Coupon Purchase Prediction

CSCE 5300.002 - Fall 2021

This repository (github.com/danwaters/brute-force-project) contains code and artifacts for Team "BruteForce" and our final project submission. This README serves as the project report, as well as the instructions and links for reproducing our work.

Looking at the GitHub repository is encouraged, because GitHub has a very good .ipynb viewer.

NOTE

It is very important to run the notebooks in Google Colab, as the data files are hosted in Google Drive and acquired via Colab's built-in `!gdown` command.

Team members are listed in the separately submitted presentation deck to protect privacy.

About the project

The challenge is simple: using the data and the parameters of this Kaggle competition, with a goal of predicting which coupons a user is likely to purchase given some training data.

Exploratory data analysis, data structures, assumptions, and more are covered extensively in our presentation. This report contains more high level information and instructions to execute the code.

The team

A list of team members, tasks, and assignees is available in our project management sheet.

Effort since the presentation

The highest score yet was achieved by augmenting 91 empty coupon lists with the Gradient Boosted Trees predictions for those users. This previously brought the mean average precision down by quite a lot, since the AP for those users was zero.

The data

The data is provided by Recruit Ponpare. The tables, relationships, and field descriptions are described in detail here. Exploratory data analysis is described in our presentation document.

Japanese -> English translation

The original data had several fields in Japanese. Since nobody on our team speaks Japanese, it presented a problem for understanding the data and EDA tasks. Because place names don't work so well with machine translation, it took a couple of iterations to get the translation right.

In the original translation pipeline notebook, two different translation libraries are used (txtai and pykakasi) are used, to some comic effect. In the v2 pipeline notebook, a simple mapping is used instead. While tedious, it resulted in far more readable and accurate place names, genres, and capsule texts. There are not many translations required; most are repeated, so it was worth the time to do this manually.

Downloading the data

The translated data is temporarily stored on Google Drive (caution: it will be deleted at the end of the semester after grades are put in). The data can be easily downloaded with the `!gdown` commands found in the notebooks.

Preprocessing

Preprocessing (for everything but the decision tree / random forest approach) consisted of one-hot encoding categorical variables, replacing NaN values with 0 (as these were not necessary for training), uniting user data with coupon data, and dropping unnecessary columns. The one-hot encoding expanded eight columns to over 140 features, which were then fed to logistic regression and cosine similarity algorithms.

TensorFlow Decision Forests and Gradient Boosted Trees thankfully did not require any preprocessing other than feature selection. These approaches also required far less code, and were faster to iterate with. The TFDF blog post is a great place to start to understand this great library.

Cosine Similarity and Logistic Regression approaches employed feature selection to facilitate hybrid collaborative filtering - those coupons which match a user based on demographics or content.

Evaluation

The Kaggle competition submission is a .csv file. Each row contains a user ID hash and a space-delimited list of 10 coupons. Kaggle evaluates these submissions against a held-out test data set containing coupons that a user actually purchased during the test period.

The evaluation criteria is Mean Average Precision (k=10), taking the average precision of the 10 coupons for each user, averaged across all ~22K users.

The solutions

First, a baseline model of random selection was created with the following code. You can also find this code about halfway down in the Coupon EDA notebook.

```
# create random baseline
user_list = df_users['USER_ID_hash']

rows = []
for u in user_list:
  coupon_list = df_c_list_test.sample(n=10, replace=False)['COUPON_ID_hash']
  coupon_list_str = ' '.join(coupon_list)

  row = {'USER_ID_hash': u, 'PURCHASED_COUPONS': coupon_list_str}
  rows.append(row)

df_pred = pd.DataFrame.from_dict(rows)
df_pred.to_csv('sample_submission.csv', header=True, index=False)
```

Solution	Notebook URL	Mean Avg Precision @ 10	Notes
Random Selection Baseline	ipynb, pdf	**0.00051**	Randomly assign coupons
Logistic Regression #1 (Browsing Data)	Evolved into #2. ipynb, pdf	*0.00028*	Worse than baseline performance due to class imbalance.
Logistic Regression #2 (Browsing Data)	ipynb, pdf	0.00061	Slightly better with class imbalance corrections (stratified splits, scaling, bias tuning, etc)
Cosine Similarity (Purchase History)	ipynb, pdf	0.00269	Great score, computationally expensive
TF Random Forest (Browsing Data) (also tried CART and untuned Boosted Trees with same result)	Very minor code changes from the one below; not included here	*0.00047*	Easy configuration, needs tuning
TF Gradient Boosted Trees (Browsing Data)	ipynb, pdf	0.00263	Amazing results for imbalanced browsing data
New since presentation: Cosine Similarity backfilled with Gradient-Boosed Trees	ipynb, pdf	**0.00283**	Best results so far.

Our Conclusions

- Browsing history has less-than-expected correlation with purchasing habits. Class imbalance did not help.
- Past purchasing habits are better indicators of future purchases.

- Hybrid collaborative filtering measured using cosine similarity on purchase data far outperforms logistic regression (as a binary classifier) on browsing data, again, mostly due to class imbalance.

HOWEVER:

- TFDF's Gradient Boosted Trees algorithm, with appropriate hyperparameter tuning, performs at the same level as cosine similarity, though they are trained on browsing data. This merits further investigation.

Resources

- Our Presentation: Includes more detailed information about exploratory data analysis, assumptions, future enhancements etc. in a well-organized flow.
- The Kaggle Competition: to learn more about the target problem and the data which was provided.
- Bonus: Interview with competition winner Halla Yang. We discovered this interview *after* doing our project (unfortunately), but we learned that we are on the right track with gradient boosting. Halla did far more work on the feature selection than we did. Halla incorporated windowing, locality, and more, with a focus on time series problems, which is not how we approached the problem. It is very interesting to see the process of a competitions grandmaster approaching this exact problem.