard numerical optimization[2].

$$LL(r, \alpha, a, b) = \sum_{i=1}^{N} \ln\left[L(r, \alpha, a, b \mid X_i = x_i, t_{x_i}, T_i)\right]$$

The above model needs the total visits (x), recency (t_x) and the observation duration (T) values for each customer

6 Gamma-Gamma Model - Theory

The Gamma-Gamma submodel is used to estimate the average monetary value of transactions for a customer. This model was first presented in Fader et al. (2005) and the detailed derivations are shown in Fader, Hardie (2013) [3].

For a customer with x transactions, let $z_1, z_2, ...z_x$ denote the value of each transaction. The customer's observed average transaction is $\bar{z} = \sum_{i=1}^{x} z_i / x$.

$\bar{z}$ is an imperfect estimate of their (unobserved) mean transaction value ζ. The goal of the Gamma-Gamma submodel is to make inferences about ζ given $\bar{z}$, which is denoted as E(Z | $\bar{z}$,x)

6.1 Assumptions

The spend model is based on the following assumptions:

i) Average transaction value $\bar{z}$ varies across customers but does not vary over time for any individual customer

ii) The monetary value of a customer's given visit varies randomly around their average visit spend

iii) The distribution of the average visit spend across customers is indepedent of the transaction process

Furthermore, the following assumptions are made about the distribution of spend:

i) It is assumed that z_i ~ Gamma(p,ν), with E(Z_i | p,ν) = ζ = p/ν

ii) For heterogeneity across customers, we assume that ν ~ gamma(q,γ)

6.2 Outputs

Finally the objective is to make inferences about an individual customer's ζ given $\bar{z}$ where $\bar{z}$ itself follows the below distribution given p,q,γ; x [3]. (Note: the parameters p,q,γ are estimated from the data):

$$f(\bar{z}|p,q,\gamma;x) = \frac{1}{\bar{z}B(px,q)}(\frac{\gamma}{\gamma+x\bar{z}})^q(\frac{x\bar{z}}{\gamma+x\bar{z}})^{px}$$

The final result is derived to be as follows [3]:

$$E(Z|p,q,\gamma;\bar{z},x) = (\frac{q-1}{px+q-1})\frac{p\gamma}{q-1} + (\frac{px}{px+q-1})\bar{z}$$

Please see the paper for complete derivation. This is called the conditional expectation of the customer's inherent mean transaction value - which is a latent variable. The point to note here is that as the number of visits x of customer increases, the formula places more weight on the customer's observed average spend.

The above formula needs as inputs the number of visits (x) and the Monetary ($\bar{z}$) values for each customer. The model parameters are estimated from this data.

Expected future value = Expected number of visits in future period (using BG/NBD model) *x* Expected spend per visit (using Gamma-Gamma model)

7 Model Implementation

The *lifetimes* module in python is used for implementing the BG/NBD and Gamma-Gamma models.

The entire timeline of the dataset is divided into a Calibration period and a Holdout period. Ideally the length of calibration duration should be kept a minimum of 5-10 times the visit cycle. As discussed in the EDA, the typical visit cycle for customers is 30-45 days. Keeping this in mind, the calibration period is taken up to 8^{th} August 2011. The Holdout period spans from 9^{th} August to 9^{th} December (3 months). The model then estimates the distribution parameters from the transactions of all customers who have made a repeat visit up to 8^{th} August 2011. Then each customer's expected visits and average spend is projected for the holdout period. The projected values are compared with the actual values in the holdout period for model validation. Projected values are also compared against a **baseline** projection made by using average values per unit time. Refer to the code for calibration and holdout data creation. A snapshot of the summarized data is shown here.

7.1 Fit BG/NBD model for predicting expected visits in future

The BetaGeoFitter() function in the *lifetimes* module randomly initializes the parameter values and then maximizes the $LL(r, \alpha, a, b)$ using gradient descent to estimate the model parameters. Refer to the code for fitting BG/NBD model. The estimated values are tabulated below:

	coef	se(coef)	lower 95% bound	upper 95% bound
r	2.380728	0.123647	2.138380	2.623076
alpha	119.357263	6.589274	106.442285	132.272241
a	0.018101	0.010517	-0.002513	0.038715
b	0.161093	0.095241	-0.025580	0.347766

The standard error for the a and b parameters (the Beta distribution parameters for the drop-out probability p) is high and the estimations are slightly unstable. This could be due to a bias in the sample, since the data does not capture the entire transaction history of customers. The high average recency value of 135 in a calibration window of 187 days for customers with repeat transactions indicates that customers in the sample largely remain active throughout the training duration.

Based on the estimated parameters, the posterior distributions for the visit rate (time between subsequent visits) and the drop-out probability are shown in figure 9.This is equivalent to saying that the estimated visit rate and drop-out probability after each visit for any individual is a random sample drawn from the below distributions.

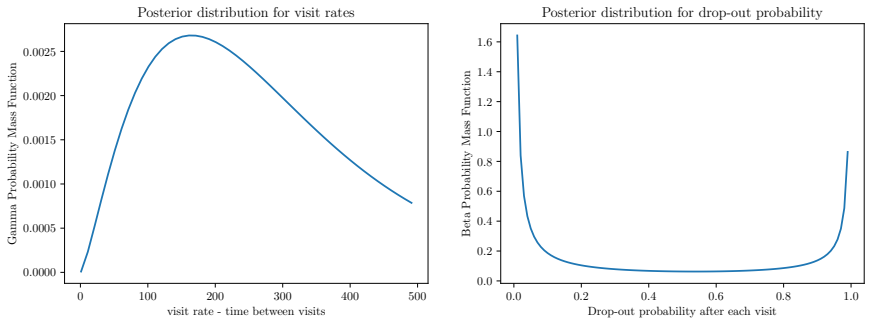


Fig 9. Posterior distributions based on estimated parameters

The posterior distribution for drop-out probability shows that most customers in the dataset either have a very small drop-out probability after each visit or are likely to drop-out almost immediately after a visit.

7.2 Assess model fit

The fitted model parameters are used to simulate a distribution for the calibration visit frequency. This distribution is then compared with the actual distribution. The below plot shows that the estimated model parameters closely resembles the real distribution for all visit frequency segments except 0. Please note that customers with 0 frequency were filtered out for training.

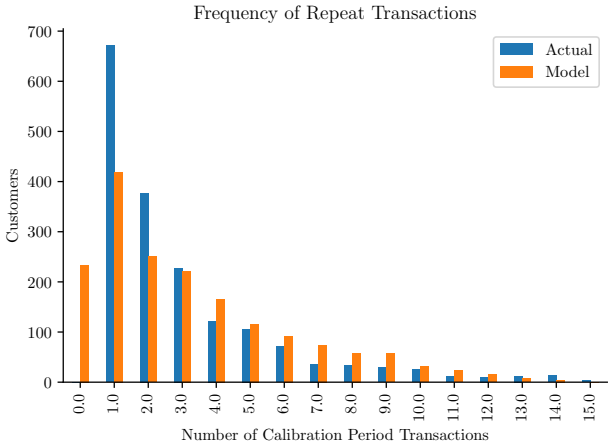
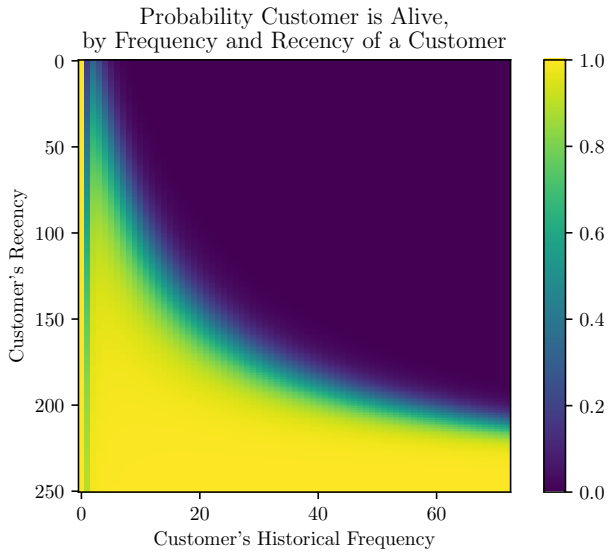


Fig 9. Actual vs Fitted distributions

The fitted model is used to estimate conditional expected number of visits in the calibration duration. The predictions are compared with the actual total number of visits in the calibration period. The model is fitted back on the training data (calibration period) to calculate the error in prediction. The Root Mean Squared Error in predictions over the training data is 1.97. Refer to the training RMSE code. Since it is calculated by averaging over all customers, the overall RMSE might be influenced by outliers. A more granular error estimation is shown in the validation section.

7.2.1 Probability of being alive by customer recency and frequency

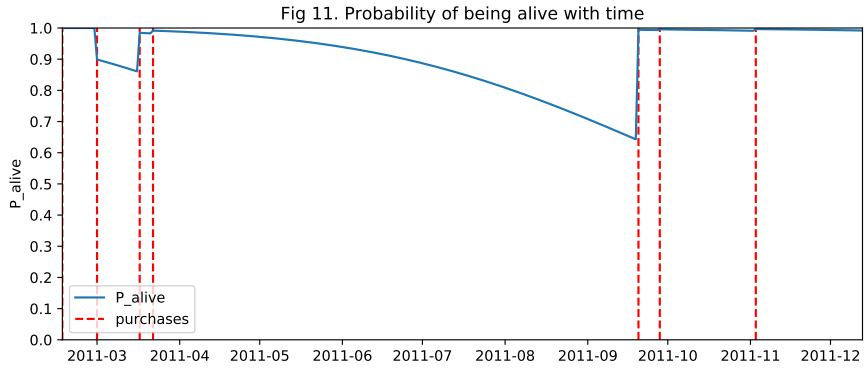


Below are the possible interpretations of the plot:

1. **Customers with low frequency and low recency values**: These customers are relatively young i.e. they have entered the system recently. They have made a few repeat visits over a short duration and are most likely alive.
2. **Customers with low frequency and high recency values**: These customers space out their purchases over time i.e they have longer purchase cycles. They have made a visit recently and are likely to come back again in some time.
3. **Customers with high frequency and high recency values**: These are the most active customers who have also visited recently. They are most likely alive.
4. **Customers with very high frequency but low recency values (i.e. no recent visits)**: These customers have made frequent visits in the past but have not visited in the recent past. They are most likely to have dropped out.

7.2.2 Survival probability with time

The below plot generates the probability of being alive with visits made across time for a selected customer. This illustrates how the probability decreases if a repeat visit is not made as per the historic visit cycle. However, once a customer makes a visit he is considered "alive". Refer to the code to generate probability_alive here.



7.3 Validation

The BG/NBD model is now used to predict the conditional expected number of visits for each customer for the holdout period. The predicted frequency is then compared with the actual holdout frequency. Overall prediction error is calculated by averaging error metrics over all customers. Additionally, Customers are segmented based on the calibration visit frequency ("repeat visits") and the prediction error is reported across these segments for granularity. Please note that the frequency in the holdout period refers to the total visits for both predicted and actual values (contrary to the "repeat visits" in the calibration period used for training).

Figure 12 compares the average predicted visits vs. the average actual visits for the aforementioned customer segments (shown on x-axis). Prediction error is reported using the below metrics. Refer to the code for calculating RMSE and MAPE for predictions

- Root Mean Squared Error (RMSE): sqrt((predicted - actual)^2/N)
- Mean Absolute Percentage Error: mean(abs(predicted/actual - 1))

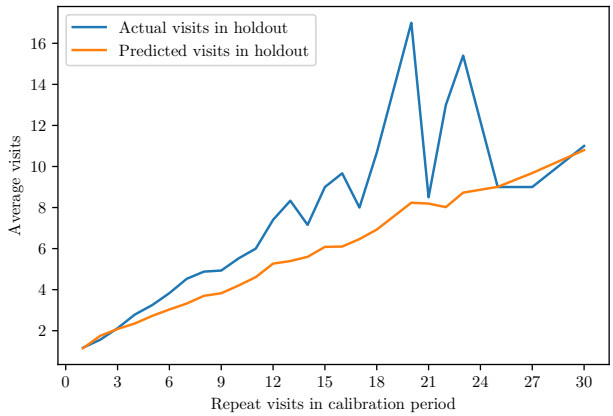


Fig 12. Actual vs Predicted visits in holdout period

The model systematically underpredicts holdout frequency for customers with >3 frequency in the calibration period. The error is high for frequent buyers who have shown an outlying behavior in the holdout duration - marked by the peaks in actual visits.

The below table shows the RMSE and MAPE in holdout predictions across the customer segments based on calibration frequency:

Calibration frequency segment	Customer count	RMSE	MAPE(%)
<=3	1276	1.42	46.93
4-6	299	2.32	54.93
7-9	98	2.85	58.34
10-12	46	2.91	33.03
> 12	67	7.77	31.26
Overall	**1786**	**2.29**	**48.08**

7.3.1 Observations

The RMSE value measures the error in the units of the quantity predicted - number of visits in this case. For instance, the holdout predictions for the customer segment with less than equal to 3 calibration period frequency deviates from the actual value, on an average, by 1.4 visits. This error is relatively small in the segments with > 10 calibration frequency as the MAPE is just above 30%. **Overall, predictions deviate from the actual value by 48%.**

7.3.2 Baseline

The baseline model projects the holdout visits for an individual based on their average visits per unit time in the calibration period. The baseline model assumes that the visit frequency for an individual does not change in the calibration and holdout period. That is, if a customer made x visits in a duration of T - t_0 days, where T is the end of calibration period and t_0 is the first visit date, then the expected number of visits during holdout for an individual is given by the below equation:

$$E(holdout\,visits|x, T, t_0) = \frac{x * holdout\,duration\,length}{T - t_0}$$

The below table reports the error in predictions by the Baseline model:

Calibration frequency segment	Customer count	RMSE	MAPE(%)
<=3	1276	2.12	70.11
4-6	299	2.39	73.06
7-9	98	2.91	78.49
10-12	46	2.82	41.42
> 12	67	5.17	31.69
Overall	**1786**	**2.42**	**68.47**

The overall MAPE for the baseline model is 68.5% compared to the 48% for the BG/NBD model

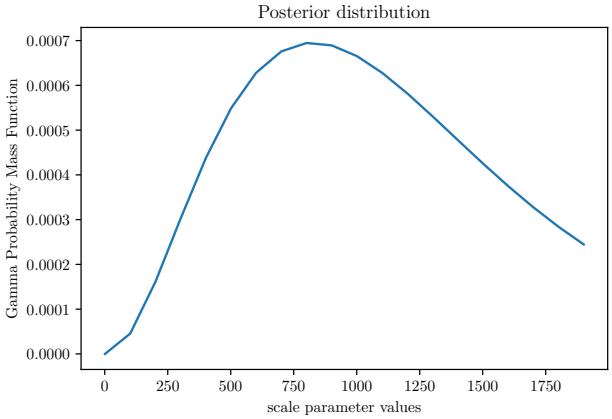
7.4 Fit Gamma-Gamma model for average visit spend prediction

The GammaGammaFitter() function in the *lifetimes* module is used to fit the average spend prediction model. Refer to the Gamma-Gamma model fitting code.

The confidence interval and standard error for the fitted parameters are tabulated below:

	coef	se(coef)	lower 95% bound	upper 95% bound
p	2.497450	0.174225	2.155968	2.838931
q	3.230153	0.160882	2.914824	3.545481
v	370.217995	41.786305	288.316837	452.119152

The v parameter estimated by the GammaGammaFitter function is actually the γ parameter discussed in the original paper (as shown in the formula for the conditional expected average spend value in the theory section). The shape parameter p determines the number of events. The posterior distribution for the scale parameter v ~ Gamma(3.23,370.21). This is shown below:



After fitting the Gamma-Gamma submodel, the estimated parameters are used to get the expected value of mean spend per visit for each customer. Refer to the code for estimating expected spend per visit here. Finally, the expected value of mean spend per visit is multiplied with the expected number of purchases projected by the BG/NBD model to obtain net expected value for each customer.

7.5 Validation

For validation of the spend model, the observed average spend (i.e. Monetary value) in the hold-out period is compared with the Expected mean spend estimated by the model. This is done in two steps:

1) The sample of expected means is compared with the sample of observed means for all customers using a hypothesis test to test if the difference in samples is statistically significant. The null hypothesis H_0 being that the sample means are equal. A two-tailed Welch t-test is used for comparing the sample means

2) The Root Mean Square Error in prediction (calculated the same way as discussed for the frequency model) is calculated between the expected mean spend from the model and the observed average spend per visit

In both steps, to remove the effect of outliers, the sample is filtered for customers with a positive observed average spend < 1500

Refer to the T-test code here. A borderline p-value of 0.04 is obtained and thus the null hypothesis cannot be convincingly rejected.

The distribution plot in figure 13 compares the distribution of observed average spend per visit

in the sample and the conditional expected mean spend per visit estimated by the model. Additionally, the bar plot shows the RMSE in average spend prediction calculated across customer segments based on the calibration frequency (**on the x-axis**).

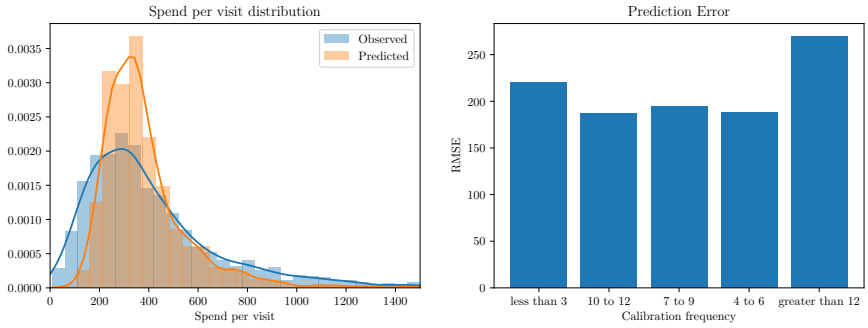


Fig 13. Spend per visit prediction validation

There is an average prediction error of about GBP 200 ((averaged over all customers) in the expected spend per visit, which is quite high considering the average spend per visit. The distribution of the predicted spend is shifted slightly to the right compared to the observed average spend. Also, the observed distribution has a higher variance than the expected mean spend distribution estimated by the model.

Finally, the total expected spend in the holdout period is calculated by multiplying the expected number of visits predicted by the BG/NBD model with the expected spend per visit predicted by the Gamma-Gamma submodel. Refer to the code for total expected value in the holdout period.

8 Prioritization using expected future value predictions

One of the primary objectives of predicting future value of a customer is to identify high-value customers and prioritize them for marketing efforts. For the purpose of this exercise, customers with the actual total spend > 1000 in the holdout period are flagged as *High Value* customers.

The total expected spend calculated by combining the results of the two models is used to rank the customers in deciles i.e. the top 10% customers with the highest expected total spend are placed in the first decile, the next 10% in the 2nd decile and so on. The proportion of actual *High Value* customers that fall into each decile is reported.

8.1 Baseline

For the baseline prioritization model, total spend in the holdout period is projected as the product of the average spend per day in the calibration period and the length of the holdout duration in days. For instance, for a customer who has spent a total S units in a duration of t days in the calibration period, the baseline projection for total spend in the holdout period is calculated as:

$$E(Total\ spend\ baseline|S,T,t_0) = \frac{S * T_H}{t}$$

Where t = T - t_0, is the number of days between first visit date and the calibration period end date. T_H is the length of holdout duration in days.

Customers are also ranked into deciles based on projections made by the baseline model. The lift in targeting obtained by using the model projections for prioritization is reported against the baseline projections. Refer to the code for rank-ordering customers into decile based on the model and the baseline formula.

8.2 Lift

The model lift is now calculated based on the cumulative proportion of true high value customers captured within the deciles.

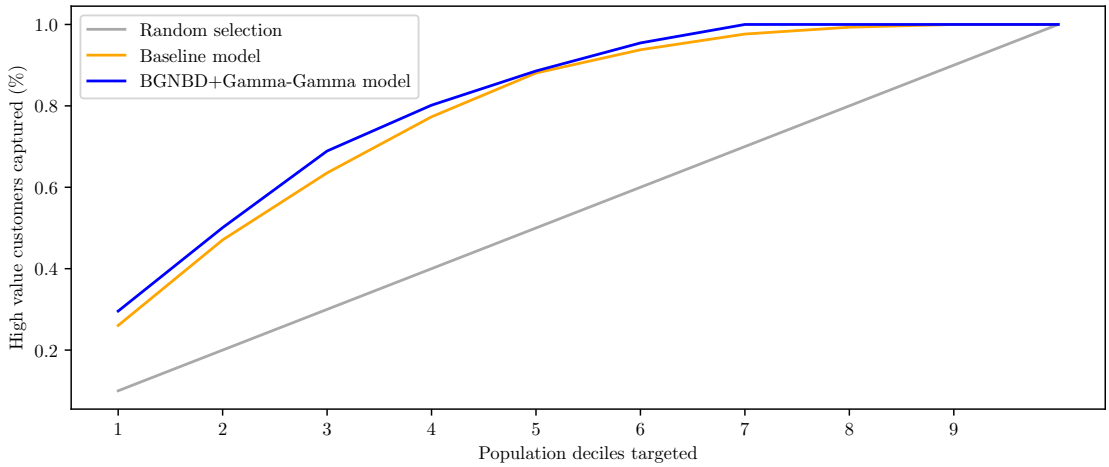


Fig 14. Model Lift Chart

Interpretation

If customers are segmented into priority deciles based on the model predictions, then targeting the top 3 deciles - 30% of the popuation - will capture about 69% of the true high value customers. This results in a lift of 2.3x compared to random selection. The baseline model captures ~ 63% of the true high value customers by targeting the top 3 deciles. Therefore, the lift obtained compared to the baseline model is 9.5% or 1.095x.

9 Results

The BG/NBD model projects the expected number of visits in the future period with an error margin of 48%. The error margin is 68% if projections are made by extrapolating the average calibration frequency (Baseline model). The errors in prediction can be reduced with longer calibration period lengths.

The Gamma-Gamma model predicts the expected spend per visit for each customer. The error in model prediction is about £200 compared to the observed spend per visit in the holdout sample.

Together, the expected number of visits is multiplied by the expected spend per visit to project the net expected value for each customer in the holdout period. The model predictions provide a lift of 2.3x compared to random selection and 9.5% compared to the baseline model in capturing *High value customers* in the first 3 deciles (i.e. 30% of the population). This result is satisfactory and can be useful in correctly prioritizing customers for future investments.

Appendix (Codes)

import code

```
source = pd.read_csv("data.csv",encoding = "ISO-8859-1", engine='python')
print('Total customers in data: ',source['CustomerID'].nunique())
print('Total transactions in data: ',source[source['Quantity']>0]['InvoiceNo'].
 ↪nunique())
```

```
Total customers in data:  4372
Total transactions in data:  20728
```

here

Column name	Data type	Description	Non-missing values (%)
InvoiceNo	Char	Unique Identifier for a transaction	100
StockCode	Char	Unique Identifier for an item in invoice	100
Description	Char	Item description	99.7
Quantity	Int	Item units purchased in an invoice	100
InvoiceDate	Char	Date of purchase	100
UnitPrice	Float	Per unit cost of item	100
CustomerID	Float	Unique Customer Identifer	75
Country	Char	Customer location	100

python utilities

```
import pandas as pd, numpy as np, scipy, matplotlib as mpl,matplotlib.pyplot as␣
 ↪plt,seaborn as sns,statsmodels.api as sm, \
lifetimes,sys,gc, os, warnings,IPython
warnings.filterwarnings('ignore')
```

```
from matplotlib import rc
from lifetimes.plotting import *
from lifetimes.utils import *
from scipy import stats
from scipy.stats import poisson
```

data aggregation code

```
# Create a copy of source data for working
df = source.copy()
# Remove records with missing customer IDs and order cancellation records
df['InvoiceDate'] = pd.to_datetime(pd.to_datetime(df['InvoiceDate'],format='%m/
 ↪%d/%Y %H:%M').dt.date)
df = df[pd.notnull(df['CustomerID'])]
df = df[(df['Quantity']>0)]
df['Sales'] = df['Quantity'] * df['UnitPrice']
```

```
# Keep columns of interest
cols_of_interest = ['CustomerID', 'InvoiceDate', 'Sales']
df = df[cols_of_interest]
# Aggregate data
agg = df.groupby('CustomerID').agg({'InvoiceDate':'nunique','Sales':'sum'}). \
      reset_index()
agg.columns = agg.columns.get_level_values(0)
agg.columns = ['CustomerID','CountVisits','TotalSales']
# Categorize customers based on visits
agg['CountVisitsBin'] = agg['CountVisits'].apply(lambda x: 'single visit' if␣
 ↪x==1 else 'multiple visits')
agg['CountVisitsClusters'] = agg['CountVisits'].apply(lambda x: '1 visit' if␣
 ↪x==1 \
                                                    else ('2-5 visits' if (2<=x) &␣
 ↪(x<=5) \
                                                    else ('6-10 visits' if (6<=x) &␣
 ↪(x<=10) \
                                                    else('11-20 visits' if (11<=x)␣
 ↪& (x<=20) else 'gt20 visits'))))

# Average Spend per visit
agg['AvgSpendPerVisit'] = agg['TotalSales']/agg['CountVisits']
agg.head()
```

Below is a snapshot of the aggregated data:

```
   CustomerID  CountVisits  TotalSales   CountVisitsBin CountVisitsClusters  \
0     12346.0            1    77183.60     single visit             1 visit
1     12347.0            7     4310.00  multiple visits         6-10 visits
2     12348.0            4     1797.24  multiple visits          2-5 visits
3     12349.0            1     1757.55     single visit             1 visit
4     12350.0            1      334.40     single visit             1 visit

   AvgSpendPerVisit
0      77183.600000
1        615.714286
2        449.310000
3       1757.550000
4        334.400000
```

summary statistics for the number of visits for repeat visitors

```
mean        5.454122
std         6.930419
min         2.000000
25%         2.000000
50%         4.000000
```

```
75%          6.000000
max        132.000000
```

summary statistics for *visits per month*

```
Visits per month summary table
mean         1.396369
std          4.088515
min          0.166322
25%          0.463506
50%          0.707834
75%          1.217475
max         60.873750
```

code for calculating average spend over subsequent visits

```
# Create column for sequence of visits
agg_0 = df.groupby(['CustomerID','InvoiceDate'])[['Sales']].sum().reset_index().
 ↪rename(columns={'Sales':'VisitSpend'})
agg_0.
 ↪sort_values(by=['CustomerID','InvoiceDate'],ascending=[True,True],inplace=True)
agg_0['visitsequence'] = agg_0.groupby('CustomerID')['InvoiceDate'].
 ↪rank("dense",ascending=True)
# Cap/remove outliers
percentiles = agg_0['VisitSpend'].quantile([0.01,0.99]).values
agg_0['VisitSpend'][agg_0['VisitSpend'] <= percentiles[0]] = percentiles[0]
agg_0['VisitSpend'][agg_0['VisitSpend'] >= percentiles[1]] = percentiles[1]
# Map count visits cluster
agg_0 = agg_0.
 ↪merge(agg[['CustomerID','CountVisitsClusters']],how='left',on='CustomerID')
# Calculate average spend for nth transaction
plotdf1 = agg_0.groupby(['visitsequence','CountVisitsClusters'])[['VisitSpend']].
 ↪mean().reset_index().rename(columns={'VisitSpend':'AvgVisitSpend'})
plotdf1 = plotdf1[plotdf1['CountVisitsClusters'] != '1 visit']
```

code for visit cycle summary

```
# Calculate visit frequency for repeat buyers
custrec = agg_0.groupby('CustomerID').agg({'InvoiceDate':
 ↪['nunique','max','min']}).reset_index()
custrec.columns = custrec.columns.get_level_values(0)
custrec.columns = ['CustomerID','CountVisits','MaxDate','MinDate']
custrec['Recency'] = (custrec['MaxDate'] - custrec['MinDate'])/np.timedelta64(1,␣
 ↪'M')
custrec['VisitsPerMonth'] = custrec['CountVisits']/custrec['Recency']
custrec[custrec['CountVisits']>1]['VisitsPerMonth'].describe()
```

RFM summary code

```
rfmdata = summary_data_from_transaction_data(df, 'CustomerID', 'InvoiceDate',␣
 ↪monetary_value_col='Sales', observation_period_end='2011-12-09')
rfmdata.head()
```

```
rfmdata.describe()
```

code for calibration and holdout data creation

```
# Filter transactions for customers who visited in the calibration period
customers = pd.DataFrame(df[df['InvoiceDate']<pd.
 ↪to_datetime('2011-08-08',format='%Y-%m-%d')]['CustomerID'].
 ↪unique(),columns=['CustomerID'])
input_tran = df.groupby(['CustomerID','InvoiceDate'])[['Sales']].sum().
 ↪reset_index().merge(customers,how='inner',on='CustomerID')
# Summarize transactions into RFM metrics
from lifetimes.utils import calibration_and_holdout_data
summary_df = calibration_and_holdout_data(input_tran, 'CustomerID',␣
 ↪'InvoiceDate',
                                          calibration_period_end='2011-08-08',
                                          observation_period_end='2011-12-09',
                                           monetary_value_col='Sales')
# Remove customers who visited only once in the calibration period
summary_df = summary_df[summary_df['frequency_cal'] >0]
print(summary_df.head())
```

here

"T_cal" is the duration between first visit and the end of calibration period

```
            frequency_cal  recency_cal  T_cal  monetary_value_cal  \
CustomerID
12347.0               4.0        238.0  244.0            519.7675
12348.0               2.0        110.0  235.0            297.2200
12352.0               3.0         34.0  173.0            421.7700
12356.0               1.0         80.0  202.0            481.4600
12359.0               2.0        142.0  208.0           1474.1150
```

```
            frequency_holdout  monetary_value_holdout  duration_holdout
CustomerID
12347.0                   2.0              759.570000               123
12348.0                   1.0              310.000000               123
12352.0                   3.0              314.743333               123
12356.0                   1.0               58.350000               123
12359.0                   1.0             2876.850000               123
```

code for fitting BG/NBD model

```
# Import model
from lifetimes import BetaGeoFitter
from lifetimes.plotting import plot_period_transactions
from lifetimes.plotting import plot_probability_alive_matrix,␣
 ↪plot_frequency_recency_matrix
# Fit Model
bgnbd = BetaGeoFitter(penalizer_coef=0.0)
bgnbd.fit(summary_df['frequency_cal'], summary_df['recency_cal'],␣
 ↪summary_df['T_cal'],verbose=True)
```

```
Optimization terminated successfully.
         Current function value: -2.590181
         Iterations: 29
         Function evaluations: 30
         Gradient evaluations: 30

<lifetimes.BetaGeoFitter: fitted with 1786 subjects, a: 0.02, alpha: 119.36, b:
0.16, r: 2.38>
```

```
bgnbd.summary
```

training RMSE code

```
#Predict purchases for the calibration duration
cal_df = summary_df[['frequency_cal','recency_cal','T_cal']]
cal_df['actual_calibration_visits'] = cal_df['frequency_cal']+1
cal_df['predicted_calibration_visits'] = bgnbd.
 ↪predict(cal_df['T_cal'],cal_df['frequency_cal'],cal_df['recency_cal'],
    cal_df['T_cal'])
# Calculate RMSE of predictions
import math
a1 = np.array(cal_df['actual_calibration_visits'])
a2 = np.array(cal_df['predicted_calibration_visits'])
print('The RMSE for predictions in calibration period is: ',np.around(math.
 ↪sqrt(np.mean(np.square(np.subtract(a1,a2)))),decimals=2))
```

```
The RMSE for predictions in calibration period is:  1.97
```

code to generate probability$_a$*live*

```
from lifetimes.plotting import plot_history_alive
mpl.rcParams['figure.dpi']= 100
rc('text', usetex=False)
fig = plt.figure(figsize=(10,4))
days_since_birth = 300
sp_trans = input_tran.loc[input_tran['CustomerID'] == 12352.0]
plot_history_alive(bgnbd, days_since_birth, sp_trans, 'InvoiceDate');
```

code for calculating RMSE and MAPE for predictions

```
# Calculate length of holdout period (in days)
holdout_begin = '2011-08-09'
holdout_end = '2011-12-09'
t_timedelta = pd.to_datetime(holdout_end,format="%Y-%m-%d") - pd.
 ↪to_datetime(holdout_begin,format="%Y-%m-%d")
t_days = t_timedelta.days
# Predict holdout frequency using the model
holdout_data =␣
 ↪summary_df[['frequency_cal','frequency_holdout','duration_holdout',
    'monetary_value_holdout']]
holdout_data['frequency_predicted'] = bgnbd.
 ↪conditional_expected_number_of_purchases_up_to_time(t_days,␣
 ↪summary_df['frequency_cal'], summary_df['recency_cal'], summary_df['T_cal'])
holdout_data.reset_index(inplace=True)
holdout_data.head()
```

	CustomerID	frequency_cal	frequency_holdout	duration_holdout	\
0	12347.0	4.0	2.0	123	
1	12348.0	2.0	1.0	123	
2	12352.0	3.0	3.0	123	
3	12356.0	1.0	1.0	123	
4	12359.0	2.0	1.0	123	

	monetary_value_holdout	frequency_predicted
0	759.570000	2.119084
1	310.000000	1.356512
2	314.743333	1.757650
3	58.350000	0.812331
4	2876.850000	1.556559

```
# Calculate total spend in calibration period at customer level
spend_data = input_tran[input_tran['InvoiceDate']<pd.
 ↪to_datetime('2011-08-08',format='%Y-%m-%d')].groupby('CustomerID')[['Sales']].
 ↪sum().reset_index()
# Calculate error metrics
```

```
error_table = holdout_data.copy()
error_table['squared_error'] = (error_table['frequency_predicted'] -␣
 ↪error_table['frequency_holdout'])**2
error_table['absolute_percentage_error'] =␣
 ↪abs(error_table['frequency_predicted']/error_table['frequency_holdout'] -1)*100
error_table.replace(np.inf,np.nan,inplace=True)
error_table['cal_frequency_buckets'] = error_table['frequency_cal'].apply(lambda␣
 ↪x: "<=3" if x<=3 else \
                                        ("4 to 6" if 4<=x<=6 else("7 to 9" if␣
 ↪7<=x<=9 else \
                                        ("10 to 12" if 10<=x<=12 else ">12"))))
# Merge total calibration spend data
error_table = error_table.merge(spend_data,how='left',on='CustomerID')
#Aggregate
error_table_summary = error_table.groupby('cal_frequency_buckets').
 ↪agg({'CustomerID':'nunique','Sales':'sum','squared_error':
 ↪'mean','absolute_percentage_error':'mean'}).reset_index()
error_table_summary.columns = error_table_summary.columns.get_level_values(0)
error_table_summary.columns = ['Calibration frequency segment','Customer␣
 ↪count','Total Revenue in calibration period','MSE of holdout␣
 ↪prediction','MAPE(%) of holdout prediction']
error_table_summary['RMSE of holdout prediction'] = (error_table_summary['MSE of␣
 ↪holdout prediction'])**(1/2)
error_table_summary['S.No'] = [4,2,3,1,5]
error_table_summary = error_table_summary.set_index(error_table_summary['S.No']).
 ↪sort_index()
error_table_summary = error_table_summary[['Calibration frequency␣
 ↪segment','Customer count','Total Revenue in calibration period','RMSE of␣
 ↪holdout prediction','MAPE(%) of holdout prediction']]
newrow = {'Calibration frequency segment':'Overall','Customer count':np.
 ↪sum(error_table_summary['Customer count']),'Total Revenue in calibration␣
 ↪period':np.sum(error_table_summary['Total Revenue in calibration␣
 ↪period']),'RMSE of holdout prediction':math.sqrt(np.
 ↪mean(error_table['squared_error'])),'MAPE(%) of holdout prediction':np.
 ↪mean(error_table['absolute_percentage_error'])}
error_table_summary = error_table_summary.append(newrow, ignore_index = True)
display(error_table_summary)
```

Gamma-Gamma model fitting code

```
from lifetimes import GammaGammaFitter
ggf = GammaGammaFitter(penalizer_coef = 0.0)
ggf.fit(summary_df['frequency_cal'],
        summary_df['monetary_value_cal'])
print(ggf)
```

```
<lifetimes.GammaGammaFitter: fitted with 1786 subjects, p: 2.50, q: 3.23, v:
```

370.22>

code for estimating expected spend per visit

```
ExpSpenddf = pd.DataFrame(ggf.
 ↪conditional_expected_average_profit(summary_df['frequency_cal'],
    summary_df['monetary_value_cal']))
ExpSpenddf = ExpSpenddf.reset_index()
ExpSpenddf.columns = ExpSpenddf.columns.get_level_values(0)
ExpSpenddf.columns = ['CustomerID','ExpSpendPerVisit']
ExpSpenddf.head()
```

```
   CustomerID  ExpSpendPerVisit
0     12347.0        500.572644
1     12348.0        333.448769
2     12352.0        420.123222
3     12356.0        449.915760
4     12359.0       1147.072231
```

T-test code here

```
from scipy import stats
ObservedMonetaryValue =  holdout_data[(holdout_data['monetary_value_holdout']>0)␣
 ↪& (holdout_data['monetary_value_holdout']<1500)]['monetary_value_holdout']
PredictedMonetaryValue = ␣
 ↪holdout_data[(holdout_data['monetary_value_holdout']>0) &␣
 ↪(holdout_data['monetary_value_holdout']<1500)]['ExpSpendPerVisit']
stats.ttest_ind(ObservedMonetaryValue,PredictedMonetaryValue,equal_var=False)
```

```
Ttest_indResult(statistic=2.049278303836273, pvalue=0.04054076520590877)
```

code for total expected value in the holdout period

```
# Calculate the expected net value in holdout
holdout_data['ExpectedTotalValue'] =␣
 ↪holdout_data['frequency_predicted']*holdout_data['ExpSpendPerVisit']
holdout_data.head()
```

```
   CustomerID  frequency_cal  frequency_holdout  duration_holdout  \
0     12347.0            4.0                2.0               123
1     12348.0            2.0                1.0               123
2     12352.0            3.0                3.0               123
3     12356.0            1.0                1.0               123
4     12359.0            2.0                1.0               123

   monetary_value_holdout  frequency_predicted  BaselineExpVisits_holdout  \
0              759.570000             2.119084                   2.500000
1              310.000000             1.356512                   1.557447
```

```
2              314.743333            1.757650                 2.820809
3               58.350000            0.812331                 1.207921
4             2876.850000            1.556559                 1.759615

   ExpSpendPerVisit  ExpectedTotalValue
0        500.572644         1060.755387
1        333.448769          452.327202
2        420.123222          738.429379
3        449.915760          365.480524
4       1147.072231         1785.485679
```

code for rank-ordering customers into decile based on the model and the baseline formula

```
# Total holdout value - Actual
df_holdout = df[df['InvoiceDate']>pd.to_datetime('2011-08-08',format='%Y-%m-%d')]
actual_val_holdout = df_holdout.groupby(['CustomerID'])[['Sales']].sum().
 ↪reset_index().rename(columns={'Sales':'ActualValue'})
actual_val_holdout['HighValueCustomer'] = actual_val_holdout['ActualValue'].
 ↪apply(lambda x: 1 if x>1000 else 0)
# Calculate average spend value in the calibration period
df_cal_summary = input_tran.groupby(['CustomerID']).agg({'Sales':
 ↪'sum','InvoiceDate':'nunique'}).reset_index()
df_cal_summary.columns = df_cal_summary.columns.get_level_values(0)
df_cal_summary.columns = ['CustomerID','TotalSpendCal','TotalVisitsCal']
df_cal_summary['SpendPerVisit_Cal'] = df_cal_summary['TotalSpendCal']/
 ↪df_cal_summary['TotalVisitsCal']

# error_table['BaselineExpVisits_holdout']
# Merge High value customer flag and avg spend per visit in calibration
liftdata = holdout_data[['CustomerID','ExpectedTotalValue']]
liftdata = liftdata.
 ↪merge(actual_val_holdout[['CustomerID','HighValueCustomer']],how='left',
on='CustomerID').
 ↪merge(df_cal_summary[['CustomerID','SpendPerVisit_Cal']],how='left',
on='CustomerID')
# Merge expected holdout visits from the Baseline model in the previous section
liftdata = liftdata.
 ↪merge(error_table[['CustomerID','BaselineExpVisits_holdout']],how='left',
on='CustomerID')
# Calculate total value by baseline
liftdata['TotalValue_Baseline'] =␣
 ↪liftdata['BaselineExpVisits_holdout']*liftdata['SpendPerVisit_Cal']
# Create deciles based on model and baseline predictions
liftdata['decile_model'] = rank_quantile(liftdata,'ExpectedTotalValue')
liftdata['decile_baseline'] = rank_quantile(liftdata,'TotalValue_Baseline')
liftdata.head()
```

	CustomerID	ExpectedTotalValue	HighValueCustomer	SpendPerVisit_Cal	\
0	12347.0	1060.755387	1.0	615.714286	
1	12348.0	452.327202	0.0	449.310000	
2	12352.0	738.429379	0.0	358.005714	
3	12356.0	365.480524	0.0	937.143333	
4	12359.0	1785.485679	1.0	1593.145000	

	BaselineExpVisits_holdout	TotalValue_Baseline	decile_model	\
0	2.500000	1539.285714	3	
1	1.557447	699.776426	7	
2	2.820809	1009.865830	4	
3	1.207921	1131.994917	8	
4	1.759615	2803.322452	1	

	decile_baseline
0	3
1	6
2	4
3	4
4	1

References

[1] Sunil Gupta PhD Donald R. Lehmann PhD (2006) *Customer Lifetime Value and Firm Valuation*, Journal of Relationship Marketing, 5:2-3, 87-110, DOI: 10.1300/J366v05n02_06 .

[2] Fader, P. S., Hardie, B. G., Lee, K. (2005). *"Counting Your Customers" the Easy Way: An Alternative to the Pareto/NBD Model*. Marketing Science, 24 (2), 275-284.

[3] Fader, P. S.,Hardie, Bruce G. S. (2013) *"The Gamma-Gamma Model of Monetary Value."*. Semantic Scholar, Corpus ID: 36160098.

[4] Richard Colombo, Weina Jiang (1999) *A Stochastic RFM Model*, Journal of Interactive Marketing, 1999