TZ Gaming: Optimal Targeting of Mobile Ads

Prof. Hema Yoganarasimhan, Foster School of Business, University of Washington
Prof. Vincent Nijs, Rady School of Management, UCSD

Winter 2022

As a developer of games for mobile devices TZ gaming has achieved strong growth of its customer base. A prominent source of new customers has come from ads displayed through the Vneta ad-network. A mobile-ad network is a technology platform that serves as a broker between (1) app developers (or publishers) looking to sell ad space and (2) a group of advertisers.

App developers sell "impressions", i.e., a space where an ad can be shown, through the Vneta network to companies such as TZ gaming looking to advertise to app users. Vneta acts as a broker for 50-60 millions impressions/ads per day.

TZ gaming uses ads to appeal to prospective customers for their games. They generally use short (15 sec) video ads that help to emphasize the dynamic nature of the games. In the past, TZ has been able to, approximately, break-even on ad-spend with Vneta when calculating the benefits that can be directly attributed to ad click-through. Many senior executives at TZ believe that there are additional, longer-term, benefits from these ads such as brand awareness, etc. that are harder to quantify.

Currently, TZ has access to very limited data from Vneta. Matt Huateng, the CEO of TZ gaming, is intrigued by the potential for data science to enhance the efficiency of targeted advertising on mobile devices. Specifically, two options are under consideration: (1) Buy access to additional data from Vneta and use TZ's analytics team to build targeting models or (2) Subscribe to Vneta's analytics consultancy service, which provides impression-level click-through rate predictions based on Vneta's proprietary data and algorithms.

Vneta has shared behavioral information linked to 115,488 recent impressions used to show TZ ads and has also provided a set or predictions based on their own (proprietary) algorithm. Matt is not convinced that the consulting services offered by Vneta will be worth the money for future ad campaigns and has asked you to do some initial analyses on the provided data and compare the generated predictions to Vneta's recommendations. The following options will be evaluated to determine the best path forward.

Options:

1. Spam all propects
2. Continue with the current targeting approach
3. Use predictions from a logistic regression model for ad targeting
4. Use predictions generated by Vneta for ad targeting

The assumptions used for the analysis are as follows:

- Targeting of impressions to consumers covered by the Vneta ad-network to date has been (approximately) random
- Cost per 1,000 video impressions (CPM) is $10
- Conversion to sign-up as a TZ game player after clicking on an ad is 5%
- The expected CLV of customers that sign-up with TZ after clicking on an ad is approximately $25
- The price charged for the data by Vneta is $50K
- The price charged for the data science consulting services by Vneta is $150K

Approach:

- Use the 87,535 rows in the data with "training == 'train' " to estimate different models. Then generate predictions for all 115,488 rows in the dataset
- Options 1-4 should be evaluated *only* on the predictions generated for the 27,953 rows in the data with "training == 'test' ". These are the observations that were *not* used to estimate your model
- Extrapolate the cost and benefits for options 1-4 above for an upcoming advertising campaign where TZ will commit to purchase 20-million impressions from Vneta

Although TZ gaming has used RFM for targeting existing customers this approach is not appropriate for prospective customers. Instead, you have decided to use logistic regression. This is a powerful and widely used tool to model consumer response. It is similar to linear regression but the key difference is that the response variable is binary (e.g., click or no-click) rather than continuous. For each impression, the logistic regression model will predict the probability of click-through, which can be used for ad targeting. Like linear regression, you can include both continuous and categorical predictors in your model as explanatory variables.

Matt is eager to assess the value of logistic regression as a method to predict ad click-through and target prospects and has asked you to complete the following analyses.

Part I: Logistic Regression (10 points)

> Note: For the following questions, use only the "training" sample of impressions (i.e., 87,535 rows where "training == 'train' "). `click_yes` is defined in the example python code shown below

a. Estimate a logistic regression model using `click_yes` as the response variable and the following as explanatory variables (1 point):

`time_fct, app, mobile_os, impua, clua, ctrua`

Use the formula api for `statsmodels` as shown below. The basic structure of your code to estimate a logistic regression in python should be as follows:

```
import statsmodels.formula.api as smf
tz_gaming["click_yes"] = rsm.ifelse(tz_gaming.click == "yes", 1, 0)
lr = smf.glm(
    formula="click_yes ~ time_fct + app + mobile_os + impua + clua + ctrua",
    family=Binomial(link=logit()),
    data=
).fit()
lr.summary()
```

b. Summarize and interpret the logistic regression results. Which of these explanatory variables are statistically significant? Which variables seem to be most "important"? Make sure your model evaluation includes (1) an interpretation of the odds-ratios estimated for the explanatory variables `mobile_os`, `impua`, `clua`, and `ctrua` and (2) an evaluation of the model as a whole (4 points).

Use functions from the `pyrsm` package to facilitate your analysis (see e.g., https://github.com/vnijs/pyrsm/blob/master/pyrsm/logit.py). Tips:

- Use the `or_plot` function from the `pyrsm` package to visualize the Odds-ratios
- Use the `or_ci` function from the `pyrsm` package to get estimates of the odd-ratios

```
import pyrsm as rsm
rsm.or_ci(lr)
```

- Calculate the (overall) model-fit statistics using the code below. Provide an appropriate interpretation of the returned value of the pseudo R-squared value and the Chi-square statistic.

```
rsm.model_fit(lr)
```

Hint: The Chi-square test for a logistic regression model is equivalent to an F-test for a linear regression model

c. Predict the probability of a click (1 point)

The estimated logistic regression model can predict the probability of `click_yes == 1`. Create a new variable `pred_logit` with the predicted click-through probabilities linked to each impression. Make sure to generate predictions for all rows in the training and test data.

```
tz_gaming["pred_logit"] =
```

d. Re-estimate the logistic regression after standardizing the numeric explanatory variables (see sample code below). What is the interpretation of the standardized odds-ratios for the explanatory variables? Which variables seem to be most "important" (3 points)?

Tip: Use `or_plot` from the `pyrsm` packages to visualize the (standardized) odds-ratios.

```
# select variables to standardize
to_std = tz_gaming.loc[:, "impup":"ctrpat"].columns

# scale numeric variables by (x - mean(x)) / sd(x)
tz_std = tz_gaming.copy()
tz_std[to_std] = rsm.scale_df(tz_gaming[to_std], sf=1, train=tz_gaming.training == "train")
```

e. Estimate a logistic regression with `click_yes` as the response variable and `rnd` as the **only** explanatory variable. As before, the model should be estimated on the training sample (i.e., "training == 'train' "). Create a new variable `pred_rnd` with the predicted click-through probabilities (1 point).

Part II: Understanding Multicollinearity (10 points)

a. Estimate a logistic regression model with `click_yes` as the response variable and `imppat`, `clpat`, and `ctrpat` as the only explanatory variable. Make sure to "standardize" the explanatory variables before estimation (see sample code below). What is the interpretation of the standardized odds-ratios for the explanatory variables? Tip: Use `or_plot` from the `pyrsm` packages to visualize (1) the odds-ratios and (2) the standardized odds-ratios (2 points).

```
# select variables to standardize
to_std = tz_gaming.loc[:, "impup":"ctrpat"].columns

# scale numeric variables by (x - mean(x)) / sd(x)
tz_std = tz_gaming.copy()
tz_std[to_std] = rsm.scale_df(tz_gaming[to_std], sf=1, train=tz_gaming.training == "train")

# alternative approach
# scaler = preprocessing.StandardScaler()
# sf = scaler.fit(tz_gaming.query("training == 'train'")[to_std])
# tz_std = tz_gaming.copy()
# tz_std[to_std] = sf.transform(tz_std[to_std])

lr_mc1 = smf.glm()
```

b. Some of the variables in the dataset are highly correlated with each other. In particular, `imppat` and `clpat` have a positive correlation of 0.97. Discuss the implications of this (very) high level of collinearity and also different approaches to deal with it. What are the implications for the model and the interpretation of the estimated (standardized) coefficients? As part of your answer, discuss the change in the estimated (standardized) odd-ratio for `imppat` when you remove `clpat` from the model (4 points).

Note: To calculate VIF statistics you can use the `vif` function from the `pyrsm` packages. Note that you must use the `lr_mod` model object created above and not the fitted model `lr`

```
rsm.vif(lr_mc1)
```

c. Estimate another logistic regression model with `click_yes` as the response variable and `time_fct`, `app`, `imppat`, `clpat`, and `ctrpat` as the explanatory variable. Why are the odds ratios for `imppat`, `clpat`, and `ctrpat` different in the two models? Please be specific and investigate beyond simply stating the statistical problem (4 points).

Note: You may find the code below useful. The wald test evaluates is a (set of) coefficients are equal to 0

```
lr_mc3 = smf.glm()

lr_mc3.wald_test_terms().table.round(3)
```

Part III: Decile Analysis of Logistic Regression Results (5 points)

Note: For the following questions, use only the "test" sample of impressions (i.e., 27,953 rows where "training == 'test' ")

a. Assign each impression to a decile based on the predicted probability of click through (`pred_logit`) based on the model estimated in Part I Create a new variable `dec_logit` that captures this information. Note: The first decile should have the highest average click-through rate. If not, make sure to "reverse" the decile numbers (i.e., 10 becomes 1, 9 becomes 2, etc.). Use the `xtile` function from the `pyrsm` package to create the deciles (2 points)

b. Report the number of impressions, the number of clicks, and the click-through rate for the TZ ad per decile and save this information to a dataframe. Use the name `df_dec_logit` for the new data frame (2 points)

c. Create a bar chart of click-through rates per decile (i.e., use `dec_logit` as the x-variable and `click_yes` as the y-variable). Note that the "click through rate" is not the same as the "predicted probability of click." The click-through rate captures the proportion of impressions in a given group (e.g., in a decile) that actually resulted in a click (1 point)

Part IV: Lift, Gains, and Profit (15 points)

Use the `df_dec_logit` DataFrame you created in Part III for the following calculations.

a. Write python code to generate a table with the cumulative proportion of impressions and the cumulative lift for each decile (3 points)
b. Use `seaborn`, `matplotlib`, or `pandas` to create a chart showing the cumulative lift per decile. Put cumulative lift on the Y-axis and cumulative proportion of impressions on the X-axis (2 points)
c. Write python code to generate a table with the cumulative proportion of impressions and the cumulative gains for each decile (3 points)
d. Use `seaborn`, `matplotlib`, or `pandas` to create a chart showing the cumulative gains per decile along with a (diagonal) reference line to represent the "no model" scenario. Put cumulative gains on the Y-axis and cumulative proportion of impressions on the X-axis (2 points)
e. Write python code to generate a table with the cumulative proportion of impressions and profits for each decile (3 points)
f. Use `seaborn`, `matplotlib`, or `pandas` to create a chart showing the profits per decile. Put profit on the Y-axis and cumulative proportion of impressions on the X-axis (2 points)

Note: Do NOT use any specialized packages to construct the lift, gains, and profit tables and charts. Write the python code from scratch

Part V: Confusion matrix (10 points)

a. Create a "confusion matrix" based on the predictions from the logistic regression model you estimated in Part I (i.e., `pred_logit`). Again, use **only** data from the test set here (i.e., "training == 'test' "). Use the financial assumptions mentioned above, and repeated in section VI below, to determine an appropriate cut-off (i.e., break-even). Calculate "accuracy" based on the confusion matrix you created (see http://lab.rady.ucsd.edu/sawtooth/RBusinessAnalytics/logit_models.html for an example using R) (2 points)

Note: Do NOT use any specialized packages to construct the confusion matrix. Code the matrix from scratch.

b. Calculate a confusion matrix based on `pred_rnd` created in Part I and calculate "accuracy" based on the confusion matrix you created (2 points)

c. Discuss the similarities and differences between the two confusion matrices. Which prediction (model) is best, based on the confusion matrix? Provide support for your conclusions(3 points)

d. Recalculate the confusion matrices from V.a and V.b using 0.5 as the cutoff. Based on these new matrices, again discuss the similarities and differences. Which model is best based on these new confusion matrices? Provide support for your conclusions (3 points)

Part VI: Model comparison (12 points)

Use the following cost information to assess the profitability each of these models for targeting purposes during the upcoming advertising campaign where TZ will purchase 20-million impressions from Vneta:

- Cost per 1,000 video impressions (CPM) is $10
- Conversion to sign-up as a TZ game player after clicking on an ad is 5%
- The expected CLV of customers that sign-up with TZ after clicking on an ad is approximately $25
- The total cost of the data from Vneta is $50K
- The total cost charged for the data science consulting services by Vneta is $150K

Use `pred_logit`, `pred_rnd`, and the predictions from Vneta based on their proprietary model `pred_vneta`) to compare model performance.

Note: The currently available data (+ the `pred_vneta` prediction) are free as part of the partnership between Vneta and TZ-gaming.

a. Create a new variable `target_logit` that is `True` if the predicted click-through (`pred_logit`) probability is greater than the break-even response rate and `False` otherwise (1 point)

b. Create a new variable `target_rnd` that is `True` if the predicted click-through (`pred_rnd`) probability is greater than the break-even response rate and `False` otherwise (1 point)

c. Create a new variable `target_vneta` that is `True` if the predicted click-through (`pred_vneta`) probability is greater than the break-even response rate and `False` otherwise (1 point)

d. Based only on the test set (i.e, `training == 'test'`), calculate the expected profit (in dollars) and the expected return on marketing expenditures (ROME) if TZ (1) "spams" everyone in the test set, (2) continues to target using their current approach (`pred_rnd`), (3) purchases the data from Vneta and uses the logistic regression from I (`pred_logit`) for targeting, or (4) used Vneta's data science consulting services (`pred_vneta`) (3 points)

Note: For efficiency, you can adapt the `perf_calc_actual` function you created for the Tuango case to do the relevant performance calculations for the different models.

e. Based on the results from VI.d discuss which of these 4 approaches you would recommend and why (2 points)

f. Calculate the profit and ROME implications for each of the 4 options mentioned in VI.d if TZ purchases 20-million impression for the upcoming ad campaign (2 points)

Note: For efficiency, you can adapt the `perf_calc` function you created for the Tuango case to do the relevant performance calculations for the different models.

g. Based on the results from VI.f, discuss which of the 4 approaches you would recommend to put into production and why. Is your recommendation different from VI.e? Why (not)? (2 points)

Data description

Data on 115,488 impressions is contained in the tz_gaming.pkl file in the data/ directory of the GitLab repo that will be forked to your account. Each row in the dataset represents an impression that showed a TZ ad. All explanatory variables are created by Vneta based on one month tracking history of users, apps, and ads. The available variables are described below.

- *training* – Dummy variable that splits the dataset into a training ("train") and a test ("test") set
- *inum* – Impression number
- *click* – Click indicator for the TZ ad served in the impression. Equals "yes" if the ad was clicked and "no" otherwise
- *time* – The hour of the day in which the impression occurred (1-24). For example, "2" indicates the impression occurred between 1 am and 2 am
- *time_fct* – Same as *time* but coded as categorical
- *app* – The app in which the impression was shown. Ranges from 1 to 49
- *mobile_os* – Customer's mobile OS
- *id* – Anonymized user ID
- *impup* – Number of past impressions the user has seen in the app
- *clup* – Number of past impressions the user has clicked on in the app
- *ctrup* – Past CTR (Click-Through Rate) (x 100) for the user in the app
- *impua* – Number of past impressions of the TZ ad that the user has seen across all apps
- *clua* – Number of past impressions of the TZ ad that the user has clicked on across all apps
- *ctrua* – Past CTR (x 100) of the TZ ad by the user across all apps
- *imput* – Number of past impressions the user has seen within in the hour
- *clut* – Number of past impressions the user has clicked on in the hour
- *ctrut* – Past CTR (x 100) of the user in the hour
- *imppat* – Number of past impressions that showed the TZ ad in the app in the hour
- *clpat* – Number of past clicks the TZ ad has received in the app in the hour
- *ctrpat* – Past CTR (x 100) of the TZ ad in the app in the hour
- *rnd* – Simulated data from a normal distribution with mean 0 and a standard deviation of 1
- *pred_vneta* – Predicted probability of click per impressions generated by Vneta's proprietary machine learning algorithm

The last three letters of a feature name indicate the sources of variation:

- u — denotes user
- t — denotes time
- p — denotes app
- a — denotes ad

Note that there is a clear relationship between the impressions, clicks, and ctr variables within a strata. Specifically: $ctrup = \frac{clup}{impup}$, $ctru = \frac{clu}{impu}$, $ctrut = \frac{clut}{imput}$, and $ctrpat = \frac{clpat}{impat}$.

Professor Hema Yoganarasimhan (Foster School of Business, University of Washington) and Professor Vincent Nijs, Rady School of Management, UCSD prepared this case to provide material for class discussion rather than to illustrate either effective or ineffective handling of a business situation. Names and data have been disguised to assure confidentiality.