Problem Solving By Computer - Project 3

4 Appendix

The data set `purchasing_order.csv` can be found on the following GitHub gist.
`https://gist.github.com/jTanG0506/eb2f7f7f3a98d95f0da420f0a7e438ae`

Listing 1: refund_probability.m

```
AllOrders = readtable('purchasing_order.csv');

% Get all the users who have made at least one refund.
ReturnedOrders = AllOrders(strcmp(AllOrders.Return, 'Y'), :);
UsersWithReturns = unique(ReturnedOrders.Customer_ID);

% Get the orders that are placed by users that have made a refund.
Orders = AllOrders(ismember(AllOrders.Customer_ID, UsersWithReturns), :);
OrdersWithRatings = Orders(Orders.Rating > 0, :);

% Find parameters by minimising the cost function.
orderRating = OrdersWithRatings.Rating;
isReturned = strcmp(OrdersWithRatings.Return, 'Y');
lrParams = fminsearch(@(a) costFunction(a, orderRating, isReturned), [0 0]);
fprintf('[alpha, beta] = [%.5f, %.5f]\n', lrParams);

% Plot the linear regression graph.
hx = linspace(0, 6, 1001);
plot(orderRating, isReturned, 'o', ...
     hx, 1 ./ (1 + exp(-lrParams(1) * hx - lrParams(2))));
lg = legend('Raw Data', 'Logistic Regression');
set(lg, 'Location', 'east');
xlabel('Rating'); xticks([1 : 5]);
ylabel('Order Returned'); yticks([0, 1]); yticklabels({'No', 'Yes'})
axis([0.5, 5.5, -0.2, 1.2]);
```

Listing 2: costFunction.m

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

Listing 3: returning_customers.m

```
AllOrders = readtable('purchasing_order.csv');

% Get all the users who have made at least one refund.
ReturnedOrders = AllOrders(strcmp(AllOrders.Return, 'Y'), :);
UsersWithReturns = unique(ReturnedOrders.Customer_ID);

% Get the orders that are placed by users that have made a refund.
Orders = AllOrders(ismember(AllOrders.Customer_ID, UsersWithReturns), :);

initialState = struct('hasRefunded', false, 'totalBefore', 0, ...
                      'totalAfter', 0);
initialStates = repmat({initialState}, length(UsersWithReturns), 1);
m = containers.Map(UsersWithReturns, initialStates);

% Enumerate over all orders placed by users with at least one refund.
for row = 1 : height(Orders)
  order = Orders(row, :);
  customer = m(order.Customer_ID);

  % If the order was returned, set the customer's hasRefunded flag to true.
  if strcmp(order.Return, 'Y')
    customer.hasRefunded = true;
  else
    % Update the total value based on is customer's hasRefunded flag.
    if customer.hasRefunded
      customer.totalAfter = customer.totalAfter + order.Product_Value;
    else
      customer.totalBefore = customer.totalBefore + order.Product_Value;
    end
  end

  m(order.Customer_ID) = customer;
end

mapValues = m.values
customerData = [mapValues{:}]
totalSpending = [customerData.totalBefore] + [customerData.totalAfter];
percentSpentAfterRefund = [customerData.totalAfter] ./ totalSpending * 100;

% Average of a customers total order value after the first returned order.
averageSpentAfter = mean(percentSpentAfterRefund);
fprintf('The average of a customers total order value after the first
         refund is %.2f%%\n', averageSpentAfter);

histogram(percentSpentAfterRefund, 20);
grid on;
xlabel('Percentage of total value spent after the first refund');
ylabel('Number of customers'); ylim([0 13]);
```

Listing 4: valuable_customers.m

```
Orders = readtable('purchasing_order.csv');

% Mean order value for Category C products, by Customer ID.
OrdersFromC = Orders(strcmp(Orders.Product_Category, 'C') ...
                     & strcmp(Orders.Return, 'N'), :);
CatCValue = groupsummary(OrdersFromC, {'Customer_ID'}, ...
                         'mean', 'Product_Value');

% Mean rating, by Customer ID.
OrdersWithRatings = Orders(Orders.Rating > 0, :);
Ratings = groupsummary(OrdersWithRatings, {'Customer_ID'}, ...
                       'mean', 'Rating');

% Create a table of the form [Customer ID, Mean Value, Mean Rating].
TLeft = table(CatCValue.Customer_ID, CatCValue.mean_Product_Value, ...
              'VariableNames', {'Customer_ID', 'Mean_Value'});
TRight = table(Ratings.Customer_ID, Ratings.mean_Rating, ...
               'VariableNames', {'Customer_ID', 'Mean_Rating'});
T = outerjoin(TLeft, TRight, 'MergeKeys', true);

% Replace any potential NaN values with 0.
T = fillmissing(T, 'constant', 0);

% Standardize attributes and calculate ranking by summing the z-scores.
T.Mean_Value = zscore(T.Mean_Value);
T.Mean_Rating = zscore(T.Mean_Rating);
Ranking = table(T.Mean_Value + T.Mean_Rating, 'VariableNames', {'Ranking'});
T = [T Ranking];

% Sort customers based on ranking and output top 10 customers.
T = sortrows(T, 'Ranking', 'Descend');
disp(T(1:10, {'Customer_ID', 'Ranking'}));
```