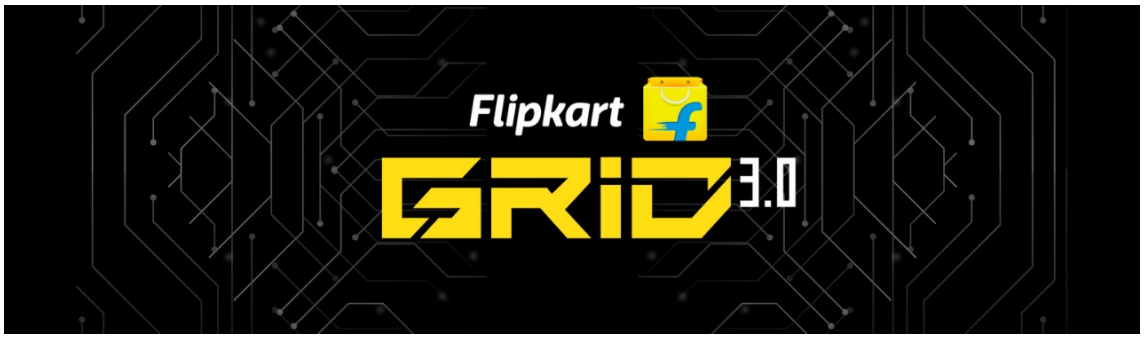


Smart Bag Creator Challenge

Institute - Indian Institute of Information Technology Kota

Team - Amigos

Team Member 1 - Arpit Agarwal

Team Member 2 - Nishchhal

Team Member 3 - Arjun Saini

Objective

Deliverable 1:

A scalable algorithm/approach with block diagram and detailed explanation to achieve the below: Given a set of users and their order history: Identify repeat purchase products for all users along with the frequency with which they are repeating Identify groups of users who showcase similar buying needs Create a smart basket for every user based on the following inputs: Time of Visit Past Purchases Relevant products purchased by similar users Building for New users with no purchase history would be given extra points

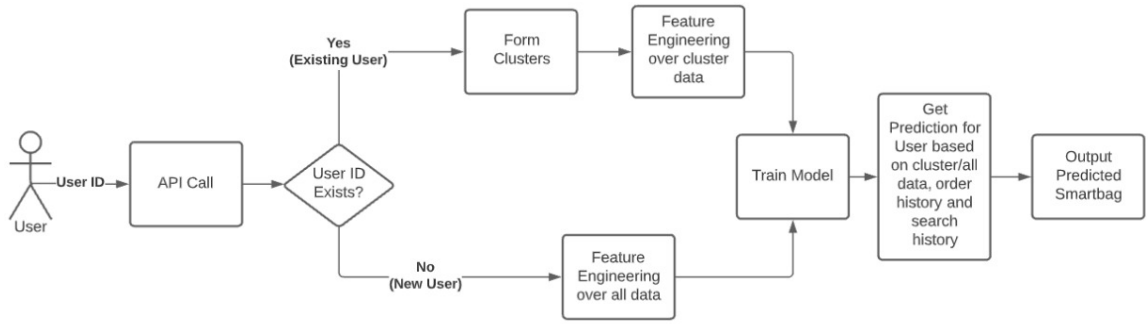
Assume that user browse history and order history would be available to you from Flipkart's platform

Deliverable 2: Design a robust, responsive and performant Progressive Web Application to showcase the list of products (Smart Bag) derived from Deliverable 1 with the following capabilities: User must be able to move items from Smart bag to the Cart in an easy manner Ranking of these products should be dependent on the relevance of the product to the user. [Optional] Nudges to add additional products - Bigger Pack Sizes, Free Sponsored Products

Methodology

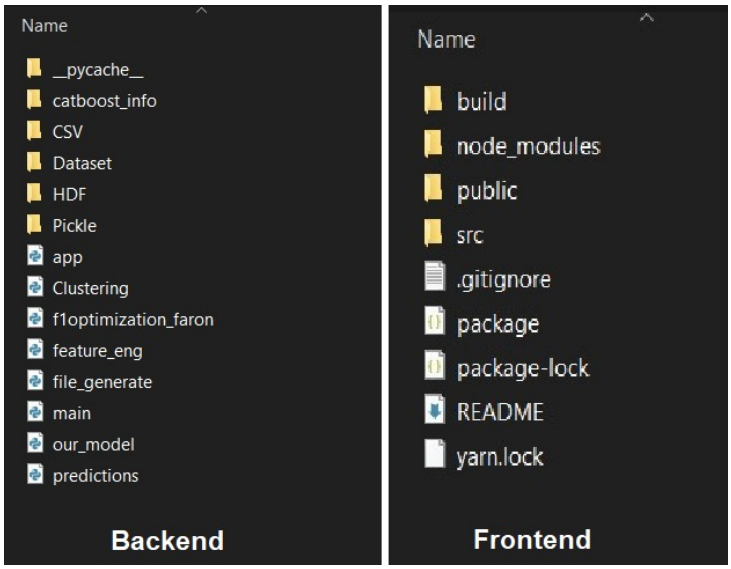
We solved the SmartBag creator challenge problem of Flipkart Grid 3.0 using React for frontend and flask for backend of our project.

As Flipkart database gets updated frequently, we decided to fetch the data, train on it and update smartbags for all users after a fixed period of time. For example, after every 12 hours the users' smartbag gets updated with the most recent predictions using the most recent data. For predicting the smartbag for a certain user, rather than using the data of all the users we decided to make clusters of similar users and use only the data of the users in that particular cluster of which, the user for whom we are predicting the smart bag, is part of. By doing so, the predicted items will be more relevant to that particular user.



Data Files

The following directory structure is followed for backend and frontend folders:



For running the application the frontend and backend servers should be running separately at localhost:3000 (Frontend) and 0.0.0.0:8080 (Backend)

Backend

The directory structure should be same as shown in image above are:

- Dataset : The data on which we trained and tested our model is included in this folder.
- CSV, Pickle, HDF : Contains all the respective files generated through feature engineering, needed for training and testing our model.
- Requirement.txt : To install all the libraries required to run the project.

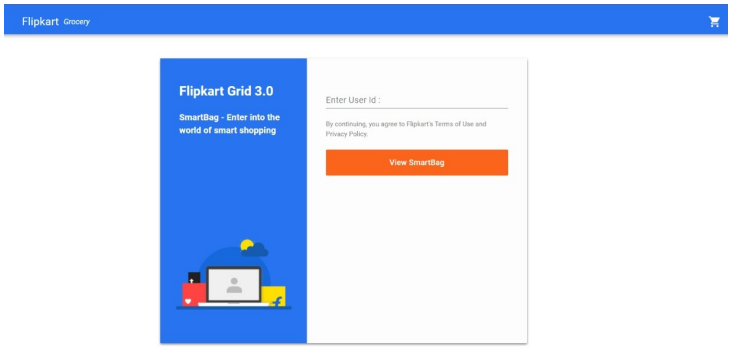
Run app.py to start the backend server at 0.0.0.0:8080

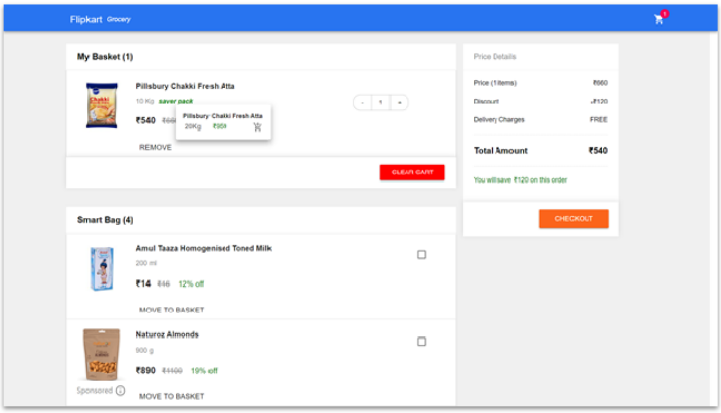
- 0.0.0.0:8080/predict : It is a post route which takes input as an user id and returns the list of objects in JSON format.

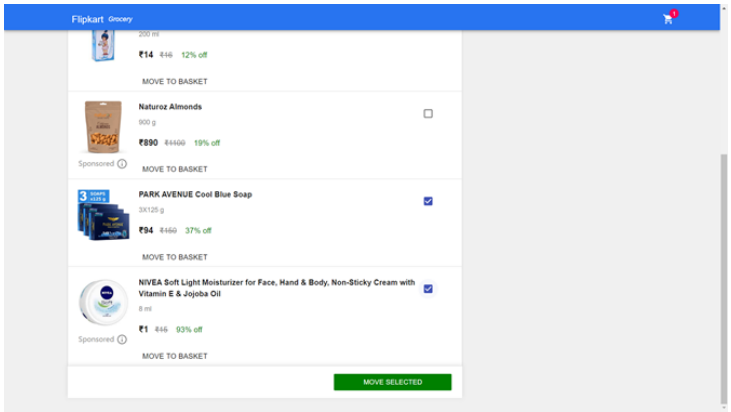
Frontend

Our fronted is based on react.js. Firstly, we have to install node modules using "npm install" command Using "npm start" command, run the frontend at localhost:3000

- localhost:3000/ : It contains the form to input user id and call the appropriate API which in turn leads to the smart bag page containing the recommended products.







This is how nudges (big saver packs and sponsored products) will be shown.

Feature Engineering

Dataset:

- departments.csv : Contains the types of products available and searched by users.
- orders.csv : Contains details of orders placed by users. Contains the previous and latest order details, along with future orders whose predictions are to be made.
- products.csv : Contains details of each product available.
- previous-orders-data.csv : Contains product-wise details of every previous order.
- training-orders-data.csv : Contains product-wise details of latest order data for every user.

For existing users, we have formed clusters of similar users on the basis of K-Means clustering and featured engineered and then train the model for a particular cluster to get most relevant predictions.

For new users, we have trained our model using the data from whole dataset.

We generated 3 type of features -

1. Product only features:
   (a) product-reorder-rate : How frequently the product was reordered regardless of the user preference ?
   (b) average-pos-incart : Average position of product in the cart
   Next 3 features are picked as they were most reordered product type/ dept. Now these values are then reduced to 3 columns using Non-Negative Matrix Factorization, to reduce sparsity
   (c) p-reduced-feat-1
   (d) p-reduced-feat-2
   (e) p-reduced-feat-3
   (f) department-reorder-rate : How frequently a product is reordered from the department to which this product belongs
2. User only features:

(a) user-reorder-rate : Average reorder rate on orders placed by a user.

(b) user-unique-products : Count of distinct products ordered by a user.

(c) user-total-products : Count of all products ordered by a user.

(d) user-avg-cart-size : Average products per order by a user = average cart size.

(e) user-avg-days-between-orders : Average number of days between 2 orders by a user.

(f) user-reordered-product-ratio : Number of unique products reordered / number of unique products ordered

3. User Product features:

   Now that we have created product only and user only features , we will now create features based on how user interacts with a product.

   (a) u-p-order-rate : How frequently user ordered the product.

   (b) u-p-reorder-rate : How frequently user reordered the product.

   (c) u-p-avg-position : Average position of product in the cart on orders placed by use.

   (d) u-p-orders-since-last : Number of orders placed since the product was last ordered.

   (e) max-streak : Number of orders where user continuously brought a product without miss.

Merge above features :

Now merge these independent features (user only features , product only features, and user — product features) → call it as merged-df .

How to get more accurate predictions :

- Product features based on time

  feature : reorder frequency of a product given any hour of the day

- Product features based on day of week

  feature : feature: What is the reorder frequency of any product given any day of week ?

- Product features based on difference between 2 orders

  feature : How frequently a product was reordered given a difference between 2 orders (days) contains the product.

- User features based on difference between 2 orders

  feature : How frequently user reorders any product given a difference between 2 order (days).

- User — product reorder rate based on difference between 2 orders

  feature : How frequently user reordered a product given difference between 2 orders (days).

These features can also be used to improve the relevance of predicted item if we have sufficient data.

Training Model

This problem is restructured into a binary classification problem, where we will predict the probability of an item being reordered by a user.

Performance Metric -

Mean F1-Score — Mean of all F1 Scores for every order.

Since we need to know how many of the actual recommended products match with predicted ones, we will use F1 score on each order. For all orders combined, we will take mean of those F1-scores.

We tried and compared two different approaches, and selected the best performing one.

1. XGBoost Classifier
2. CatBoost Classifier

The code includes only the Catboost one as it performs better

We can improve performance of model and get more better results by enabling some more features(x,y,z,w) which we have commented because of scarcity of data.

Conclusion

If we had adequate amount of data, we could have improved model to get more relevant predictions for each user. But still we managed to make it as accurate as possible using whatever data we could get our hands on.

In future we plan to make it run such that, the model gets trained and updates each user's smartbag which would then be shown along with the shopping cart on the frontend. This process is repeated after a certain amount of time again and again, so that the smartbag which is shown to the user would be most relevant as the most recent data would have been used to predict it.