Udacity Machine Learning Engineer Nanodegree Capstone Project Proposal: Starbucks

DOMAIN BACKGROUND
Machine Learning

PROBLEM STATEMENT
Does the reward size of the coupon influence the decision of whether the customer uses it or not?
If so, how big of a reward does an offer need to be to entice the customer to use the offer to make a transaction?

DATASETS AND INPUTS
portfolio.json
- id (string) - offer id
- offer_type (string) - type of offer ie BOGO, discount, informational
- difficulty (int) - minimum required spend to complete an offer
- reward (int) - reward given for completing an offer
- duration (int) - time for offer to be open, in days
- channels (list of strings)

profile.json
- age (int) - age of the customer
- became_member_on (int) - date when customer created an app account
- gender (str) - gender of the customer (note some entries contain 'O' for other rather than M or F)
- id (str) - customer id
- income (float) - customer's income

transcript.json
- event (str) - record description (ie transaction, offer received, offer viewed, etc.)
- person (str) - customer id
- time (int) - time in hours since start of test. The data begins at time t=0
- value - (dict of strings) - either an offer id or transaction amount depending on the record

SOLUTION STATEMENT
Problem 1: Does the reward size of the coupon influence the decision of whether the customer uses it or not?
We can solve this by first selecting the important features, using unsupervised learning, to see if reward size is even a factor in using offers to make transactions. Intuitively, it makes sense that it does have an impact.

Problem 2: How big of a reward does an offer need to be to entice the customer to use the offer to make a transaction?
We now know reward size does influence a customer's decision. We can use a supervised machine learning algorithm to predict whether an offer will be completed or not some time in the future.

BENCHMARK MODEL
For comparison, I'll use a naive model that always predicts a customer to respond to an offer, no matter its reward size.

EVALUATION METRICS
I'll use accuracy and F-score (which encompasses precision and recall) to measure the quality of the model predictions.

PROJECT DESIGN
1. Load, explore, and visualize data
    1.1. Combine all 3 *.json* files into a single DataFrame by linking *id* in *portfolio* and *profile* with *person* and *value* in *transcript*
    1.2. Calculate and visualize percentage of offers completed: $\frac{\text{\# of offers completed}}{\text{\# of offers received}}$
    1.3. Calculate and visualize percentage of transactions with offers: $\frac{\text{\# of offers completed}}{\text{\# of transactions - \# of offers completed}}$ (Note that we need to subtract by number of offers completed in the denominator to take care of double counting)
2. Clean and pre-process data by combining all 3 data files and removing the data points with transactions
    2.1. Drop the following columns: *transaction*, *become_member_on, value*, *person, id* (from *portfolio.json* and *profile.json*)
    2.2. For incomplete features: In *income*, fill in with the mean of the column
    2.3. For categorical features:
        2.3.1. In *gender*, one-hot encode to create *M* and *F* columns
        2.3.2. In *events* (which are our labels), map *offer_received* and *offered_viewed* to 0, and *offer_completed* to 1
        2.3.3. In *channels*, one-hot encode to create *web*, *email*, *mobile* and *social*
        2.3.4. In *offer_type*, one-hot encode to create *bogo*, *informational* and *discount*
    2.4. For numerical features, normalize them: *reward*, *difficulty*, *duration*, *age*, *income*, *time*
3. Split data into train/test datasets (labels will be the *events* column)
4. Reduce dimensionality using PCA and find vectors of maximal variance
    4.1. Keep top n principal components that account for at least 90% of the variance
5. Test and evaluate several supervised learning algorithms with default parameters and pick the one with the highest accuracy and f-score
6. Tune Model with most promising algorithm
    6.1. Tune hyperparameters
    6.2. Check and address a potential class imbalance issue (with may result in overfitting) by calculating percentage of training data labeled as 1 - if it's low, weigh errors in classifying negative and positive examples to have equal impact on training loss
7. Evaluate model using accuracy and f-score
8. Compare model performance against a naive predictor that always predicts a customer to respond to an offer (in other words, the naive model always outputs 1)
9. Done! Clean up all SageMaker resources!