Name: Sai Ram Teja

Email address: chinuteja2008@gmail.com

Contact number:

Anydesk address:

Years of Work Experience: 0.6

Date: 7th Oct 2020

Self Case Study -1: ***Instacart Market Basket Analysis***

"After you have completed the document, please submit it in the classroom in the pdf format."

Please check this video before you get started: https://www.youtube.com/watch?time_continue=1&v=LBGU1_JO3kg

---

Overview

Business Problem

Instacart is just like a big basket. They deliver groceries online. They have a delivery app where people can buy groceries. So they want to predict based on previous orders what will be their next order. This helps Instacart by reducing the amount of time taken by the user to browse through a catalog of products that the user generally reorders instead of browsing all the products available.

ML formulation

So it seems like a recommendation problem but which is actually not because we only look at previous orders of a person

not the others. So we built a model with a classification problem . Using this we can predict if an item can be reordered or not.

Metric: Mean f1_score

Here we calculate the f1 score of each order and we take the mean of it.

F1_score is the harmonic mean of precision and recall.

F1_score = (2*precision*recall)/(precision+recall)

Data set column analysis:

1) **Aisles.csv:** This dataset has two columns asile_id and aisle here asile_id is the primary key and aisle has various varieties of items. This data set has a total of 134 rows and 2 columns.
2) **Departments.csv:** It also has two columns one is department_id (d_type=int) and other one is department (d_type = str) (which has department name). There are around 21 departments and the shape of department.csv is (21,2)
3) **Order_product_prior.csv:** It has previous order contents for all customers.The various columns in this dataset are order_id, product_id,add_to_card_order, reordered and all of them are of d_type = int. The shape of the dataset is (32434489, 4).
4) **Order_products__train.csv:** Dataset for training the models. The shape of this dataset is (1384617, 4)
5) **Orders.csv:** This dataset consists of all the orders. The various columns in the dataset are order_id, user_id,eval_set,order_number,order_dow,order_hour_of_day,days_since_prior_order

6) **Products.csv:** It consists of product_id, product_name, asile_id and department_id
7) **Sample_submission.csv:** It consists of all the test order ids that we have to predict

---

Research-Papers/Solutions/Architectures/Kernels

*** Mention the urls of existing research-papers/solutions/kernels on your problem statement and in your own words write a detailed summary for each one of them. If needed you can include images or explain with your own diagrams. it is mandatory to write a brief description about that paper. Without understanding of the resource please don't mention it***

1) **Association Rule - Extracting Knowledge using Market Basket Analysis**

By Kulkarani R.V, Jitkar B.D

This paper explains the experimental analysis that has been done employing association rules using Market Basket analysis.

Let us discuss their methodology.

a) **Data Collection Method:** The entire data is collected from Shetkari Bazar in kolhapur city in maharashtra. The entire data is transactional data i.e the actual transactions are made by customers which consists of various products.
b) **Methodology:** So the actual task is to figure out if a customer is likely to purchase the product on a trip. So Market Basket analysis with association rule mining is performed on the retail of customers transaction at

store. So each item can be represented as a boolean vector i.e 1 or 0 (bought or not bought) . Boolean vectors can be analyzed for buying patterns that reflect items that are frequently purchased together. Rule support and rule confidence are two measures of rule interest that reflect the usefulness and certainty of discovered rules. Before moving to actual procedure we must know about support and confidence

Support: This measures how frequent the item set is in all transactions.

support({X}==> {Y}) = (Transactions of both X and Y)/(total no of transactions)

For example if there are around 10000 transactions and only 50 itemsets are found having X and Y in common so the probability will be 0.005 if an itemset has very low support we don't have enough information on the relationship between X and Y hence no conclusions can be drawn.

Confidence: This measure defines the likeliness of occurrence of consequent on the cart given that the cart already has the antecedents.

Confidence({X} ⇒ {Y}) = (Transaction of both X and Y)/(transaction that is having only X)

Actual work: As of now we are familiar with terms like support and confidence we can discuss about the actual work what the authors had done by taking an example that they mentioned in the paper. We have categorical data of transaction records as input to the analysis and the output of the analysis are association rules as new knowledge directly from stored data.

Table–1
Transaction Dataset

Transaction ID	Items from customers who brought more than 1 item
1	Sugar, Wheat, Pulses. Rice
2	Sugar, Pulses
3	Wheat, Pulses
4	Pulses, Wheat, Rice
5	Wheat, Pulses
6	Sugar, Wheat
7	Sugar, Rice, Pulses

Based on the data we can generate all possible association rules compute confidence and support of all association rules apply thresholds criteria like minimum support and minimum confidence

Table– 2
Combination of purchased items

	X	Y
1	[A] →	[B]
2	[A] →	[C]
3	[A] →	[B,C]
4	[B] →	[A]
5	[B] →	[C]
6	[B] →	[AC]
7	[C] →	[A]
8	[C] →	[B]
9	[C] →	[AB]
10	[A,B] →	[C]
11	[A,C] →	[B]
12	[B,C] →	[A]

If we have 3 items A,B and C we can form various association rules as shown beside

Conversion of transaction data into Binary data

Tran ID	Items from the customers Who bought more than 1 item	Tran. ID	A	B	C	D
1	Sugar, Wheat, Pulses, Rice	1	1	1	1	1
2	Sugar, Pulses	2	1	0	0	1
3	Wheat, Pulses	3	0	1	0	1
4	Wheat, Rice, Pulses	4	0	1	1	1
5	Wheat, Pulses	5	0	1	0	1
6	Sugar, Wheat	6	1	1	0	0
7	Sugar, Rice, Pulses	7	1	0	1	1
		Sum	4	5	3	6

Convert each categorical variable to binary data.

For simplicity we call A as sugar B as wheat and so on

Table–6

Obtained transaction in association rule

No	X	->	Y	%support	Confidence	Is in Rules?
3	A	->	D	43%	75 %	√
10	B	->	D	57%	80 %	√
15	C	->	A	29%	67 %	√
16	C	->	B	29%	67 %	√
17	C	->	D	43%	100 %	√
19	C	->	B D	29%	67 %	√
20	C	->	A D	29%	67 %	√
23	D	->	B	57%	67 %	√

Calculate the support and confidence of each association and pick only those combinations that have support above 25% and confidence of 65%. So these transactions are more likely to be in the basket.

Drawbacks : As the number of items are more the association rules will be more so it becomes more difficult for calculation and framing the association rules. And moreover the time and space complexity increases exponentially as the number of items increases.

2) XGBoost: A Scalable Tree Boosting System

As we mentioned the drawbacks of apriori algorithm and we also mentioned in the ML formulation we are going with classification approach we will look at XGBoost classifier. XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework.
XGBoost and Gradient boost machines are both ensemble tree methods that apply the principle of boosting weak learners using the gradient descent architecture. However, XGBoost improves upon the base GBM framework through systems optimization and algorithmic enhancements.In this paper they had made minor changes on regularisation objective.

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

$$\text{where } \Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

Where L is the differentiable convex loss function and omega is the penalty factor.The additional regularization term helps to smooth the final learnt weights to avoid over-fitting.
Beside regularisation they have also implemented two further techniques to overcome the overfitting problem. The first technique is shrinkage introduced by Friedman.Shrinkage scales newly added weights by a factor η 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. The second technique is column (feature) subsampling.

Algorithm enhancements

1) Regularization
2) Sparsity Awareness: XGBoost naturally admits sparse features for inputs by automatically 'learning' best missing value depending on training loss and handles different types of sparsity patterns in the data more efficiently.
3) Weighted Quantile Sketch: XGBoost employs the distributed weighted quantile sketch algorithm to find optimal split points among weighted datasets.
4) Cross - Validation: The algorithm comes with a built-in cross-validation method at each iteration, taking away the need to explicitly program this search and to specify the exact number of boosting iterations required in a single run.

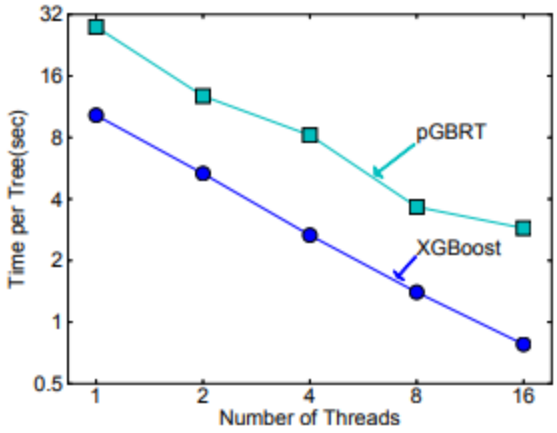
The authors in the paper implemented the XGBoost classifier on various datasets and their results are shown below.

Table 2: Dataset used in the Experiments.

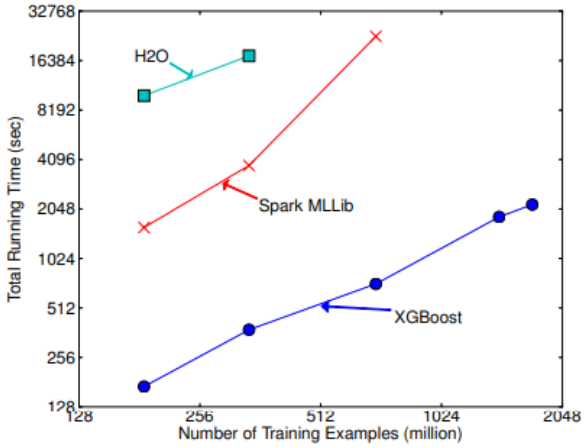
Dataset	n	m	Task
Allstate	10 M	4227	Insurance claim classification
Higgs Boson	10 M	28	Event classification
Yahoo LTRC	473K	700	Learning to Rank
Criteo	1.7 B	67	Click through rate prediction

Table 3: Comparison of Exact Greedy Methods with 500 trees on Higgs-1M data.

Method	Time per Tree (sec)	Test AUC
XGBoost	0.6841	0.8304
XGBoost (colsample=0.5)	0.6401	0.8245
scikit-learn	28.51	0.8302
R.gbm	1.032	0.6224



As u can see the results both XGBoost and scikit-learn have almost the same Test AUC but time per Tree of XGBoost is far better than compared to traditional scikit-learn.



(a) End-to-end time cost include data loading

The above graph shows Comparison of different distributed systems on 32 EC2 nodes for 10 iterations on different subsets of criteo data. XGBoost runs more 10x than spark per iteration and 2.2x as H2O's optimized version (However, H2O is slow in loading the data, getting worse end-to-end time). Note that spark suffers from drastic slow down when running out of memory. XGBoost runs faster and scales smoothly to the full 1.7 billion examples with given resources by utilizing out-of-core computation.

First Cut Approach

As we mentioned in the ML formulation we are going with a classification approach let's discuss it.

1) **Preprocessing**

   The first thing that needs to be done is preprocessing the data.Different users are identified by user_id in orders.csv file and orders can be identified by order_id each order of a user is characterized by an order_number which specifies when it has been made to specified user.Each order consists of a set of products, where each characterized by an *add_to_cart_order* feature representing the sequence in which they have been added to the cart in that order.We also need to categorical data to numerical data before applying it to model

2) **EDA**

   We need to perform EDA on the datasets so that we can know about the distribution of data. We can use dimensionality techniques to check how data is distributed in 2-D.

3) **Random Model**

   Build a random model and compute the f1 score. This would be the worst (low) f1 score and any model we train must be able to get a better f1 score.

4) **Model Approach**

   As we are doing a binary classification problem we can go for various classification models like Logistic Regression, Random Forest,XGBoost,SVM etc.

5) **Hyper parameter tuning:**

   To find the best parameters for a model we can use GridSearchCV or RandomSearchCV.Its better to choose RandomSearchCV as it is faster and yields better results.

6) **Metric evaluation:**

   As it's a mentioned to choose f1 score we will go for it.We compute the f1 score for the each model we train and select the best model.

Notes when you build your final notebook:

1. You should not train any model either it can be a ML model or DL model or Countvectorizer or even simple StandardScalar
2. You should not read train data files
3. The function1 takes only one argument "X" (a single data points i.e 1*d feature) and the inside the function you will preprocess data point similar to the process you did while you featurize your train data
    a. Ex: consider you are doing taxi demand prediction case study (problem definition: given a time and location predict the number of pickups that can happen)
    b. so in your final notebook, you need to pass only those two values
    c. def final(X):
        preprocess data i.e data cleaning, filling missing values etc
        compute features based on this X
        use pre trained model
        return predicted outputs
    final([time, location])
    d. in the instructions, we have mentioned two functions one with original values and one without it
    e. final([time, location]) # in this function you need to return the predictions, no need to compute the metric
    f. final(set of [time, location] values, corresponding Y values) # when you pass the Y values, we can compute the error metric(Y, y_predict)
4. After you have preprocessed the data point you will featurize it, with the help of trained vectorizers or methods you have followed for your train data
5. Assume this function is like you are productionizing the best model you have built, you need to measure the time for predicting and report the time. Make sure you keep the time as low as possible
6. Check this live session: https://www.appliedaicourse.com/lecture/11/applied-machine-learning-online-course/4148/hands-on-live-session-deploy-an-ml-model-using-apis-on-aws/5/module-5-feature-engineering-productionization-and-deployment-of-ml-models