The University of Texas at Dallas

School of Management

BUAN 6383/MIS 6386 Modeling for Business Analytics

Syam Menon

Group: 07

Noyal Jonnalagadda
Yagnika Kancherla
Sreeja Kandula
Keerthi Kolli
Shreya Shamarthi

Project 01

1. **The Poisson Model**
   Consider the example related to billboard exposures from class. The associated data is in the file billboard.csv. Write code to estimate the parameters of the Poisson model using maximum likelihood estimation (MLE). Report your code, the estimated parameters and the maximum value of the log-likelihood.

Reading the data from billboard csv file

In [1]:
```
import os
import math
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize, least_squares
from scipy.stats import poisson, gamma
```

Question 1

In [35]:
```
billboard = pd.read_csv('billboard.csv')
billboard.head()
```

Out[35]:

	EXPOSURES	PEOPLE
0	0	48
1	1	37
2	2	30
3	3	24
4	4	20

In [99]:
```
a1 = np.array(billboard.loc[:,'PEOPLE'])#define the first array that will go through the poisson_model function
a2 = np.array(billboard.loc[:,'EXPOSURES'])#define the second array that will go through poisson_model
[a2,a1]
```

Out[99]:
```
[array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23]),
 array([48, 37, 30, 24, 20, 16, 13, 11,  9,  7,  6,  5,  5,  3,  3,  2,  2,
         2,  1,  1,  2,  1,  1,  1])]
```

Possession model

In [102]:
```
def poisson_model(lamda, a1, a2):
    ll = 0 #we create a variable that will store the cummulative sum of the iteration below
    for i in range(len(a2)):
        prob = poisson.pmf(k=a2[i], mu=lamda)
        ll += a1[i]*np.log(prob)
    return -1*ll
soln = minimize(
    poisson_model,
    args = (a1, a2),
    x0 = np.array((1)),
    bounds=[(0.000001,None)],
    tol=1e-10,
    options={'ftol' : 1e-8},
)
```

```
/var/folders/z4/v_l0j_d955xggmv5v8_n36640000gn/T/ipykernel_44863/2739926952.py:5: RuntimeWarning: divide by zero en
countered in log
  ll += a1[i]*np.log(prob)
```

In [104]:
```
lamda = 1
[poisson_model(lamda, a1, a2),
soln]
```

Out[104]:
```
[1729.640056651523,
      fun: array([929.04388273])
      jac: array([-7.62939453e-06])
  message: 'Optimization terminated successfully'
     nfev: 19
      nit: 7
     njev: 7
   status: 0
  success: True
        x: array([4.45600006])]
```

The estimated parameters and the maximum value of the log-likelihood.

In [105]:
```
lmbda_result = soln.x[0]
z = 0
for i in range(len(a1)):
    z += a1[i]*np.log(poisson.pmf(a2[i],lmbda_result))

print("Lambda value for poisson model after optimzation:{}".format(lmbda_result))
print("MAX_LL:{}".format(c))
```

```
Lambda value for poisson model after optimzation:4.456000057631195
MAX_LL:-929.0438827272923
```

2. **The NBD Model**

Next, write code (for the same dataset) to estimate the parameters of the NBD model using MLE. Report your code, the estimated parameters and the maximum value of the log-likelihood. Evaluate the NBD model vis-`a-vis the Poisson model; explain which is better and why.

The NBD Model

In [109]:
```
def nbd(params, a1, a2):
    shape, scale = params
    ll = 0
    prob = []
    for i in range(len(a2)):
        k = a2[i]
        if i == 0:
            prob.append((scale/(scale+1))**shape)
        else:
            prob.append(((shape+k-1)/(k*(scale+1)))*prob[i-1])
        ll += a1[i]*np.log(prob[i])
    return -1*ll
soln = minimize(
    nbd,
    method='SLSQP',
    args = (a1, a2),
    x0 = np.array((10, 10)),
    bounds=[(0.000000001, None), (0.000000001, None)],
    tol=1e-10,
    options={'ftol' : 1e-8},
)
```

```
/var/folders/z4/v_l0j_d955xggmv5v8_n36640000gn/T/ipykernel_44863/2080087608.py:11: RuntimeWarning: divide by zero e
ncountered in log
  ll += a1[i]*np.log(prob[i])
```

In [110]:
```
params = (0.6, 0.6)
[nbd(params, a1, a2), soln]
```

Out[110]:
```
[868.3794964318045,
      fun: 649.6888274839724
      jac: array([-0.00040436,  0.0009079 ])
  message: 'Optimization terminated successfully'
     nfev: 102
      nit: 28
     njev: 28
   status: 0
  success: True
        x: array([0.96925708, 0.21751751])]
```

The estimated parameters and the maximum value of the log-likelihood.

```
In [15]: params = (soln.x[0], soln.x[1])
         max_ll = -1*nbd(params, a1, a2)
         print('shape:', soln.x[0], '\nscale:', soln.x[1], '\nMAX_LL:', max_ll)

         shape: 0.9692570782518828
         scale: 0.21751751012440285
         MAX_LL: -649.6888274839724
```

The log-likelihood value of a regression model is a way to measure the goodness of fit for a model. The higher the value of the log-likelihood, the better a model fits a dataset. The log-likelihood value for a given model can range from negative infinity to positive infinity. The actual log-likelihood value for a given model is mostly meaningless, but it's useful for comparing two or more models.

We often fit several regression models to a dataset and choose the model with the highest log-likelihood value as the model that fits the data best. So the NBD Model has higher Log likelihood value(-649.6888) than the Poisson model, we suggest that the NBD model is a good fit for the given data set.

3. **The Poisson Regression**

Now consider the khakichinos.com example from class; The associated data is in the file khakichinos.csv. Estimate all relevant parameters for Poisson regression using MLE. Report your code, the estimated parameters and the maximum value of the log-likelihood.

Reading the data set.

```
In [17]: khakichinos = pd.read_csv('khakichinos.csv')
         khakichinos.tail()
```

Out[17]:

	ID	NumberofVisits	LnInc	Sex	LnAge	HHSize
2723	2724	0	9.528794	1	2.944439	2
2724	2725	0	11.379394	0	3.970292	2
2725	2726	0	11.191342	1	3.044522	3
2726	2727	0	10.532096	1	2.890372	4
2727	2728	0	11.736069	1	2.833213	3

Standardizing the data with standradscaler.

In [19]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x = sc.fit_transform(khakichinos.drop(columns=['a1', 'ID']))
khakichinos[khakichinos.drop(columns=['a1', 'ID']).columns] = x
khakichinos.head()
```

Out[19]:

	ID	NumberofVisits	LnInc	Sex	LnAge	HHSize
0	1	0	0.761947	0.752938	0.612645	-0.788124
1	2	5	-1.930956	0.752938	1.011624	-1.496993
2	3	0	0.264588	-1.328131	-0.638724	-0.788124
3	4	0	-0.014926	0.752938	0.798477	-0.079254
4	5	0	-0.014926	0.752938	-1.797214	-0.079254

Possession Regression

In [113]:
```
def Possion_Regression(params, a1, a2, a3, a4, a5):
    lambda_0, b1, b2, b3, b4 = params
    ll = []
    for i in range(len(a1)):
        k = a1[i]
        regr = (b1*a2[i]+b2*a3[i]+b3*a4[i]+b4*a5[i])
        lambda_i = float(lambda_0*np.exp(float(regr)))
        ll.append(k*(np.log(lambda_0)+regr)-lambda_i-round(float(np.log(float(math.factorial(k)))), 3))
    return -1*sum(ll)
soln = minimize(
    Possion_Regression,
    method='SLSQP',
    args = (a1, a2, a3, a4, a5),
    x0 = np.array((0.1,0.1,0.1,0.1,0.1)),
    tol=1e-10
)
```

```
/var/folders/z4/v_l0j_d955xggmv5v8_n36640000gn/T/ipykernel_44863/2918586287.py:7: RuntimeWarning: overflow encounte
red in exp
  lambda_i = float(lambda_0*np.exp(float(regr)))
/var/folders/z4/v_l0j_d955xggmv5v8_n36640000gn/T/ipykernel_44863/2918586287.py:7: RuntimeWarning: overflow encounte
red in double_scalars
  lambda_i = float(lambda_0*np.exp(float(regr)))
```

In [114]:
```
params = (0.1,0.1,0.1,0.1,0.1)
[Possion_Regression(params, a1, a2, a3, a4, a5),soln]
```

Out[114]:
```
[9848.96971560914,
      fun: 6291.472631965174
      jac: array([-0.00189209, -0.00189209, -0.00067139, -0.00189209, -0.00115967])
  message: 'Optimization terminated successfully'
     nfev: 126
      nit: 16
     njev: 16
   status: 0
  success: True
        x: array([ 0.91550775,  0.05607705,  0.00204699,  0.2533736 , -0.05065456])]
```

The estimated parameters and the maximum value of the log-likelihood after optimization.

lambda_0: 0.9155077489393663

beta1: 0.05607705109807718

beta2: 0.0020469868733725854

beta3: 0.253373597335255

beta4: -0.05065455573962687

MAX_LL: -6291.472631965174

4. The NBD Regression

Consider the khakichinos.com example again. Estimate all relevant parameters for NBD Regression using MLE. Report your code, the estimated parameters and the maximum value of the log-likelihood. Evaluate the NBD regression vis-`a-vis the Poisson regression; explain which is better and why.

Question-4. NBD Regression

In [117]:
```
def NLL_NBR(params, a1, a2, a3, a4, a5):
    shape, scale, b1, b2, b3, b4 = params
    ll = 0
    prob = []
    for i in range(len(a1)):
        k = a1[i]
        regr = (b1*a2[i]+b2*a3[i]+b3*a4[i]+b4*a5[i])
        expo = np.exp(float(regr))
        gam = np.log(math.gamma(shape+k))-np.log(float(math.gamma(shape)))-np.log(float(math.factorial(k)))
        term1 = -1*(shape+k)*np.log(float(scale+expo))
        term2 = shape*np.log(float(scale))+k*regr
        ll += gam+term1+term2
    return -1*ll
soln = minimize(
    NLL_NBR,
    args = (a1, a2, a3, a4, a5),
    method='SLSQP',
    x0 = np.array((1,1,1,1,1,1)),
    tol=1e-10

)
```

```
params = (0.1,0.1,0.1,0.1,0.1,0.1)
[NLL_NBR(params, a1, a2, a3, a4, a5),soln]
```

```
[2921.371991203894,
      fun: 2888.96611373066
      jac: array([ 1.06811523e-03,  6.40869141e-04,  2.74658203e-04, -3.05175781e-04,
         2.74658203e-04,  6.10351562e-05])
  message: 'Optimization terminated successfully'
     nfev: 301
      nit: 36
     njev: 36
   status: 0
  success: True
        x: array([ 0.1387515 ,  0.15400068,  0.0438726 , -0.00445615,  0.3886018 ,
       -0.03430509])]
```

The estimated parameters and the maximum value of the log-likelihood after optimization.

shape: 0.13875150075413273

scale: 0.15400067690092847

beta1: 0.0438725999452609

beta2: -0.004456152254205826

beta3: 0.3886018046447337

beta4: -0.03430508907817904

MAX_LL: -2888.96611373066

The log-likelihood value of a regression model is a way to measure the goodness of fit for a model. The higher the value of the log-likelihood, the better a model fits a dataset. The log-likelihood value for a given model can range from negative infinity to positive infinity. The actual log-likelihood value for a given model is mostly meaningless, but it's useful for comparing two or more models.

We often fit several regression models to a dataset and choose the model with the highest log-likelihood value as the model that fits the data best. So the NBD Model has higher Log likelihood value(-2888.97) than the Poisson model, we suggest that the NBD model is a good fit for the given data set.

5. For each of the models above, can you provide some managerial takeaways?

Poisson model performs better than Negative binomial regression

Covariance type is non robust in both the models

For the Poisson model We got a mean exposure rate (Lambda) of 4.456 exposures per person. The NBD model loosens the requirement that every observer has an identical exposure rate. It is predicated on the idea that each person's exposure rate follows a gamma distribution with alpha (shape and scale) characteristics. This results in a higher log likelihood (-649 as opposed to -929), demonstrating the benefit of assuming heterogeneity.

Even in this model, the variables do not completely account for the variations between people due to the Poisson regression (in this case, to characteristics like income, sex, age, etc., with their own betas). The final regression, known as the NBD regression, uses the same gamma distribution as Q2 but includes all of the Poisson regression's features.

1. Read books.csv and generate two new datasets –
   (a) books01.csv, with the structure of the dataset used in the billboard exposures example (i.e., with only two columns – (i) the number purchases, and (ii) the number of people making the corresponding number of purchases), and

   (b) books02.csv, with the structure of the dataset used in the khakichinos.com example, with a new column containing a count of the number of books purchased from barnesandnoble.com by each customer, while keeping the demographic variables (remember to drop date, product, and price).

   Print the first and last few records of both new datasets.

   Before formatting.

```
df=pd.read_csv('books.csv')
df = df.drop(['date','product','price'],axis=1)
df
```

	userid	education	region	hhsz	age	income	child	race	country	domain	qty
0	11443031	4.0	1.0	2	11.0	4	1	1	0	amazon.com	1
1	11443031	4.0	1.0	2	11.0	4	1	1	0	amazon.com	1
2	11443031	4.0	1.0	2	11.0	4	1	1	0	amazon.com	1
3	11519009	NaN	2.0	3	5.0	3	1	2	0	amazon.com	1
4	11519009	NaN	2.0	3	5.0	3	1	2	0	amazon.com	1
...	...	...	...	...	...	...	...	...	...	...	...
40940	15557019	NaN	2.0	3	6.0	6	1	1	0	amazon.com	1
40941	15557019	NaN	2.0	3	6.0	6	1	1	0	amazon.com	1
40942	15557019	NaN	2.0	3	6.0	6	1	1	0	amazon.com	1
40943	15557019	NaN	2.0	3	6.0	6	1	1	0	amazon.com	1
40944	15562604	NaN	3.0	2	11.0	6	0	1	0	amazon.com	1

a) Data of first and last 5 records of books01.csv

```
In [119]: books01.head()
```

Out[119]:

	Num_purchases	Num_people
0	0	7639
1	1	790
2	2	354
3	3	174
4	4	121

```
In [120]: books01.tail()
```

Out[120]:

	Num_purchases	Num_people
43	56	1
44	58	1
45	63	1
46	86	1
47	111	1

b) Data of first and last 5 records of books02.csv

```
books02
```

Out[52]:

	education	region	hhsz	age	income	child	race	country	Num_purchases
0	5.0	1.0	2.0	11.0	7.0	0.0	1.0	0.0	1
1	2.0	2.0	2.0	8.0	4.0	0.0	1.0	0.0	1
2	4.0	3.0	5.0	10.0	3.0	1.0	1.0	0.0	1
3	NaN	4.0	2.0	10.0	5.0	1.0	1.0	0.0	2
4	NaN	1.0	3.0	8.0	7.0	1.0	1.0	0.0	5
...	...	...	...	...	...	...	...	...	...
9446	NaN	3.0	6.0	6.0	5.0	1.0	1.0	0.0	0
9447	NaN	3.0	3.0	10.0	4.0	1.0	1.0	1.0	0
9448	1.0	2.0	2.0	8.0	6.0	0.0	1.0	0.0	0
9449	NaN	3.0	2.0	3.0	2.0	1.0	1.0	0.0	0
9450	NaN	3.0	2.0	6.0	1.0	0.0	1.0	0.0	0

9451 rows × 9 columns

2. Develop a Poisson model using books01.csv. Report your code, the estimated parameters and the maximum value of the log-likelihood (and any other information you believe is relevant).

Question-2

In [61]:
```
b = np.array(books01[['Num_purchases']])
a = np.array(books01[['Num_people']])
lmbda = 0
soln = minimize(
    poisson_model,
    args = (a, b),
    x0 = np.array((2.0)),
    bounds=[(1e-6,None)],
    tol=1e-10,
    options={'ftol' : 1e-8},
)
lmbda = soln.x[0]
poisson_pd = 0
for i in range(len(a)):
    poisson_pd += a[i]*np.log(poisson.pmf(b[i],lmbda))
print("Lambda {}".format(lmbda))
print("Log likelihood {}".format(poisson_pd))
```

```
Lambda 0.7043698792467571
Log likelihood [-17773.13492767]
```

```
/Users/noelfranklin/opt/anaconda3/lib/python3.9/site-packages/scipy/optimize/optimize.py:282: RuntimeWarning: Value
s in x were outside bounds during a minimize step, clipping to bounds
  warnings.warn("Values in x were outside bounds during a "
/var/folders/z4/v_l0j_d955xggmv5v8_n36640000gn/T/ipykernel_44863/2459367178.py:5: RuntimeWarning: divide by zero en
countered in log
  ll += people_exposed[i]*np.log(prob)
```

The estimated parameters and the maximum value of the log-likelihood
Lambda 0.7043698792467571 Log likelihood [-17773.13492767]

3. Develop a Poisson model using books02.csv, i.e., by ignoring the independent variables available. Report your code, and confirm that the estimated parameters and the maximum value of the log-likelihood are identical to those obtained with the Poisson model developed using books01.csv.

```
In [63]: num_people02 = books02_trans.loc[:,'Num_people']
         num_purchases02 = books02_trans.loc[:,'Num_purchases']
         num_people02.head()

Out[63]: 0    7639
         1     790
         2     354
         3     174
         4     121
         Name: Num_people, dtype: int64

In [64]: lmbda = 0
         soln = minimize(
             poisson_model,
             method='SLSQP',
             args = (num_people02, num_purchases02),
             x0 = np.array((2.0)),
             bounds=[(1e-6,None)],
             tol=1e-10,
             options={'ftol' : 1e-8},
         )
         lmbda = soln.x[0]
         poisson_pd = 0
         for i in range(len(num_people02)):
             poisson_pd += num_people02[i]*np.log(poisson.pmf(num_purchases02[i],lmbda))
         print("Lambda {}".format(lmbda))
         print("Log likelihood {}".format(poisson_pd))

         Lambda 0.7043698792467571
         Log likelihood -17773.134927673065

         /Users/noelfranklin/opt/anaconda3/lib/python3.9/site-packages/scipy/optimize/optimize.py:282: RuntimeWarning: Valu
```

The estimated parameters and the maximum value of the log-likelihood
Lambda 0.7043698792467571
Log likelihood -17773.134927673065

Yes, the estimated parameters and the maximum value of the log-likelihood are identical to those obtained with the Poisson model developed using books01.csv.

4. Develop an NBD model using books01.csv. Report your code, the estimated parameters and the maximum value of the log-likelihood (and any other information you believe is relevant).

Question-4

In [127]:
```
b = np.array(books01[['Num_purchases']])
a = np.array(books01[['Num_people']])
def nbd(params, a, b):
    shape, scale = params
    ll = 0
    prob = []
    for i in range(len(b)):
        k = b[i]
        if i == 0:
            prob.append((scale/(scale+1))**shape)
        else:
            prob.append(((shape+k-1)/(k*(scale+1)))*prob[i-1])
        ll += a[i]*np.log(prob[i])
    return -1*ll
soln = minimize(
    nbd,
    method='SLSQP',
    args = (a, b),
    x0 = np.array((10, 10)),
    bounds=[(0.000000001, None), (0.000000001, None)],
    tol=1e-10,
    options={'ftol' : 1e-8},
)
soln
```

```
/Users/noelfranklin/opt/anaconda3/lib/python3.9/site-packages/scipy/optimize/optimize.py:282: RuntimeWarning: Value
s in x were outside bounds during a minimize step, clipping to bounds
  warnings.warn("Values in x were outside bounds during a "
/var/folders/z4/v_l0j_d955xggmv5v8_n36640000gn/T/ipykernel_44863/3380229210.py:13: RuntimeWarning: divide by zero e
ncountered in log
  ll += a[i]*np.log(prob[i])
```

Out[127]:
```
      fun: array([8233.00000363])
      jac: array([-0.00634766, -0.00231934])
  message: 'Optimization terminated successfully'
     nfev: 235
      nit: 63
```

The estimated parameters and the maximum value of the log-likelihood

shape: 0.10407732707669293

scale: 0.15238332478696814

MAX_LL: [-8233.00000363]

5. Develop an NBD model using books02.csv (again, ignoring the variables available). Report your code, and confirm that the estimated parameters and the maximum value of the log-likelihood are identical to those obtained with the NBD model developed using books01.csv.

Question-5

In [130]:
```
soln = minimize(
    NLL_nbd,
    method='SLSQP',
    args = (num_people02, num_purchases02),
    x0 = np.array((1, 10)),
    bounds=[(1e-36, None), (1e-36, None)],
    tol=1e-10,
    options={'ftol' : 1e-8},
)
soln
```

```
/var/folders/z4/v_l0j_d955xggmv5v8_n36640000gn/T/ipykernel_44863/299203914.py:11: RuntimeWarning: divide by zero en
countered in log
  ll += people_exposed[i]*np.log(prob[i])
```

Out[130]:
```
     fun: 8233.000003626761
     jac: array([0.00024414, 0.00024414])
 message: 'Optimization terminated successfully'
    nfev: 134
     nit: 37
    njev: 34
  status: 0
 success: True
       x: array([0.10407744, 0.15238355])
```

In [131]:
```
params = (soln.x[0], soln.x[1])
max_ll = -1*NLL_nbd(params, num_people02, num_purchases02)
print('shape:', soln.x[0], '\nscale:', soln.x[1], '\nMAX_LL:', max_ll)
```

```
shape: 0.10407743709415807
scale: 0.15238354884015795
MAX_LL: -8233.000003626761
```

The estimated parameters and the maximum value of the log-likelihood are

shape: 0.10407743709415807

scale: 0.15238354884015795

MAX_LL: -8233.000003626761

Yes, the estimated parameters and the maximum value of the log-likelihood are identical to those obtained with the Poisson model developed using books01.csv.

6. Calculate the values of (i) reach, (ii) average frequency, and (iii) gross ratings points (GRPs) based on the NBD model. Show your work.

Reach 30.223547117929574
Average Frequency 4.844356117340861
GRPS 146.4136253684819

Question-6

In [132]:
```
shape = 0.10407743709415807
scale =  0.15238354884015795
MAX_LL = -8233.000003626761
P= (shape/(shape+1))**scale
e= scale/shape
# (i)Reach
print("Reach {}".format(100*(1-P)))
Reach = 100*(1-P)
# (ii)Average Frequency
print("Average Frequency  {}".format(e/(1-P)))
Avg = e/(1-P)
# (iii) GRPS
print("GRPS {}".format(Reach*Avg))
```

```
Reach 30.223547117929574
Average Frequency  4.844356117340861
GRPS 146.4136253684819
```

7. Identify all independent variables with missing values. How many values are missing in each? Drop any variable with many missing values (specify how you are defining 'many'). If the number of missing values are very few (again, specify how you are defining 'few'), delete the rows involved. For the r.emaining variables (if any), replace the missing values with the means of the corresponding variables. Report your code.

We have a total 9451 records in which 6914 entries of the education column are null i.e, 73.15% entries are empty. So, we decided to remove the column. And there are very few (1.16% of total records) we decided to remove.

Question-7

In [79]:
```
books02.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9451 entries, 0 to 9450
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   education      2537 non-null   float64
 1   region         9440 non-null   float64
 2   hhsz           9451 non-null   float64
 3   age            9450 non-null   float64
 4   income         9451 non-null   float64
 5   child          9451 non-null   float64
 6   race           9451 non-null   float64
 7   country        9451 non-null   float64
 8   Num_purchases  9451 non-null   int64  
dtypes: float64(8), int64(1)
memory usage: 664.6 KB
```

In [80]:
```
books02.drop(columns = ['education'], inplace=True)
books02.dropna(inplace=True)
```

In [81]:
```
books02.isna().sum()
```

Out[81]:
```
region           0
hhsz             0
age              0
income           0
child            0
race             0
country          0
Num_purchases    0
dtype: int64
```

8. Incorporate the available customer characteristics and estimate all relevant parameters for Poisson regression using MLE. Report your code, the estimated parameters and the maximum value of the log-likelihood (and any other information you believe is relevant). What are the managerial takeaways — which customer characteristics seem to be important?

Question-8

In [89]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
xvars = sc.fit_transform(books02.drop(columns=['Num_purchases']))
books02[books02.drop(columns=['Num_purchases']).columns] = xvars
books02.head()
```

Out[89]:

	region	hhsz	age	income	child	race	country	Num_purchases
0	-1.443484	-0.899071	1.653716	1.177957	-1.545319	-0.189941	-0.443085	1
1	-0.508917	-0.899071	0.409640	-0.369392	-1.545319	-0.189941	-0.443085	1
2	0.425649	1.404285	1.239024	-0.885176	0.647115	-0.189941	-0.443085	1
3	1.360215	-0.899071	1.239024	0.146391	0.647115	-0.189941	-0.443085	2
4	-1.443484	-0.131286	0.409640	1.177957	0.647115	-0.189941	-0.443085	5

In [90]:
```
a1 = np.array(books02.loc[:,'Num_purchases'])
a2 = np.array(books02.loc[:,'a2'])
a3 = np.array(books02.loc[:,'a3'])
a4 = np.array(books02.loc[:,'a4'])
a5 = np.array(books02.loc[:,'a5'])
a6 = np.array(books02.loc[:,'a6'])
a7 = np.array(books02.loc[:,'a7'])
a8 = np.array(books02.loc[:,'a8'])
```

In [91]:
```
def Possion_Regression(params, a1, a2, a3, a4, a5, a6, a7, a8):
    lambda_0, b1, b2, b3, b4, b5, b6, b7 = params
    ll = []
    for i in range(len(a1)):
        k = a1[i]
        regr = (b1*a8[i]+b2*a7[i]+b3*a6[i]+b4*a5[i]+b5*a4[i]+b6*a3[i]+b7*a2[i])
        lambda_i = float(abs(lambda_0)*np.exp(float(regr)))
        ll.append(k*(np.log(abs(lambda_0))+regr)-lambda_i-round(float(np.log(float(math.factorial(k)))), 3))
    return -1*sum(ll)
```

```
soln = minimize(
    Possion_Regression,
    method='SLSQP',
    args = (a1, a2, a3, a4, a5, a6, a7, a8),
    x0 = np.array((0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1)),
    bounds=None,
    tol=1e-10
)
soln
```

```
/var/folders/z4/v_l0j_d955xggmv5v8_n36640000gn/T/ipykernel_44863/2707789201.py:10: RuntimeWarning: overflow encount
ered in exp
  lambda_i = float(abs(lambda_0)*np.exp(float(regr)))
/var/folders/z4/v_l0j_d955xggmv5v8_n36640000gn/T/ipykernel_44863/2707789201.py:10: RuntimeWarning: overflow encount
ered in double_scalars
  lambda_i = float(abs(lambda_0)*np.exp(float(regr)))
```

```
     fun: 17680.803552203855
     jac: array([0.01464844, 0.01464844, 0.01953125, 0.01342773, 0.01123047,
       0.01123047, 0.00732422, 0.01049805])
 message: 'Optimization terminated successfully'
    nfev: 265
     nit: 23
    njev: 23
  status: 0
 success: True
       x: array([ 0.69606917, -0.10121027, -0.00387679,  0.06814624,  0.02278101,
        0.00376963, -0.0653215 , -0.05698282])
```

The estimated parameters and the maximum value of the log-likelihood

lambda_0: 0.6960691689344141

beta1: -0.10121026890065225

beta2: -0.003876785167961687

beta3: 0.06814623652153202

beta4: 0.022781008606692073

beta5: 0.0037696325924013088

beta6: -0.0653215017367833

beta7: -0.056982822740641995

MAX_LL: -17680.803552203855

9. Estimate all relevant parameters for NBD regression using MLE. Report your code, the estimated parameters and the maximum value of the log-likelihood (and any other information you believe is relevant). What are the managerial takeaways — which customer characteristics seem to be important?

```
def NLL_NBR(params, a1, a2, a3, a4, a5, a6, a7, a8):
    shape, scale, b1, b2, b3, b4, b5, b6, b7 = params
    ll = 0
    prob = []
    for i in range(len(a1)):
        k = a1[i]
        regr = (b1*a8[i]+b2*a7[i]+b3*a6[i]+b4*a5[i]+b5*a4[i]+b6*a3[i]+b7*a2[i])
        expo = np.exp(float(regr))
        gam = np.log(float(math.gamma(shape+k)))-np.log(float(math.gamma(shape)))-np.log(float(math.factorial(k)))
        term1 = -1*(shape+k)*np.log(float(scale+expo))
        term2 = shape*np.log(float(scale))+k*regr
        ll += gam+term1+term2
#         break;
    return -1*ll
```

```
soln = minimize(
    NLL_NBR,
    method='SLSQP',
    args = (num_purchases, a2, a3, a4, a5, a6, a7, a8),
    x0 = np.array((2,1,0.1,0.2,0.3,0.4,0.5,0.6,0.7)),
    bounds=None,
    tol=1e-10,
#     options={'ftol' : 1e-8},
)
soln
```

```
Out[96]:       fun: 8242.259467588943
               jac: array([-0.01501465, -0.00488281, -0.00402832,  0.00012207, -0.00317383,
                 -0.0032959 , -0.00402832, -0.00231934, -0.00561523])
           message: 'Optimization terminated successfully'
              nfev: 388
               nit: 32
              njev: 32
            status: 0
           success: True
                 x: array([ 0.10220741,  0.1468624 , -0.10213137,  0.00511048,  0.07643193,
                  0.02587902, -0.00401285, -0.06539525, -0.05067842])
```

The estimated parameters and the maximum value of the log-likelihood

shape: 0.10220741345011225

scale: 0.14686239793924827

beta1: -0.1021313694168271

beta2: 0.005110476228908581

beta3: 0.0764319298366121

beta4: 0.025879016021248317

beta5: 0.025879016021248317

beta6: -0.004012851855950636

beta7: -0.06539525070779345

MAX_LL: -8242.259467588943

10. Evaluate all the models developed using the log-likelihood ratio, AIC, and BIC. What are your recommendations on which model to use based on each of these criteria? Are the recommendations consistent? Explain why you are recommending the model you have selected. Are there any significant differences among the results from the models? If so, what exactly are these differences? Discuss what you believe could be causing the differences.

aic = 2k - 2 × CL (k: number of parameters)
bic = kln(n) - 2 × LL(n: number of records)

where *L* is the value of the likelihood, *N* is the number of recorded measurements, and *k* is the number of estimated parameters.

	Poisson Model	NBD Model	Poisson Regression	NBD Regression
No of parameters	**1**	**2**	**8**	**9**
log-likelihood ratio	**-17773.13**	**-8233.00**	**-17680.80**	**-8242.26**
AIC	**35548.26**	**16470**	**35377.6**	**16502.52**
BIC	**35555.41**	**16484.31**	**35434.83**	**16566.9**

A good model is the one that has minimum AIC among all the other models. A lower AIC or BIC value indicates a better fit.

When comparing the Bayesian Information Criteria and the Akaike's Information Criteria, the penalty for additional parameters is more in BIC than AIC.

I recommend the NBD and NBD Regression Model because it has very low value's(AIC and BIC) when compared to Poisson models.

Bayesian Information Criteria is consistent whereas Akaike's Information Criteria is not so.
Akaike's Information Criteria is good for making asymptotically equivalent to cross-validation. On the contrary, the Bayesian Information Criteria is good for consistent estimation.

I think the difference was caused by the log-likelihood ratio value.