**Project Selection: Instacart Market Basket Analysis**

<https://www.kaggle.com/c/instacart-market-basket-analysis/overview>

**Section 1: Executive summary.**

In this competition, the problem is to use the provided anonymize data on large amounts of customer orders over time to predict which previously purchased products will more likely to be in the user’s next order.

The dataset for this competition is related to set of files describing the customers’ orders over time.

There are mainly six datasets provided. The first one is aisles.csv which includes all kinds of aisles with unique id for each of them. The second one is departments.csv which cludes all kinds of departments with unique id for each of them. The third one

Is order\_products\_prior.csv which incudes order\_id, product\_id, add\_to\_cart\_order, and binary column reordered where 1 represents the product has been ordered before and 0 represents the product has not been ordered before. The fourth one is order\_products\_train.csv which has the same columns as the order\_products\_prior.csv. The fifth one is orders.csv, which includes details of each order such as order\_id, user\_id, order\_number, etc.. The last one is products.csv, which includes details about each product like product\_id, product\_name, aisle\_id, and department\_id.

By looking through the dataset, I found out that because there are multiple of them, so I may need to combine then with specified columns such as order\_id or product id or both. Since different products are from different departments and aisles, so I think the problem can be solved using multiclass classification or cluster based on these attributes. Also, there are attributes like order day of week, order day, and days since prior order that can be used to cluster or help building the model.

**Section 2: Benchmarking of Other Solutions**

|  |  |  |  |
| --- | --- | --- | --- |
| Notebook Name | Feature Approach | Model Approach | Train/Test Perf |
| instacart ML-xgboost-last5  <https://www.kaggle.com/>  dimosraptis/  instacart-ml-xgboost-last5 | 1. Convert character variables ‘aisle’, ‘department’, ‘eval\_set’, and ‘product\_name’ into category. 2. Merge the orders DataFrame with order\_products\_prior by their order\_id, keep only these rows with order\_id that they are appear in both DataFrames. 3. Create User predictors with number of orders per customer and how frequent a customer has reordered products. 4. Create Product predictors with number of purchases for each product and the mean of reorders of products that have more than 40 purchases. 5. Create distinct groups for each combination of user and product, count orders, and save the result for each user X product to a new DataFrame. 6. Create DataFrame for the last five orders and the last five orders ratio group by user id and product id. 7. Create feature of how frequently a customer bought a product after its first purchases:    1. Calculate the numerator - How many time a customer bought a product    2. Calculate the denumerator       1. Calculate the total number of orders for each customer       2. Get the order number where the customer bought a product for first time       3. For each product get the total orders placed since its first order       4. Divide the numerator by the denumerator | 1. Merge all features created or extracted or selected from the Feature Approach. 2. Merge these features with the User DataFrame and Product DataFrame. 3. Create train and test DataFrames based on the evel\_set attribute in the orders DataFrame. 4. Apply the xgboost model to the train DataFrame.   xgb\_params = {  "objective" : "reg:logistic"  ,"eval\_metric" : "logloss"  ,"eta" : 0.1  ,"max\_depth" : 6  ,"min\_child\_weight" :10  ,"gamma" :0.70  ,"subsample" :0.76  ,"colsample\_bytree" :0.95  ,"alpha" :2e-05  ,"lambda" :10  }  watchlist= [(d\_train, "train")]  bst = xgboost.train(params=xgb\_params, dtrain=d\_train, num\_boost\_round=80, evals=watchlist, verbose\_eval=10) | 0.37674 |
| LightGBM benchmark Implementation  <https://www.kaggle.com/mandan/lightgbm-benchmark-implementation> | 1. Add three new columns to the product DataFrame that are orders, reorders, and reorder\_rate by grouping together the product\_id in the order\_products\_prior DataFrame. 2. Merge the order\_products\_prior DataFrame with the orders DataFrame. 3. Computing User Features    1. Compute the average days between orders group by user id and the total number of orders group by user id.    2. Creating features from the customer buying patterns including total items group by user id, all products group by user id, total distinct items group by user id, and the average number of product a particular user’s basket have (total items/number of orders). 4. User X Product Features   4.1 Create a new DataFrame for each unique combination of user and product including the number of times that particular product was order by the user. | 1. Separate the final DataFrame into train and test DataFrames.  2. Train the LGB model.  d\_train = lgb.Dataset(df\_train[f\_to\_use],  label=labels,  categorical\_feature=['aisle\_id', 'department\_id'])  params = {  'task': 'train',  'boosting\_type': 'gbdt',  'objective': 'binary',  'metric': {'binary\_logloss'},  'num\_leaves': 96,  'max\_depth': 10,  'feature\_fraction': 0.9,  'bagging\_fraction': 0.95,  'bagging\_freq': 5  }  ROUNDS = 100  print('light GBM train :-)')  bst = lgb.train(params, d\_train, ROUNDS) | 0.37653 |
| XGBoost with feature engineering  <https://www.kaggle.com/errolpereira/xgboost-with-feature-engineering> | 1. Merging orders and order\_products\_prior datasets. 2. Creating Features using user\_id.    1. Create total number or orders placed by each users. Max of the order\_number column.    2. Calculate average number of products bought in each orders.       1. Calculate average products in orders placed by each users.       2. Getting the average products purchased by each user.   2.3 Calculate day of the week each user orders the most.  2.4 Calculate hour of the day each placed most of his/her orders.  2.5 Calculate reordered ratio of each user.  2.6 Calculate average days between orders of each user  2.7 Calculate total items bought by each user  3. Create feature using product\_id  3.1 Calculate number of times the product has been purchased by the users.  3.2 Calculate reorder ratio for each product. [Number of times the product was reordered / number of times it was purchased]  3.3 Calculate average add to cart order for each product.  4. Creating features using user\_Id and product\_id  4.1 Calculate how many times a User has bought a product.  4.2 Calculate how many times a user bought a product after its first purchase.  4.3 Calculate how many times a customer bought a product on its last 5 orders. | 1. Merging users, products, user X products dataframes. 2. Creating Training and Testing datasets. 3. Train the XGBoost model.   #setting boosters parameters  parameters = {  'eavl\_metric' : 'logloss',  'max\_depth' : 5,  'colsample\_bytree' : 0.4,  'subsample' : 0.8  }  #Creating a XGBoost model.  # #Initializing the model  xgb = xgb.XGBClassifier(objective='binary:logistic', parameters=parameters, num\_boost\_round=10)   1. Evaluate with confusion Matrix | 0.3759265 |

Comparison:

Since there are minor scores different between these three solutions, there must be some differences that can be compared between them. All these three solutions approach features from three perspectives, the user predictors, product predictors, and user with product predictors. The first solution with the best score focused more on the the user with product predictors, while the second solution more focused on the user predictors, and the third solution more focused on the user with product predictors. Although both the first and third solution more focused on the user with product predictors and they both used the XGBOOST model, the first solution did some filters on some fields which I think can exclude some outliers. Also, some values of the parameter of the XGBOOST model are different, which can also lead to the scores different. Although the second solution created few features than the third one, LightGMB model seems to has faster training speed and higher efficiency and has better accuracy than any other boosting algorithm. Therefore, it has higher score than the third solution.

**Section 3: Data description and Initial Processing**

This section should include basic characterization of data. You should run and report basic statistics on the data and generate at least 3 visualizations. You can review other kernels to understand some different approaches to the data, but this section you are required to generate all analyses.