Johnny-Foreigner / **predicting_purchases**

0 stars 0 forks

Star Unwatch

Code Issues Pull requests Actions Projects Wiki Security

We found potential security vulnerabilities in your dependencies.

Only the owner of this repository can see this message.

See Dependabot alerts

master

Johnny-Foreigner Done? … 7 minutes ago **39**

View code

README.md

Table of Contents

Data

Data Download Instructions

Oct 19 CSV

Nov 19 CSV

Dec 19 CSV

Jan 20 CSV

Feb 20 CSV

Data_Exploration

Exploratory Data Analysis

Figures

Reports

Final Report Notebook

Final Powerpoint Presentation

Final Readme

ReadMe

ReadMe

Environment

Environement

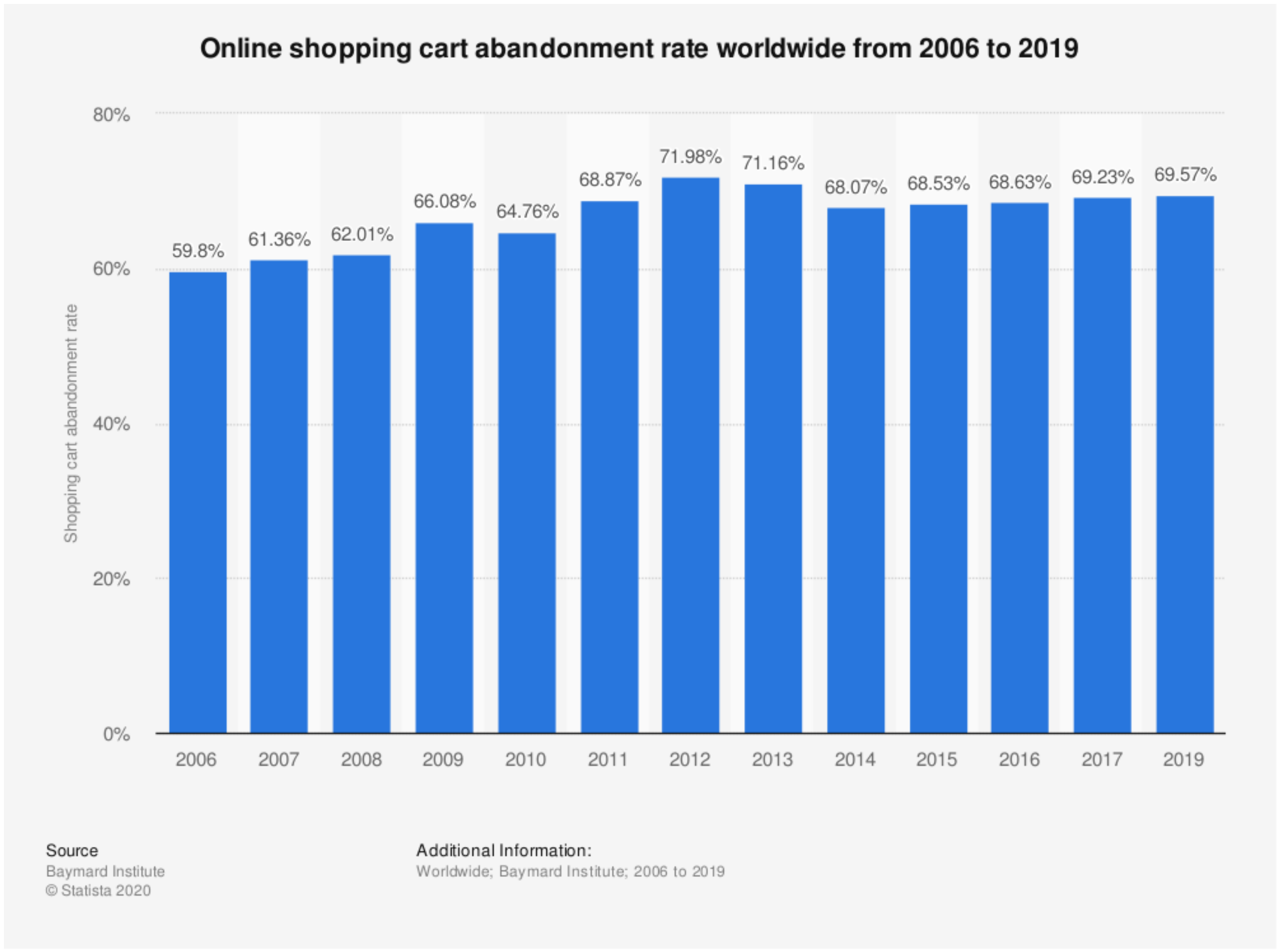
Business Understanding

In the last decade, e-commerce has fundamentally changed how we live our lives through how we shop. Companies such as Sears have gone bankrupt over the years, making the transition from brick and mortar to an online e-commerce marketplace, however other companies such as Chewy, have been able to exploit e-commerce to become a market leader in their category.

A study by emarketer.com found that the pandemic has had beneficial effects on US e-commerce. Sales will reach $794.50 billion this year, up 32.4% year-over-year. That’s a much higher growth rate than the 18.0% predicted in the Q2 forecast, as consumers continue to avoid stores and opt for online shopping amid the pandemic. By the end of the year e-commerce sales will reach 14.4% of all US retail spending for the year and 19.2% by 2024. If you further dig into the data and exclude gas and auto sales (categories sold almost exclusively offline), ecommerce penetration jumps to 20.6%.[1]

With e-commerce growing at such an unprecedented rate, many companies are to capitalise on this change in consumer behaviour. It comes as no surprise to many that purchasing items online is a different process from buying an item in a store. While in a store an employee can help guide a customer to items they are both looking for and items they may want to consider purchasing, many e-commerce marketplaces dont have the same leverage; it's much easier to close out of a "You should buy" popup, rather than to ignore the advice of an instore expert.

This has presented a serious challenge to e-commerce stores in the form of cart abandonment. Cart abandonment is when a customer leaves without buying, after adding an item to their cart. It is a trend that has remained steady since e-commerce entered the mainstream, as seen in the below chart from statista.



Find more statistics at Statista [2]

The good news for data scientists is that during an in store purchase a customer can have a large degree of privacy, while almost every move an online customer makes is tracked and stored in a series of databases. This data can be used to produce insights into customer purchasing behaviour, and spending patterns. Using the "eCommerce Events History in Cosmetics Shop" on kaggle[3] (https://www.kaggle.com/mkechinov/ecommerce-events-history-in-cosmetics-shop) I plan to analyse the purchasing patterns in the five month history the dataset provides and predict wether a customer is going to remove an item from their cart. With this knowledge a company can then provide incentives, such as free shipping or discounts to turn that removal into a purchase.

Data Understanding

The kaggle e-commerce dataset contains behavior data for five months (Oct 2019 – Feb 2020) from a medium sized unnamed cosmetics online store. Each row in the file represents an online event or action. All events are related to products and users.

The dataset contains the following features:

1. `event_time` : Time when event happened at (in UTC).
2. `event_type` : What action a customer took. For more information please see below.
3. `product_id` : ID of a product
4. `category_id` : Product's category ID
5. `category_code` : Product's category taxonomy (code name) if it was possible to make it. Usually present for meaningful categories and skipped for different kinds of accessories.
6. `brand` : Downcased string of brand name. Can be a nan value, if it was missed.
7. `price` : Float price of a product. Present.
8. `user_id` : Permanent user ID.
9. `user_session` :Temporary user's session ID. Same for each user's session. This value is changed every time user come back to online store from a long pause.

`event_type` is further broken up into four components, these are:

1. `view` - a user viewed a product
2. `cart` - a user added a product to shopping cart
3. `remove_from_cart` - a user removed a product from shopping cart
4. `purchase` - a user purchased a product

An example of a purchase funnel, may be three chronological rows, with the rows sharing `user_session` and `user_id` values. Where the first row is a view, the second is cart, and the final is a purchase.

Data Preparation

Before we start here please see the data download instructions to get the relevant datasets from Kaggle and ready for data preperation.

First: After importing our data, I first converted the CSV files to a Pandas DataFrame, and dropped rows containing `cart` or `view` as they are irrelevant to our prediction. You must first add an item to the cart before you can purchase or remove the item.

Second: I dropped the redundant feature `category_id` and filled in the NA's within `brand` as `unknown` as well as dropping the remaining NAs from the DataFrame. I also dropped any rows with a negative value in `price` .

Third: Now that are data had been cleaned up, I then needed to establish what our target column is and set it to y, with our predictors being set to X. I set y equal to `event_time`

Fourth: As our DataFrame doesn't have many feautres, I will have to build out our own features to help our models perform better. As we have a timestamp for every event, I will start with Feature Engineering using time.

Fifth: Now was the time to split the data up into a test and train set. Using `train_test_split` method in python, I split the data at a 80/20 ratio.

Sixth: Taking our training set, I then used LabelBinarizer and OneHotEncoder on several feature in the natives dataset, and several of the features I built above to create many new features.

Finally: I saw we had roughly 2:1 class imbalance in our dataset so to remedy this I SMOTEd the data to balance our targets. The data was then ready for modeling.

Model

Our first step in creating a predictive model was to choose a target metric. It was more important for our stakeholders to catch those people who are about to remove an item from their cart. If I can correctly predict those events stakeholders are able to incentive a user to make a purchase.

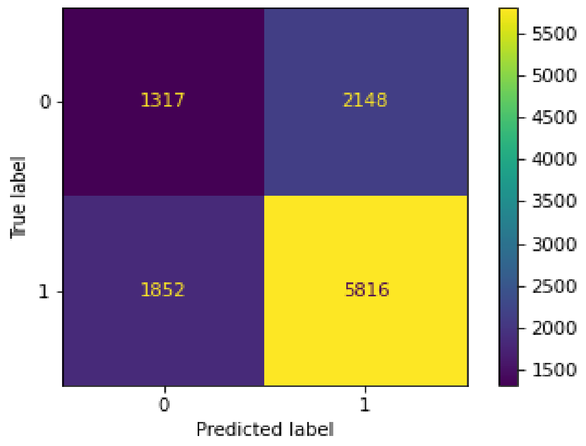
The metric I used for this model was a recall score on removed_from_cart events. This score specifically reduces false-negative results, which would lead to items being abandoned when the model has predict they will be purchased. However, it may also allows for more false positives, possibly resulting in resources being spent to convert a sale, that would have been made organicly.

I tested several different models without hyperparameter tuning. These included: LGBMClassifier, DecisionTreeClassifier, and a RandomForestClassifier. Upon comparing the scores of their classification reports I settled on the RandomForestClassifier as the best model.

After hyper tuning select parameters to improve the quality of our model I settled on a RandomForestClassifier with n_estimators set to 400, max_features set to 100, and max_depth equalling 115. The criterion was set to gini, and Bootstrap was set to True. The rest of the parameters were left as default variables.

Evalutaiton and Deployment

My focus was producing the highest Recall on The model returned a Recall Score on Class 1 (removed_from_cart) of .76 on the test set, and .87 on the training set. Recall on Class 1 measures what percentage of the truly removed_from_cart events our model correctly identifies. Our model correctly predicts truly removed_from_cart events 76% of the time on our test data.



The Best Model I built was very poor at predicting Class 0 (Purchased) events. While on our training data our model performed admirably with a recall of .83 and an f1 score of .85, it did not perform as well on our test data. On our test data our recall and f1 on our Class 0 events was .38 and .40 respectively. This was dissapointing.

Summary and Next Steps

I have addressed the problem of items being removed from the cart by creating a predictive model to identify events that will likely result in an item being `removed_from_cart` . This will allow e-commerce executives and key stakeholders to push incentives on those that are at risk of not purchasing an item. These could be:

1. Free shipping
2. Discounts
3. Bundles

I studied the problem and the data available, iterated through several model prototypes, and developed a model that successfully predicts 84% of removed from cart events. I hope my predictive model will assist companies and organizations in targeting those at high risk of removing an item from their cart. You've done the hard work getting customers to visit your site, don't lose easy money by not closing the sale.

With regards to next steps there are a few directions I would like to take a look out at our Class 0 (purchased) scores. I believe with some more feature engineering I could improve the scores for that class across the board. I would also be interested in further hyperparameter tuning. Due to time and computational power constraints I wasn't able to test as many parameters as I would have liked.

Navigation Tree

```
predicting_purchases
(Project Folder)
    |
    •README.md (Current file. Markdown file Containing Information on
Project Purpose, Process, and Findings)
    |
    •predict.yml (Environment file containg all packages used)
    |
    ├ data (Folder Containing Reference Data)
    |    |
    |    ├ data_download.ipynb
    |    |
    |    ├ 2019-Oct.csv (CSV file containing the raw data from October
2019)
    |    |
    |    ├ 2019-Nov.csv (CSV file containing the raw data from November
2019)
    |    |
    |    ├ 2019-Dec.csv (CSV file containing the raw data from December
2019)
    |    |
    |    ├ 2020-Jan.csv (CSV file containing the raw data from January
2019)
    |    |
    |    └ 2020-Feb.csv (CSV file containing the raw data from Febuary
2019)
    |
    ├ data_exploration (Folder Containing Notebooks Used as Basis of
Analysis as well as any Figures Used.)
    |    |
    |    ├ exploratory
    |    |   |
    |    |   ├ eda_index.ipynb (Jupyter Notebook containing misc data
exploration process, initial model development, and creation of
visualizations)
    |    |   |
    |    |   └Final_Notebook.ipynb (Jupyter Notebook containing all
finalized code from data preperation through final model evaluation)
    |    |
     |    └ figs (Folder containing all images created and stored for use
in EDA)
     |
     └ report (Folder Containing Finalized Notebooks, Presentation PDF, and
Visualizations)
          |
          ├ Final_Notebook_PDF.pdf (PDF of final code, calls functions
from .py that import and clean data, create model and visualizations)
          |
          ├ e_commerce_pp.pdf (PDF of final powerpoint presentation)
          |
          └ README.md (A backup README file)
```

Citations

[1] "US Ecommerce Growth Jumps to More than 30%, Accelerating Online Shopping Shift by Nearly 2 Years", https://www.emarketer.com/content/us-ecommerce-growth-jumps-more-than-30-accelerating-online-shopping-shift-by-nearly-2-years

[2] "Online shopping cart abandonment rate worldwide", https://www.statista.com/statistics/477804/online-shopping-cart-abandonment-rate-worldwide/

[3] "eCommerce Events History in Cosmetics Shop", https://www.kaggle.com/mkechinov/ecommerce-events-history-in-cosmetics-shop

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

● **Jupyter Notebook** 100.0%