PSBC Project 3: Purchasing Data Analysis

Student ID: 9641876

May 11, 2018

Introduction

For this project there is a given set of data from a large online retailer collected over a period of two years. The tasks are all based around analysis of these data.

1 Probability of returning a product in relation to customer ratings

In this section the task is to find a relation between the rating, r given by a customer for a certain item and the probability, $P(r)$ that they will return that item. The logistic function to be used is

$$P(r) = \frac{1}{1 + \exp(-\alpha r - \beta)}.$$

where α and β are parameters to be found for the logistical regression. The only ratings to be used in this calculation are those given by customers who have returned at least one item.

The first step in this task is to extract to a table, `T1`, only the data of the customers who have returned at least one item and where a rating is actually given. From this data it is possible to get a table of just the entries where the product was returned and thus obtain an array of all the customers who have returned at least one item. Then, using the MATLAB function `ismember` makes it simple to obtain an index array of *all* the purchases in `T1` made by these customers. This index gives the information required to construct an array of all the relevant ratings, and a logical array where 1 corresponds to a returned item, and 0 a non-returned item.

The code to achieve all this is as follows.

```
T1 = rmmissing(readtable('purchasing_order.csv')); %
    table of data minus the entries that have no
   rating

return_Y = T1(cell2mat(T1.Return)=='Y',:); % all
   entries in the table where the item was returned
customers_returned = return_Y.Customer_ID; % all
   unique customer ids corresponding to the previous
    table entries
```

```
ind1 = find(ismember(T1.Customer_ID,
   customers_returned)); % index array for all the
   entries in the full table for customers who
   returned at least one item

h = T1.Rating(ind1); % the ratings for all customers
    who returned at least one item

p = cell2mat(T1.Return(ind1))=='Y'; % logical array
   where 1 is an entry with a returned item, 0
   otherwise
```

Finally, using the function `logreg` defined below,

```
function S = logreg(a,h,p)  % a = [a , b]

S = 0;
for k = 1:length(h)
    S = S + (p(k)-1/(1+exp(-a(1)*h(k)-a(2))))^2;
end
```

and minimising it using `fminsearch`,

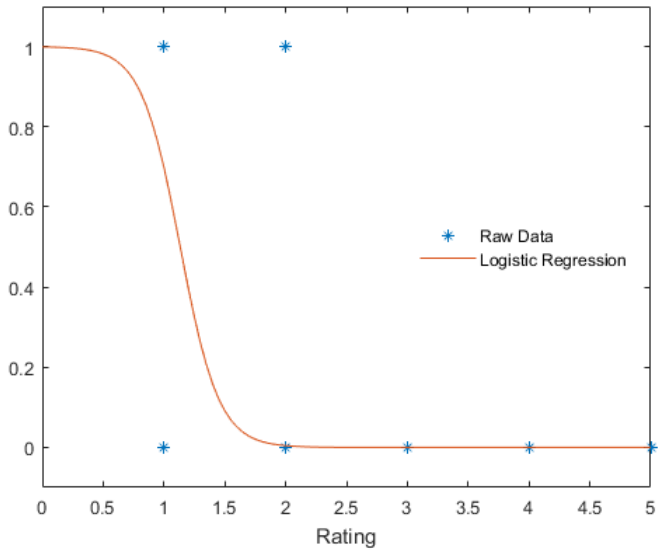
```
lr_par = fminsearch(@(a)logreg(a,h,p),[0,0])
```

gives the values of α and β that are required:

```
lr_par =

  -6.275855101753688   7.113583739393647
```

A plot of this logistic regression shows the results that should be expected; that for higher ratings the probability of returning an item is close to 0 and for lower rating the probability of returning an item gets closer to 1.



Code for the plot:

```
hx = linspace(0,5,101);

plot(h,p,'*',hx,1./(1+exp(-lr_par(1)*hx-lr_par(2))))
    ;
lg = legend('Raw Data','Logistic Regression');
set(lg,'Location','East','box','off');
xlabel('Rating');

axis([0 5 -0.1 1.1]);
```

2 Likelihood of future purchases after making a return

The task here is to calculate the likelihood that after returning one item, a customer will make further purchases. The approach is to, for each customer, find the sum of all the non-returned purchases they made after the initial return and divide it by the sum of *all* the non-returned purchases they have made total. Then taking the mean of these ratios for each customer will give a general likelihood that after returning an item any customer will make further purchases.

All purchase data is required for this task so the first step is to extract all the data to a new table, `T2`. Then using a similar method to task 1, obtain a subtable of the entries for only customers who have returned an item. This time it is important to also get an index array `ind2` for where each of the customers first returned purchase occurs in `T2`. This will be used later in the task. For now, the next step is to find the index of all purchase data in `T2` where a) the customer has returned an item at some point and b) this particular item was not returned. Then using `grpstats` this data can be grouped by customer, and a sum can be calculated of all the non-returned purchases they have made.

Next, using the previously obtained `ind2` it is possible to find, for each customer, the date on which they first returned an item, and hence calculate a sum of all the non-returned purchases they made after this date, and add this sum to an array of all such sums, for each customer. Using `datenum` is essential for comparing dates here.

Finally, sorting these sums by the customer ID corresponding to them and using vector calculations to get the appropriate ratios for each customer, and then finding the mean of all the ratios gives the result required.

What follows is the code to carry all this out.

```
T2 = readtable('purchasing_order.csv'); % table of
    all data

return_Y = T2(cell2mat(T2.Return)=='Y',:); % all
    entries in the table where the item was returned
[customers_returned, ind2] = unique(return_Y.
    Customer_ID,'stable'); % all unique customer ids
    corresponding to the previous table entries, and
    the index array such that return_Y.Customer_ID(
    ind1,:) == customers_returned
```

```
ind3 = find(ismember(T2.Customer_ID,
    customers_returned)); % index array for all the
    entries in the full table for customers who
    returned at least one item
ind4 = intersect(find(cell2mat(T2.Return)=='N'),
    ind3); % same as ind3 but without any entries
    where the item was returned
total_sums = grpstats(T2(ind4,:),'Customer_ID','sum'
    ,'DataVars',{'Product_Value'}); % table of the
    ind4 enties of T2, grouped by customer and
    calculating the sum of all purchases (not
    returned) for each customer


customers_returned_data = return_Y(ind2, :); %
    entries in T2 for the first return each customer
    made

sum_values = zeros(height(customers_returned_data)
    ,1); % preallocation for the array of sums of
    purchase values

for i = 1 : height(customers_returned_data)

    % for each customer this finds the date they
       first returned an item and finds the sum of
       all (non returned) purchases made after that
       point, adds it to an array of all such sums

    dat = datenum(customers_returned_data{i,1});
    cust = customers_returned_data{i,2};
    values = T2((T2.Customer_ID==cust)&((datenum(T2.
       Date)>dat))&(cell2mat(T2.Return)=='N'),:);
    sum_values(i,1) = sum(values{:,4});

end

after_return_sums = sortrows(table(
    customers_returned,sum_values),1); % creates a
    table (sorted by increasing customer ID) of all
    customers and the sum of all their (non-returned)
     purchases, after an initial return

ratios = after_return_sums{:,2} ./ total_sums{:,3};
```

```
    % array of probabilities that each customer will
    make another purchase

likelihood = mean(ratios) % mean of probabilities
    for each customer
```

The output is then:

```
likelihood =

    0.544794259249723
```

What this result tells us, is that 54.48% of customers are likely to make purchases after returning an item.

3 Ranking of customers based on ratings given and amount spent

The third and final task is to create a ranking system by which to determine who the top 100 customers are. This ranking system is based on the ratings they have given, and the average amount they have spent on products in category 'C', such that those who gave higher ratings and spent more will rank higher. The approach used here is to scale down the average amount spent by a customer so it becomes a value in the range 1 to 5, and then take the mean of that with the average rating that a customer gave.

To achieve this, it is required to find two index arrays, one, `ind4`, for only the entries in table `T1` where the product category is 'C' and one, `ind5`, for all the entries in table `T1` where the customer has made at least one purchase in category 'C'.

After this, `grpstats` can be used with these index arrays to find the average product value in category 'C', and the average rating given by the appropriate customers, both grouped by customer ID. This gives all the information required to then create the ranking system as described previously.

The code for this task is as follows.

```
ind4 = find(cell2mat(T1.Product_Category)=='C'); %
   index for all entries where the product is
   category C

C_customers = unique(T1.Customer_ID(ind4,:),'stable'
   ); % all customers who bought a product of
   category C

ind5 = find(ismember(T1.Customer_ID, C_customers));
   % index array of all these customers' purchases
   in the full table

avg_spends = grpstats(T1(ind4,:),'Customer_ID','mean
   ','DataVars',{'Product_Value'}); % average
   product values per order in category C, grouped
   by customer

avg_ratings = grpstats(T1(ind5,:),'Customer_ID','
   mean','DataVars',{'Rating'}); % average ratings
   given, grouped by customer
```

```
scaled_avg_spends = (4*(avg_spends{:,3}./max(
    avg_spends{:,3}))) + 1; % scales the average
    spends down to values between 1 and 5

Ranking = mean([scaled_avg_spends,avg_ratings
    {:,3}],2); % a ranking system based on the mean
    of the scaled avergae spend, and the average
    rating
ranked_customers = sortrows(table(C_customers,
    Ranking), 2, 'descend'); % sorts the customers is
     descending order according to the ranking system
     just defined

recipients = ranked_customers{1:100,1} % returns the
     top 100 customers according to the ranking
```

and the output, restricted to just the first and last few of the hundred is:

```
recipients =

     1035326
     1036872
     1037499
      .
      .
      .

     1017454
     1017250
     1034984
```