Classifying visitors to an e-commerce site as buy or not

Jason Salazer-Adams

Introduction

The data utilized in this project was from Kaggle and provided by RetailRocket. The data were gathered from a real e-commerce website between May 2015 to September 2015. The data set consisted of 3 different types of data,

1. events - Transaction log of every event logged by a visitor.
2. item properties - Transaction log by week of each item property, i.e. categoryid or item availability.
3. category tree - Hierarchal data relating categories to parent categories. This data was not utilized in this analysis.

The data was highly anonymized to protect the identity of the e-commerce web site. The more interesting data, such as price, discount, item category names, etc., were all hidden. However, there was information about the behavior of each visitor. The behavior tracked in the data set were the following 3 events,

- view - A visitor navigated to a page to view an item.
- addtocart - A visitor decided to add the item they were viewing to their virtual cart.
- transaction - A visitor decided to buy the item that was added to their virtual cart.

Given this rich information about each visitor, then I wanted to try and predict whether or not a visitor will buy any item, based solely on how the visitor interacted with the site previously. Here are three reasons why an e-commerce business would be interested in predicting if a visitor will predict the next time they visit their site.

1. Knowing who will buy next can result in targeted marketing to those who will not buy.
2. Implementing a targeted marketing strategy results in an optimal allocation of a marketing budget.
3. Profit erosion will be minimized as all discounts or promotional programs are targeted to the right visitors.

Methodology

First, I defined what a session would be for each visitor. I needed to aggregate the events of a visitor into a session, so that I could understand what a user did the first time they interacted with the website, and the next time, and so forth. The reason for aggregating is that each visitor comes to the website for an unknown intent and may record multiple events to satisfy their intent. The goal of the model was to use the event data from a visitor's first session to predict whether or not the visitor will buy at least one item in their second session. A new session was created each time at least 30 minutes elapsed between events logged by the visitor.

The total number of events in the data set was approximately 2.7M. These events were then aggregated into 1.7M sessions. The data was finally filtered down to only those visitors which visited the e-commerce site at least twice between May-September 2015. The final data set used for feature generation and modeling was 181,000 pairs of sessions. The first session is what will be used to generate features to predict the buy or no buy of the second session.

Features

I engineered 6 features to be utilized to predict whether or not the second session will result in a buy or not.

1. view_count - Count of the number of view events in the first session.
2. session_length - Length of the first session in minutes.
3. item_views - Count of unique items viewed in the first session.
4. add_to_cart_count - Count of addtocart events in the first session.
5. transaction_count - Count of transaction events in the first session.
6. avg_avail - Average availability of the items viewed during the first session. For example, if 3 items were viewed and 1 of them was not available then the average availability for the first session would be 66%.

The item availability data came from the item properties data, and some items in the event data did not appear in the item properties data. Those events with items which did not have item availability were dropped from the analysis data set.

Modeling

The train/test split was 75/25 and stratified sampling was utilized. I trained initially with Logistic Regression, Random Forest, and Gradient Boost. I found Logistic Regression with StandardScaler transformations was consistently performing worse than the other two and was dropped early in my analysis. I focused on tuning Random Forest and Gradient Boost with a 10-fold BayesSearchCV. I also experimented with different up and down sampling techniques to determine if this had an impact on my training performance of the two types of models. I decided to select my model based on AUC as I believe AUC to be a good measure of how well a classification model does in segregating the data into it's respective classes. The results of the different tests are below.

Model	Sampling	AUC
Random Forest	SMOTE	0.779
Gradient Boost	SMOTE	0.857
Gradient Boost	SMOTE + Tomek	0.875
Gradient Boost	SMOTE + Edited Nearest Neighbors	0.999

The AUC reported by BayesSeachCV is the best AUC of the test data generated in the cross validation process. The Gradient Boost with SMOTE + Edited Nearest Neighbors resulted in the best AUC and was selected as the best model.

I wanted to compare my model to some other model in an effort to derive value of the trained Gradient Boost model. I defined a baseline model as a model that would predict the outcome of the second session as the same outcome as the first session. If the first session resulted in a buy, then the second session would be predicted as a buy. Likewise, if the first session resulted in a not buy, then the second session would be predicted as a not buy. I felt this was a reasonable model as the data set contained 256,000 items, and thus if a visitor buys one item then it is not unreasonable for the visitor to come back again and buy a different item.

Results

First, I compared the F1-score, recall, and precision between the two models with the training data without any sampling methods applied. The results are below.

Score	Gradient Boost	Baseline
F1	0.21	0.079
Recall	0.54	0.073
Precision	0.13	0.087

The trained Gradient Boost model performed well. I then ran the same two models on the test data, and the results are below.

Score	Gradient Boost	Baseline
F1	0.08	0.083
Recall	0.21	0.076
Precision	0.05	0.091

The scores for the trained Gradient Boost model got worse, and appears that I may have over-fitted the model. I did not have time to analyze this further as this seems contrary given my process to use 10-fold CV. However, the model did show an improvement compared to the baseline model in accurately identifying sessions which actually bought as buy. The model however did increase the number of false positives compared to baseline, and thus a less precise model.

The model does provide value by identifying more of the actual visitors who will buy next, and thus mitigating the risk of profit erosion. The model could be improved upon by considering these items,

- Additional history - If a visitor frequents the site, then this additional history could give insight into whether or not the visitor will buy next.
- Price / Discount data - Knowledge of the item price and/or discount of the price while the visitor viewed the item could be used as a feature in the model to enhance the predictive power of the model.
- Seasonality - Adding knowledge of known holidays or different buying seasons would also fine tune the model.

Tools

I utilized the following tools for my analysis.

- Python
    - Data storage/manipulation: pandas, numpy
    - Data Analysis: sklearn,
    - Visualization: seaborn, matplotlib
- Keynote
    - Google image search for slide design

What I would consider differently?

I underestimated the amount of time needed to transform the raw data into a data set I could then train my models. I would have liked to spend more time understanding the consequence of applying different sampling techniques as this appears to have had an impact on the overall quality of the model. I am also concerned the model I trained was overfitting based on the test results, but did not have time to research this further. I would like to try for my next project to wrap up model training with at least 3 days prior to when the presentation is due. This should allow me time to refine my story, build a cleaner presentation, and also a flask app.