Big Data Analytics Final Report Template

Yini Zhang, Chenyun Zhu
Department of Statistics, Graduate School of Arts and Sciences
Columbia University in the City of New York, NY 10027
e-mail: yz3005@columbia.edu
cz2434@columbia.edu

Abstract — **Instacart is grocery focused delivery service. Customers select groceries through a web application from various retailers and delivered by a personal shopper. In our zip code, Instacart offers services at Whole Foods, Fairway, Costco, Best Market, Westside Market and so forth. In this paper, we will demonstrate how to use Instacart public datasets to recommend new products to customers, recommend the bundles of products and develop models that predict which products a user will buy again using recommendation and classification algorithms.**

Keywords - Big Data; PySpark; Recommendation; Market Basket Analysis; Machine Learning

I. Introduction

What's your daily routine that keeps you busy the whole day? Go to the gym? Shop your groceries? Instacart is a same-day grocery delivery service that can save yourself that trip to the market. It will connect you with personal shoppers in your area to shop and deliver groceries from your favorites stores in as little as an hour.

Using the Instacart Public Datasets, we would like to explore solving the following three problems: recommend new products to customers, recommend products to be bought together, and predict which products will be in a customer's next order.

The project is of great business value. Instacart, as a grocery delivery startup, has several main competitors like AmazonFresh and Shipt. In order to acquire more customers and increase customer retention, Instacart needs to provide more delightful shopping experience by making it easier to fill customer's refrigerator and pantry with personal favorites. With the right recommendation and prediction, Instacart can boosts sales and profits, and attracts more customers by helping customers save shopping time.

II. Related Works

Recent years, there are a lot of publications that analyze customer behaviors using different data mining techniques and try to help make better business decisions. There is a book detailing the classical and Bayesian multivariate statistical methodology as well as machine learning and computational data mining methods to highlight the use of data mining methods in marketing and customer support (Giudici, P., & Figini, S., 2009). Also, we read some papers related to market basket analysis to learn what other people have done to observe patterns of purchase and the technology depends on the extent to which products relate to each other (Owies, M., Qendeel, N., & Al Barri, S., 2017). Furthermore, we have seen several papers that maximize customer satisfaction through recommendation system. They offer personalized recommendation services helps improve customer satisfaction (Jiang, Y., Shang, J., & Liu, Y., 2010). We then have an initial idea of how our project should be done.

III. System Overview

In this section, we will elaborate on the datasets we used and outline the implementation of this project.

A. Datasets

The data we used was downloaded from Instacart public release, "The Instacart Online Grocery Shopping Dataset 2017". It is a relational set of files that describe customers' orders over time. The datasets are anonymized and contain a sample of over 3 million grocery orders from more than 200,000 Instacart users. For each user, between 4 and 100 of their orders are provided, as well as the sequence of products purchased in each order. It also includes the week and hour of day the order was placed, and a relative measure of time between orders.

The datasets are consisted of the following files:

aisles.csv: specifies the ids and names of aisles
departments.csv: specifies the ids and names of departments
order_products_*.csv: specifies which products were purchased in each order, their add-to-cart order, and whether they are reordered
orders.csv: specifies which set (prior, train, test) an order belongs as well as detailed order information
products.csv: specifies the products information

B. Outline

The project is divided into three parts:

First, recommend new products to customers. We identifies the features from user habits and user preferences. User habits features include the hours or days in which user places orders, the order interval, the total number of orders placed and the total number of products bought. User preferences features are the names of products customers bought, which are then mapped to vectors using word2Vec. Then similar customers are clustered into groups based on Euclidean distance using K-Means algorithm. By counting the frequency of products in each cluster, we sort the most popular ones in one cluster they haven't bought yet to recommend to customers within the same cluster.

Second, recommend the bundles of products: which product a customer will buy next after adding one product to cart. We extracted bigram features from names of product sequence in each order. Then we developed an algorithm of recommending bundled products based on the sorted bigram frequencies with some form of randomness.

Third, develop models that predict which products a customer will buy again using transactional data. We incorporated 15 new features accompany with the original 4 features to build two predictive models. In addition, we use different ranges of thresholds to train the models and determine the optimal threshold that select product into the corresponding order.

IV. Algorithm

In this section, we will introduce the algorithms employed in achieving the three project aims, including feature extraction algorithms, clustering and classification algorithms, as well as one new recommendation algorithm designed by ourselves.

A. Feature Extraction

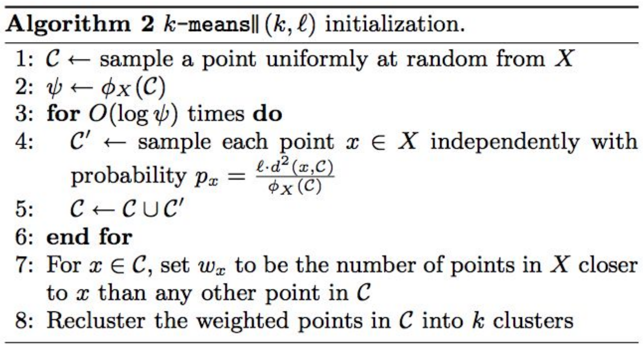
word2Vec
Word2Vec computes distributed vector representation of words. It is an Estimator which takes sequences of words representing documents and trains a Word2VecModel. The model maps each word to a unique fixed-size vector. The Word2VecModel transforms each document into a vector using the average of all words in the document; this vector can then be used as features for prediction, document similarity calculations, etc. The main advantage of the distributed representations is that similar words are close in the vector space, which makes generalization to novel patterns easier and model estimation more robust. ("Machine Learning Library Guide", 2017).

Bigram
A bigram is a sequence of two adjacent elements from a string of tokens, which are typically letters, syllables, or words. The frequency distribution of every bigram in a string is commonly used for simple statistical analysis of text in many applications, including in computational linguistics, cryptography, speech recognition, and so on (Collins, 1996). The probability of a token W_n given the preceding token W_{n-1} is equal to the probability of their bigram, or the co-occurrence of the two tokens $P(W_{n-1}, W_n)$ divided by the probability of the preceding token.

B. Clustering Algorithm

K-Means
K-means is one of the most commonly used clustering algorithms that clusters the data points into a predefined number of clusters. The algorithm we implemented here is adopted from spark.mllib, which includes a parallelized variant of the k-means++ method called kmeans|| ("Machine Learning Library Guide", 2017).

The k-means++ is an algorithm used for selecting a proper initialization of k-means for obtaining a good final solution. A major downside of the k-means++ is its inherent sequential nature, which limits its applicability to massive data: one must make k passes over the data to find a good initial set of centers. K-means|| is variant of k-means++ which drastically reduces the number of passes needed to obtain a good initialization in parallel. Experimental evaluation on real-world large-scale data demonstrates that k-means|| outperforms k-means++ in both sequential and parallel settings (Bahmani, Moseley, Vattani, et al., 2012).



```
Algorithm 2 k-means||(k, ℓ) initialization.
1: $\mathcal{C} \leftarrow$ sample a point uniformly at random from $X$
2: $\psi \leftarrow \phi_X(\mathcal{C})$
3: for $O(\log \psi)$ times do
4:     $\mathcal{C}' \leftarrow$ sample each point $x \in X$ independently with
       probability $p_x = \frac{\ell \cdot d^2(x,\mathcal{C})}{\phi_X(\mathcal{C})}$
5:     $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}'$
6: end for
7: For $x \in \mathcal{C}$, set $w_x$ to be the number of points in $X$ closer
   to $x$ than any other point in $\mathcal{C}$
8: Recluster the weighted points in $\mathcal{C}$ into $k$ clusters
```

Figure 1. Pseudocode for k-means|| algorithm

C. Classification Algorithm

XGBoost
XGBoost is short for "Extreme Gradient Boosting". It is efficient and scalable implementation of gradient boosting

framework by (Friedman, 2001) (Friedman et al., 2000). The library is laser focused on computational speed and model performance, as such there are few frills. There are three main forms of gradient boosting are supported: Gradient Boosting, Stochastic Gradient Boosting and Regularized Gradient Boosting.

There are several techniques of XGBoost are used to prevent overfitting. For instance, shrinkage scales newly added weights by a factor after each step of tree boosting. Similar to a learning rate in stochastic optimization, shrinkage reduces the influence of each individual tree and leaves space for future trees to improve the model (Chen, T., & Guestrin, C., 2016).

```
Algorithm 1: Exact Greedy Algorithm for Split Finding
Input: $I$, instance set of current node
Input: $d$, feature dimension
$gain \leftarrow 0$
$G \leftarrow \sum_{i \in I} g_i$, $H \leftarrow \sum_{i \in I} h_i$
for $k = 1$ to $m$ do
    $G_L \leftarrow 0$, $H_L \leftarrow 0$
    for $j$ in $sorted(I, by\ \mathbf{x}_{jk})$ do
        $G_L \leftarrow G_L + g_j$, $H_L \leftarrow H_L + h_j$
        $G_R \leftarrow G - G_L$, $H_R \leftarrow H - H_L$
        $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
    end
end
Output: Split with max score
```

Figure 2. Exact Greedy Algorithm for Split Finding

LightGBM

LightGBM is Gradient Boosting Decision Tree (GBDT) with Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). GBDT is a good algorithm but it is very time-consuming. With GOSS, it can exclude a significant proportion of data instances with small gradients, and only use the rest to estimate the information gain. With EFB, it bundles mutually exclusive features to reduce the number of features. The experiments on multiple public datasets show that, LightGBM speeds up the training process of conventional GBDT by up to over 20 while achieving almost the same accuracy (Ke, G., & Meng, Q., 2017). The advantages of using LightGBM include but are not limited to faster training speed, lower memory usage and capable of handling large-scale data.

D. Recommendation Algorithm

We developed this algorithm for predicting product bundles; recommend a list of products that are likely to be bought together if a customer buy a certain product. Suppose we would like to recommend k products to be bought together after product S and the number of bigrams starting with S is denoted as n.

First, we sort the frequencies for each bigram starting with the product S in decreasing order, so we have n bigrams arranged in an order from the highest frequency to the lowest. Second, we compare k with n, and fill the recommendation list with products in bigrams one by one based on the ordered bigram frequency. We consider the following two cases:

When $n \geq k$, if after sorting, there are some bigrams with the same frequency, we will pick each one with equal probability until the recommendation list reaches the total number of k.

When $n \leq k$, we will not have enough product to fill the recommendation list of product S. In this case, we will first recommend all these n products. Then goes back to the product that is of highest frequency in bigrams of S, denoted as T, and fill the rest places in recommendation list with the products followed by T, adopting the same rule as before.

V. Software Package Description

For data ETL, we utilized python pandas, python numpy, PySpark, R dplyr, R tidyverse and R forcats.

For data visualization, we used python matplotlib, R ggplot2 and R treemap.

For feature extraction and modeling, we employed PySpark.ml.feature, python xgboost, python LightGBM.

All the codes and scripts can be found at:
https://github.com/Sapphirine/Market-Basket-Analysis/tree/master/doc

VI. Experiment Results

In this section, we demonstrate the results we gained for solving each of the three project questions.

First, the classification of similar users and new product recommendation. We classified the users based on extracted features from user habits and preferences into 40 categories. The number 40 is decided according to the computation of Within Set Sum of Squared Error (WSSSE). We can reduce this error measure by increasing k, but the optimal k is usually one where there is an "elbow" in the WSSSE graph ("Machine Learning Library Guide", 2017).

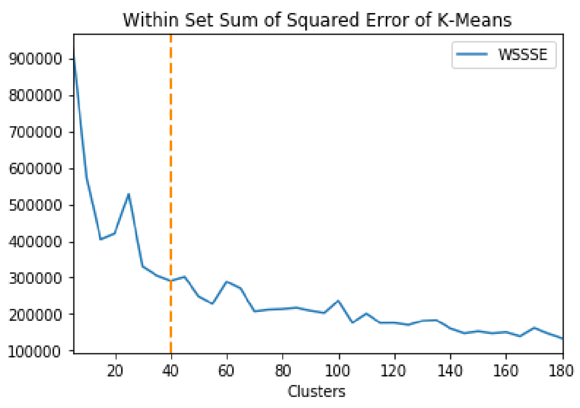


Figure 3. WSSSE of K-Means Clustering

The clustering results is demonstrated in Figure 3 below. The plot shows the distribution of each clusters.

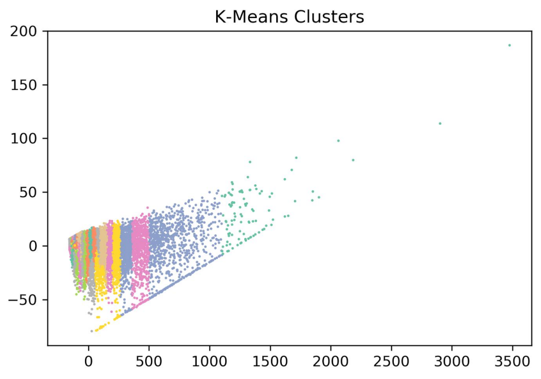


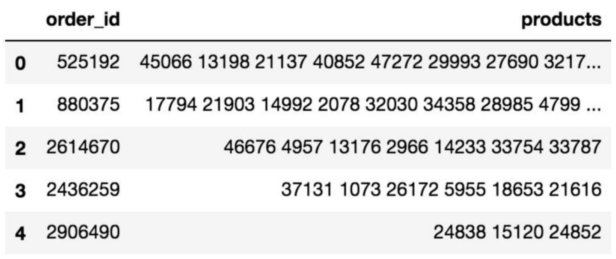
Figure 4. K-Means Clustering Results

The most popular products within each cluster are then recommended to the customers within the same cluster.

Second, recommend product bundles. The algorithm is written as functions to recommend products. We evaluate the results on test dataset.

In each order in test set, starting with the very first product, we offer a list of recommendation for this product where number of recommends is the same as the total number of products in original order. Then we see if any products later purchased in this order are in the recommendation list. If yes, it means we give right recommendation for this product, therefore label this product as a correct prediction. Next, we iterate on all products in this order to do previous calculation, and compute the percentage of correct recommends. Finally, we iterate on all orders in test dataset, and compute the average percentage of right predictions. Our final score for test data is around 0.19. This means almost 19% of products in each order in test set is from the recommendation list generated through our algorithm.

Third, build models that predict which products will be in a customer's next order. After fit the test data into the models, our idea output should be look like Figure 5 below.



	order_id	products
0	525192	45066 13198 21137 40852 47272 29993 27690 3217...
1	880375	17794 21903 14992 2078 32030 34358 28985 4799 ...
2	2614670	46676 4957 13176 2966 14233 33754 33787
3	2436259	37131 1073 26172 5955 18653 21616
4	2906490	24838 15120 24852

Figure 5. Ideal Output of Models

In order to choose the optimal threshold that selects product into the corresponding order, we first produce equivalent output with train ground truth data. That is, we split the training set into two parts: train and validation. Then, we use range of thresholds to train the model on the train set and obtain F1 score on the validation set.

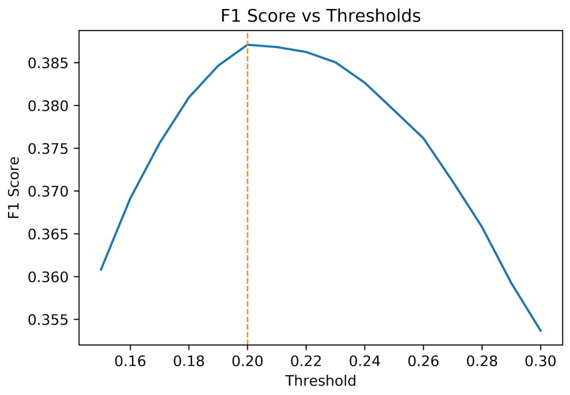


Figure 6. LightGBM: F1 vs Thresholds

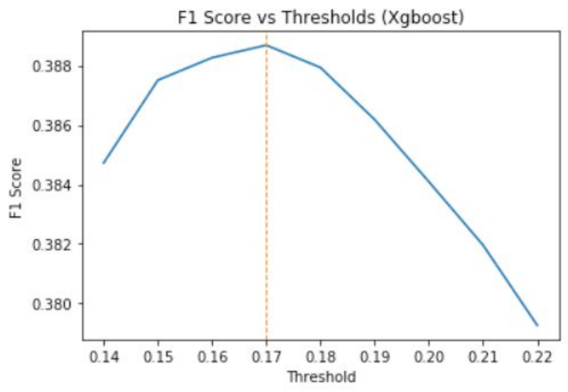


Figure 7. XGBoost: F1 vs Thresholds

We afterwards obtain two bell shaped curves. F1 score of LightGBM reaches the maximum at threshold 0.20 while XGBoost reaches the maximum at threshold 0.17. In the end, we use the optimal thresholds of each model and get accuracy for XGBoost and LightGBM with 0.383 and 0.384 on the validation set, respectively (the highest accuracy on Kaggle is 0.409).

VII. Conclusion

We implemented three major functions - recommend new products to customers, recommend product bundles, build predictive modeling on a user's next order - on the 2017 Instacart public datasets. In terms of recommendation, we have accuracy around 19%, that is, 1 in 5 products we recommend using our algorithm is actually purchased by the customer. As for predictive model, we achieve the accuracy as high as 38.4% versus kaggle highest score 40.9%. Based on our project, we do think it can create business value for Instacart and same techniques may transfer to other industries.

Future Improvements could be made in this two aspects:

- Create more correlated features
- Systematically tuning the models to improve accuracy

Acknowledgment

We would like to thank Professor Ching-Yung Lin for his inspiration and encouragement throughout the semester, as well as all teaching assistants for their kindness and efforts put into class.

Appendix

A word on Contribution:
All team members contributed equally in all stages of this project.

References

[1] Bahmani, B., Moseley, B., Vattani, A., Kumar, R., & Vassilvitskii, S. (2012). Scalable k-means++. *Proceedings of the VLDB Endowment, 5*(7), 622-633.

[2] Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System.XGBoost: A Scalable Tree Boosting System, 785-794.

[3] Collins, M. J. (1996, June). A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics* (pp. 184-191). Association for Computational Linguistics.

[4] Friedman JH (2001). "Greedy function approximation: a gradient boosting machine." Annals of Statistics, pp. 1189–1232.

[5] Giudici, P., & Figini, S. (2009). Applied data mining for business and industry. Chichester, U.K.: Wiley.

[6] Jiang, Y., Shang, J., & Liu, Y. (2010). Maximizing customer satisfaction through an online recommendation system: A novel associative classification model.

[7] Ke, G., & Meng, Q. (2017). GBDT with GOSS and EFB LightGBM.

[8] Machine Learning Library Guide. (2017, December 20). Retrieved from: https://spark.apache.org/docs/2.1.1/ml-guide.html

[9] Owies, M., Qendeel, N., & Al Barri, S. (2017). Market Basket Analysis.