Marketing and Customer Analytics

Individual Assignment
Paid Search Bid Optimization and Display Advertising

Question 1 Paid Search Bid Optimization

A travel services firm has a paid search campaign. Among the many keywords in its campaign, we have data on four keywords, denoted by kw8322228, kw8322392, kw8322393 and kw8322445. These are generic, non-branded keywords, where the prospect's query does not indicate that he/she is leaning toward a specific brand. For each keyword, the firm tried several bid values and recorded the corresponding number of clicks that it received.

The files are named **clicksdata.kw8322228.csv**, **clicksdata.kw8322392.csv**, **clicksdata.kw8322393.csv**, **clicksdata.kw8322445.csv** respectively.

Part A: Estimate the alpha and beta parameters for each of these four keywords for this firm. Hand-in: The eight numbers. No additional writeup required. Hint on checking your answers: For kw8322228, alpha should be between 70 and 76, beta should between 0.03 and 0.06, with a residual-sum-of-squares of about 230.

To estimate the alpha and beta for a keyword you need to run nonlinear regression `n.clicks` as a function of `bid.value` and using the appropriate function form. In R, nonlinear regression using nls() and the basic call is something like what is given below.

`nls_output <- nls(n.clicks ~ alpha*(1-exp(-beta*bid.value)), start = list(alpha={`starting value of alpha`}, beta={`starting value of beta`}), data={`name of data frame`})`

In Python you can use the scipy.optimize.curve_fit function.

As you may recall from previous courses, a major problem with nonlinear regression models (including neural networks) is that they are often subject to sensitivity to the starting values meaning that different starting values lead to different estimates. One way to choose the starting values is by domain experience. For this, we suggest you use initial values of 150 and 0.5 respectively for alpha and beta. For other datasets, for alpha, use as starting value the number of clicks at the highest bid. For the beta, you can guess it as 0.69/b_half where b_half is the approximate bid value at which the number of clicks is half of the initial value of alpha (i.e. half of the number of clicks at the highest bid). Another heuristic is to set beta to be the reciprocal of the average bid (i.e. average of the x variables). Of course, these are just heuristics and may not always work. For something that is more reliable, the alternative is to pick multiple starting values randomly from a broad range. The starting value will be a two-element list or vector (one for alpha, one for beta). For each such starting value vector out of many, you will need to run nonlinear regression and then identify which starting value vector leads to the final nonlinear regression output with the lowest residual sum of squares, and then take the nonlinear regression output from that starting value vector to give the final estimates of alpha.hat and beta.hat. You will do this for each keyword.

When one tries different starting-value-vectors, one sometimes finds that some starting-value-vectors lead to errors reported by R or Python. Make sure you set up the bounds so the

lower limit and each of alpha and beta is zero. In particular, if beta is allowed to go negative, then python can generate floating point errors and curve_fit will fail. R is somewhat better behaved but this problem can occur in R as well. When that happens, you should take that starting value to be sub-optimal and set the corresponding residual-sum-of-squares to be some very large number like Infinity. This ensures that in the next stage when one picks the endpoint with the lowest residual-sum-of-squares across all starting-value-vectors, such error-generating starting-value-vectors will never be considered.

Part B: Assume that the LTV dollar value and the conversion rate values for each of the keywords for this firm are as shown in the table below. These numbers are available in an Excel spreadsheet **hw-kw-ltv-conv.rate-data.xlsx**.

keyword	ltv	Conv.rate
kw8322228	354	0.3
kw8322392	181	0.32
kw8322393	283	0.3
kw8322445	107	0.3

Assume that you have no budget constraint. Using the alpha, beta parameters from Part A and the LTV and conversion rate values given above, estimate the optimal bids for each of the four keywords. Hand-in: the optimal bid value, the corresponding profit and the corresponding total expenditure for each of the four keywords. No additional writeup required.

Hint on Checking your answer: For kw8322228, at the optimal solution, the expenditure is between $1800 and $1900.

Obviously, this involves running an optimization. The easiest way to do this for this HW is to use SOLVER in Excel using steps analogous to what is shown on this web page for the simpler problem where there are only two keywords. In actual business operations with thousands of keywords, SOLVER is impractical and for that you will need to use R or Python. Therefore, for this HW you need to use R or Python, **NOT Excel or Excel SOLVER**. For R, we suggest using the optim() function. If you want to use Python, then we suggest you use the scipy.optimize.minimize function. Make sure you set up the bounds so the lower limit is zero. If the bid is allowed to go negative, then you can get degenerate solutions.

Part C: Assume now that you have a budget constraint of $3000 across these four keywords. Compute the optimal bid amounts and the corresponding expenditures for the keywords. Note this optimization in its most obvious form involves nonlinear functions and nonlinear constraints. Decide on the initial value vector x0 on your own. Hand-in: the optimal bid value, the corresponding profit and the corresponding total expenditure for each of the four keywords.

Hint on Checking your answer: At the optimal solution, the total profit is in between $17900 and $18100.

Here too, we have an optimization but this time it is a constrained optimization. Again, you need use R or Python for this HW and **NOT Excel**.

For R: if one has just linear constraints, then the recommended function is ConstrOptim(). However, for this problem the constraint is nonlinear. constrOptim() cannot handle nonlinear constraints and therefore cannot be used for this HW. To handle nonlinear constraints, you can use the solnp function from the Rsolnp library.

For Python: The scipy.optimize.minimize function is what we recommend. It handles both linear and nonlinear constraints. You first have to create a nonlinear constraint object using the `optimize.NonlinearConstraint` method. You then pass this to scipy.optimize.minimize via the `constraints` argument. One way to do create the constraint object is as shown below, where `total_expenditure()` is a function that takes as input the vector of four bids and computes the total expenditure.

```
budget = 3000
budget_constraint_object =
optimize.NonlinearConstraint(total_expenditure, 0, budget)
```

We suggest you set `method='trust-constr'`. Setting `method='SLSQP'` produced an error for me even though according to the documentation it should work. Unfortunately Python's optimization routines are buggy and are often revised, so make sure you have the latest version of scipy. Furthermore, make sure you set up the bounds, so the lower limit is zero. If the bid is allowed to go negative, then you can get degenerate solutions.

Question 2 Display Advertising Assessment

Assume that you have run five display ad campaigns, each with 1000 exposures. The data are given here in the same format as in the in-class example: the raw clicks data are in **clicks.dataset.2.xlsx** and the post-click transaction profit volumes are in **volumes.dataset.2.xlsx**.

Like in the in-class example, we are discounting clicks that do not lead to conversions. So, a "click" is equivalent to a conversion and the "post-click transaction profit volume" is equivalent to "post-conversion transaction profit volume".

Like in class, we want to make an assessment, for each campaign, of its true click-through-rate (abbreviated as "c") and its true average post-click transaction profit volume (abbreviated as "v"). You can think of "v" as the average profit from a click or conversion. We want to make an assessment, for each campaign, also of its true expected volume per exposure or impression (abbreviated as "EVI"). The true EVI of a campaign is its "c" multiplied by its "v". For each campaign, compute the following:

1. The bayesian posterior probability that the campaign's true click-through-rate is the highest across all campaigns. Assume the prior is the uniform distribution. To generate draws from the beta distribution, in R use the rbeta() function and in Python use scipy.stats.beta.rvs(a=alpha value, b=beta value, size=number of draws needed). Here alpha value and beta value are, respectively, what we referred to in the sessions as the success parameter and the failure parameter.
2. The bayesian posterior probability that the campaign's true average post-click volume per click is the highest across all campaigns. To generate draws from the t distribution in R use the rt() function. In Python, first generate draws from the standardized t distribution using scipy.stats.t.rvs(df= the appropriate degrees of freedom,

size=number of draws needed). Once you have the draws in either R or Python you have to multiply by the standard error and then add the sample mean as shown in the example Excel computation to obtain the draws from the target posterior probability density.

3. The bayesian posterior probability that the campaign's true expected volume per exposure (impression) is the highest.

In the probability computations, use a simulation with **100000 draws** (this is higher than in the in-class Excel example which has 5000 draws).

Because there is randomization, each refresh of your computations will produce different answers. A solution would be for me to instruct all of you to fix the random seed to some value. However, I have chosen not to do this, for two reasons. First, when I did the HW, I found the Monte Carlo error variation was quite small, with the differences only in the third decimal place. Second, fixing the random seed will not lead everyone to have the same answers. This is because there are different ways to write the loops or vectorization.

This HW is closely similar to the in-class example which has 3 campaigns whose response data are given in the following two files: the raw clicks data are in **clicks.dataset.1.xlsx**, and the post-click transaction profit volumes are in **volumes.dataset.1.xlsx**. I will now describe the analysis I did using Excel. For the HW you need to replicate similar work in R or Python. **You cannot use Excel for your HW submission**.

Details on the operations involved in Question 2:

The first task relates to computing the bayesian posterior probability that a certain campaign's true CTR is the highest. The basic result we use here is that with the uniform prior, the posterior distribution of the true CTR for a campaign has the beta distribution with the success parameter = "number of clicks" +1, and the failure parameter = "number of exposures that did not result in clicks" + 1. (The two 1s are there because we take the prior distribution to be the uniform distribution and this distribution is beta distribution with s0 and f0 both equal to 1.) We draw from the CTR distribution for each of the campaigns, and then compute the fraction of times (across all draws) that a particular campaign's CTR draw is the highest across all campaigns. I followed the steps given below and at the end of the analysis I was left with a spreadsheet that looks like **clicks.dataset.1.analysis.xlsx**.

- I created a copy of "clicks.dataset.1.xlsx" and called it "clicks.dataset.1.analysis.xlsx". In the "clicks.dataset.1.analysis.xlsx" file I added a worksheet called "posterior analysis". All the steps that follow are in the "posterior analysis" worksheet.
- I copied the raw data from the "clicks.dataset.1" worksheet on to cells A1:D3.
- Below that in cells A5:D7 I created a block of numbers with the same values except that the last row is not the number of exposures but is the number of "non-clicks" or failures, which is simply the number of exposures minus the number of clicks. I labeled this row as "f" for failures. The row above this, the one containing the numbers of clicks I labeled as "s" for successes.
- Below that in cells A8:D9, I created the block of numbers with the posterior beta distribution parameters, which are just the "s" and "f" numbers with 1 added to them. These give the beta.s and beta.f parameters and the rows 8 and 9 are labeled accordingly under column A.

- In row 12 in cells A12:H12, I wrote in the field names "draw.number", "draws1", "draws2", "draws3", "max", "ismax1", "ismax2", "ismax3".
- In column A from rows 13 through 5012, I entered the sequence of integers 1, 2, ..., 5000 to correspond to 5000 draws (for the HW you need to do 100000 draws). There are many ways to create this number sequence in Excel. The easiest method for me is by using the Editing->Fill->Series menu option in Excel, but I know others are more comfortable using the drag+fill mouse operation.
- In cell B13, I entered the key formula to generate the draw from the posterior distribution of the CTR of the first campaign. This is done using the Excel expression BETAINV(RAND(), beta.s, beta.f), which for Campaign 1 would be BETAINV(RAND(), B$8, B$9) because B$8 and B$9 contain the beta.s and beta.f values for Campaign 1.
- I copied this formula in B13 over to cell C13 and cell D13.
- In cell E13, I computed the maximum of the draws in row 13, using the Excel expression MAX(B13:D13).
- In cell F13, I computed a simple one-zero dummy variable which represents whether or not a campaign has the highest value. This is done by using the expression IF(B13=$E13, 1, 0).
- I copied the formula in cells F13 over to cells G13 and H13.
- I then used the fill operator to copy the formula in cell B13 to all the cells below it, all the way down to B5012. Similarly, I applied the formulas in each of C13, D13, E13, F13, G13 and H13 to all the cells below until C5012, D5012, E5012, F5012, G,5012 and H5012 respectively.
- I then computed in cell F11 the average of the cells F13 through F5012. This represents the probability or fraction of times that Campaign 1's CTR is the highest. This is computed using the Excel expression AVERAGE(F13:F5012). I then copied over the formula from cell F11 to G11 and H11. These three cells are the key outputs we need. They represent the probability of each of the campaigns having the highest CTR. I ended by adding the text labels "ave.ismax1", "ave.ismax2", "ave.ismax3", in cells F10 to H10, and highlighting cells F10:H11 in yellow.

You can see all the above operations in the **clicks.dataset.1.analysis.xlsx** file.

The second task relates to computing the bayesian posterior probability that a certain campaign's average profit volume is the highest. Let us denote the true average profit volume as "m". This time the basic result we use here is that the posterior distribution of "m" has a t distribution with mean = sample mean, scale= the standard error of the sample mean, and degrees of freedom = the sample size + 1. We draw from the distribution of "m" for each of the campaigns, and then compute the fraction of times (across all draws) that a particular campaign's "m" is the highest across all campaigns. I followed the steps given below and at the end of the analysis I was left with a spreadsheet that looks like this one: **volumes.dataset.1.analysis.xlsx**.

- I created a copy of "volumes.dataset.1.xlsx" and called it "volumes.dataset.1.analysis.xlsx". In the "volumes.dataset.1.analysis.xlsx" file I added a worksheet called "posterior analysis".

- I need for each campaign the sample mean, standard deviation and number of observations. This is straightforward to do in R or Python of course, but since I am working in Excel I used XLSTAT which produced the groupwise means, std deviation and sample size in a worksheet named "Desc". The key ouput I looked for here is the block of numbers under "Descriptive statistics (Quantitative Data)" at row 7. I selected cells B19:E18 and copied them over to cells A1:D10 in the "posterior analysis" worksheet that I previously created. The rest of the work is now in the"posterior analysis" worksheet.
- I need for each campaign the sample mean, the standard error of the mean, and the bayes posterior degrees of freedom. I started with Campaign 1. I put the sample mean in cell B11, by simply referencing cell B8. The standard error of the mean is computed by taking the standard deviation and dividing by the square root of the sample size. This is computed in cell B12 using the expression B10/SQRT(B2). Lastly,the bayes posterior degrees of freedom is computed as the sample size + 1. This is computed in cell B13 using the expression B2+1. To repeat this for Campaigns 2 and 3, I simply copied cells B11:B13 to C11:C13 and D11:D13 respectively. In cells A11:A13, I entered the text labels "mean", "std.err", "bayes.posterior.df" to indicate what these rows stand for.
- In row 17 in cells A17:H17 I wrote in the field names "draw.number", "draws1", "draws2", "draws3", "max", "ismax1", "ismax2", "ismax3".
- In column A from rows 18 through 5017, I entered the sequence of integers 1, 2, ..., 5000 as before (again, for the HW you need to do 100000 draws).
- In cell B18, I entered the key formula to generate the draw from the posterior distribution of "m" of the first campaign. This is done using the Excel expression T.INV(RAND(), degrees of freedom)*standard error of the mean + sample mean, which for Campaign 1 would be T.INV(RAND(), B$13)*B$12 + B$11 because B$13 contains the degrees of freedom, B$12 contains the standard error of the mean, B$11 contains the sample mean.
- The next few steps are identical (except for cell locations) to what I did for Task 1 above. I copied the formula in B18 over to cell C18 and cell D13.
- In cell E18, I computed the maximum of the draws in row 18, using the Excel expression MAX(B18:D18).
- In cell F18, I computed a simple one-zero dummy variable which represents whether or not a campaign has the highest value. This is done by using the expression IF(B18=$E18, 1, 0).
- I copied the formula in cells F18 over to cells G18 and H18.
- I then used the fill operator to copy the formula in cell B18 to all the cells below it, all the way down to B5017. Similarly, I applied the formulas in each of C18, D18, E18, F18, G18 and H18 to all the cells below until C5017, D5017, E5017, F5017, G,5017 and H5017 respectively.
- I then computed in cell F16 the average of the cells F18 through F5017. This represents the probability or fraction of times that Campaign 1's "m" is the highest. This is computed using the Excel expression AVERAGE(F18:F5017). I then copied over the formula from cell F16 to G16 and H16. These three cells are the key outputs we need. They represent the probability of each of the campaigns having the highest

"m". I ended by adding the text labels "ave.ismax1", "ave.ismax2", "ave.ismax3", in cells F16:H16, and highlighting cells F15:H16 in yellow.

You can see all the above operations in the **volumes.dataset.1.analysis.xlsx** file.

The third task relates to computing the bayesian posterior probability that a certain campaign's CTR*m is the highest. The CTR*m is a very important quantity in display campaign assessment because it represents the monetization per impression or exposure of a display ad. For any particular campaign, to draw from the posterior distribution of CTR*m we simply take a CTR draw (which we computed under the first task above) and a "m" draw (which we computed under the second task above) and multiply the two. We repeat this for each campaign. We then compute the fraction of times (across all draws) that a particular campaign's "m*CTR" is the highest across all campaigns. I followed the steps given below and at the end of the analysis I was left with a spreadsheet that looks like this one: **joint.dataset1.analysis.xlsx**.

- I created a new, blank Excel file and named it "joint.dataset1.analysis.xlsx". In cells A3 to M3 I entered the text labels "ctrdraws1", "ctrdraws2", "ctrdraws3", "mudraws1", "mudraws2", "mudraws3", "product.1", "product.2", "product.3", "max", "ismax1", "ismax2", "ismax3".
- I copied cells B13:D5012 from the "posterior analysis" worksheet of **clicks.dataset.1.analysis.xlsx** and pasted over to cells A4:C5003 of "joint.dataset1.analysis.xlsx". Very Important: When you do the Paste operation, make sure you do Paste Special-> Values because the referenced cells are not available in this new spreadsheet.
- Similarly, I copied cells B18:D5017 from the "posterior analysis" worksheet of **volumes.dataset.1.analysis.xlsx** and pasted over to cells D4:F5003 of "joint.dataset1.analysis.xlsx". Again, make sure you do Paste Special-> Values.
- Now we compute the "m*CTR" draws by multiplying the CTR draw with the "m" draw. For Campaign 1, I did this in cell G4 by entering the formula A4*D4. I copied this over to cell H4 and I4 for campaigns 2 and 3 respectively.
- The next few steps are similar to those we followed for the first task and the second task. In cell J4, I computed the maximum of the product draws in row 4, using the Excel expression MAX(G4:I4).
- In cell K4, I computed a simple one-zero dummy variable which represents whether or not a campaign has the highest value. This is done by using the expression IF(G4=$J4,1,0)
- I copied the formula in cells K4 over to cells L4 and M4.
- I then used the fill operator to apply the formula in cell G4 to all the cells below it, all the way down to G5003. Similarly, I applied the formulas in each of H4, I4, J4, K4, L4, M4 to all the cells below until H5003, I5003, J5003, K5003, L5003, M5003 respectively.
- I then computed in cell K2 the average of the cells K4 through F5003. This represents the probability or fraction of times that Campaign 1's CTR*m is the highest. This is computed using the Excel expression AVERAGE(K4:K5003). I then copied over the formula from cell K2 to L2 and M2. These three cells are the key outputs we need. They represent the probability of each of the campaigns having the highest CTR*m. I

ended by adding the text labels "ave.ismax1", "ave.ismax2", "ave.ismax3", in cells K1 to M1, and highlighting cells K1:M2 in yellow.

You can see all the above operations in the **joint.dataset1.analysis.xlsx** file.

How to Submit: Submit the following items online via LMS. The page-lengths given below are suggestive only. Feel free to exceed the suggested page lengths if you see the need.

1. A 3–6-page PDF report containing the deliverables requested for all parts. Do not copy the questions in the report.
2. Your R (.rmd or .html) or Python (.ipynb) codes for all parts.
3. Any additional Excel workbooks or csv files created as part of your solution.