WPI

Repeat Buyer Prediction for Online Merchant

Members: Qinmei Wu, Yujun Tian, Jiaoyan Chen
Data Science Department

Menu

- Background
- Data Preprocessing
- Feature Engineering
- Model and Evaluation

Background

It is important for merchants to identify who can be converted into repeated buyers. So, they can greatly reduce the promotion cost and enhance the return on investment. It is well known that in the field of online advertising, customer targeting is extremely challenging, especially for fresh buyers.

For individual merchant, it is difficult to track the user behavior and define who is loyal customer due to the lack of log data. However, with the long-term user behavior log accumulated by Tmall.com, we may be able to solve this problem.

Data Description

Training set:

User_Id	Merchant_Id	label (-1, 0, 1)
…….	……	……

user information:

User_Id	gender	age_range
……	……	……

user log:

Merchant_Id	item_id	brand_id	cat_id	timestamp	action
……	……	……	……	……	……

Data preprocessing

Data Preprocessing

- Filter the record with label -1
  - label "-1" means 'user_id' is not a new customer of the given merchant

- Missing Values

Features	Solution
item_id, brand_id, cat_id, timestamp	delete records
gender	replace with 2
age_range	replace with mode

Unbalanced Data

- Too much 0 in dataset comparing to 1
  - Two hundred fifty thousand label 1 but around twenty thousand label 0
- Models lose value
  - To reduce overall error rate, model predicate all labels as 0
- Solution
  - oversampling
    - Due to the dataset size, not take undersampling which involves removal of partial data
    - SMOTE: Synthetic Minority Over-sampling Technique.
      - Taking each minority class sample and introducing synthetic examples along the line segments joining any of K minority class nearest neighbors

SMOTE

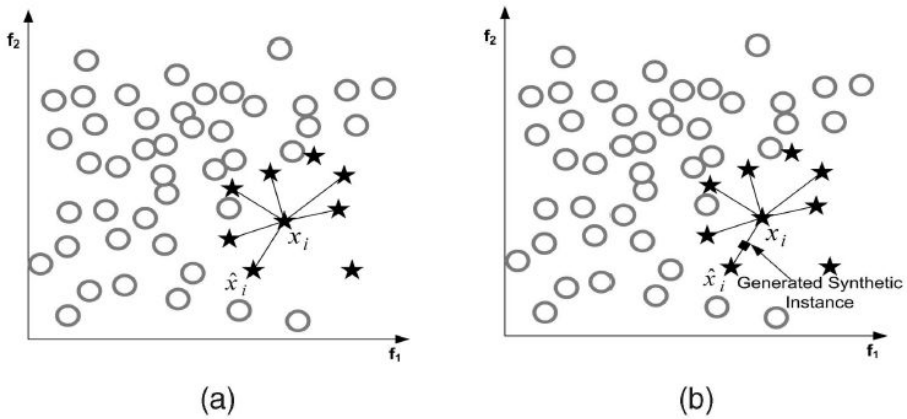
Methodology

- For each sample in minority class and compute k nearest neighbors for each sample
- Randomly choose one of the k nearest neighbors of sample
- Generate new sample as

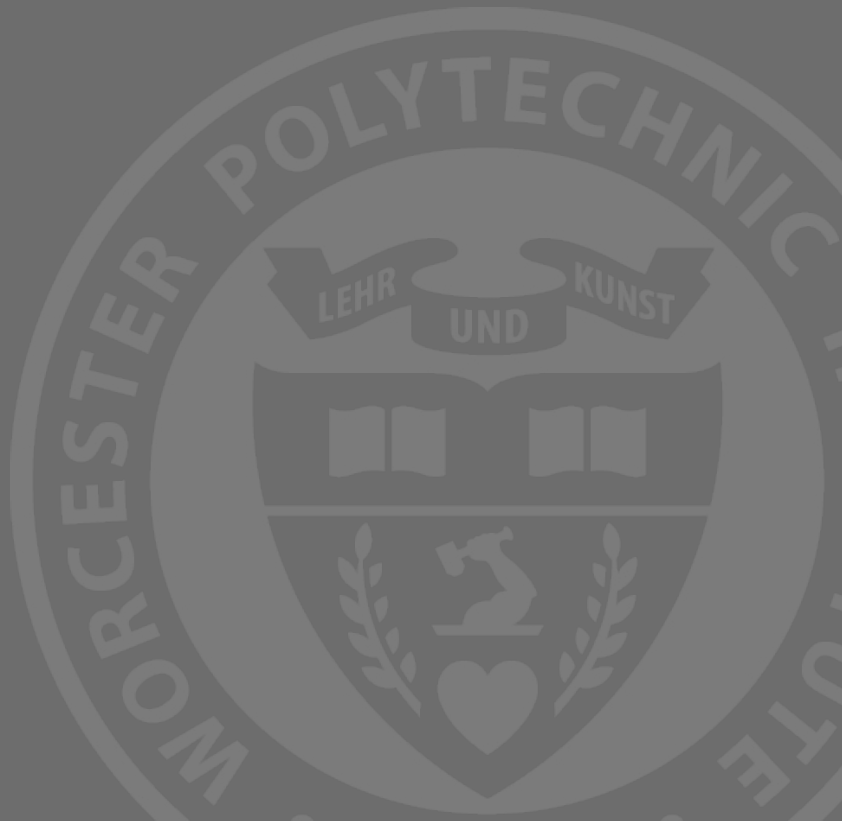
$$x_{new} = x + rand\ (0,1) \times (\tilde{x} - x)$$

Advantage

- Not making exact copies of existing examples to avoid the problem of overfitting



Feature Engineering



Numerical Measurement of Relationship

- Rating
  - A feature measures the relationships among buyers and merchants based on recommender systems
- Matrix factorization techniques for recommender system
  - latent factor models in collaborative filtering which characterizes both items and users
  - Methodology:
    - Vector q measures the extent item related to factors
    - vector p measures the extent of user's' interest based on factors
    - Inner products of two vectors to estimate users' rating for items

Steps To Compute Rating

Step 1

- Quantitate Implicit Rating
  - rank action from low to high based on user's' preference
  - click => 0
    add-to-favorite => 1
    add-to-cart => 2
    purchase => 3

Step 2

- Average rating
  - calculate the rating of merchant by averaging the implicit rating of items

Step 3

- Apply recommend ation model
  - build the recommend ation model using Alternating Least Squares
  - output feature

Feature Expanding

Aggregation features	1. Monthly click/add/favorite/purchase count for certain brand/item/category/ merchant.. 2. User's monthly action count of click/add/favorite/purchase for all items. 3.Total number of unique items/brand/category for a merchant actioned by a certain user. 4. The number of unique user who clicked/purchased/added/favourites for a given items/brands/categories/merchant 5. The number of users who bought on at least two different days from a items/brands/categories/merchant.
User Behavior	1.Counts of clicked/purchased/added/favorited of items for a certain user on the last Double 11 day. 2.Days of a certains user clicked/added/favorited/purchased items in each month. 3.The number of unique items/brands/categories/merchant that the user clicked/purchased/added/favourites in each month

- Normalization

Model and Evaluation

Valuation Indicators

Test error for all customers

- error rate = number of inaccurate prediction/total amount of customer

Test error for repeated customers only

- The point of predication is finding out repeated customers, so we compute the error rate of repeated customers only
- Error rate = number of falsely predicted repeated customers/total amount of repeated customer

Area under PR

- A PR curve is plotting Precision against Recall
- higher PR AUC, better models

Area under ROC

- The x axis of ROC is false positive rate and the y axis is true positive rate
- ROC presents the trade-off between these two rates
- The range of AUC is (0.5,1)
- Larger ROC AUC, better models

Logistic Regression

- A regression model to estimate the probability of a binary response based on predictor variables
- Result

	Training Error
All customer	**~9.33%**
Repeated customer	**~9.67%**

	Area under curve
ROC	**~90.67%**
PR	**~93.05%**

- Conclusion
  - good performance on classification
  - error rate lower than 10% and AUC higher than 0.9

SVM

- Train a Support Vector Machine (SVM) using Stochastic Gradient Descent to finds an optimal boundary between the labels. By default L2 regularization is used
- Result

	Test Error
All customer	**~50.01%**
Repeated customer	**0%**

	Area under curve
ROC	**50%**
PR	**~74.99%**

- Conclusion
  - Classified too many normal customers as repeated customers
  - The 50 percent AUC and total error rate indicated that this prediction is approximately a random guess on classification

Random Forest

- Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently. Changing the depth of tree and the number of tree to find optimized classification
- Result

	Training Error
All customer	**~7.83%**
Repeated customer	**~8.72%**

	Area under curve
ROC	**~92.31%**
PR	**~93.78%**

- Conclusion
  - perform better than logistic regression but worse than XGboost

XGBOOST

XGBoost is short for “Extreme Gradient Boosting”.

Applied to supervised learning problem

Objective Function : Training Loss + Regularization

$$Obj(\Theta) = L(\theta) + \Omega(\Theta)$$

$$L(\theta) = \sum_i (y_i - \hat{y}_i)^2$$

$$L(\theta) = \sum_i [y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i) \ln(1 + e^{\hat{y}_i})]$$

XGBOOST

Why is XGBOOST?

- Faster than GBT(gradient boosting tree)

  Split finding algorithm

- To avoid overfitting

  add regularization term
  column subsampling
  shrinkage

- Easily to be distributed

Parameters of XGBOOST

XGBoost Parameters have 3 categories

- **General Parameters:** Guide the overall functioning

booster [default=gbtree]

nthread = 1

- **Booster Parameters:** Guide the individual booster at each step

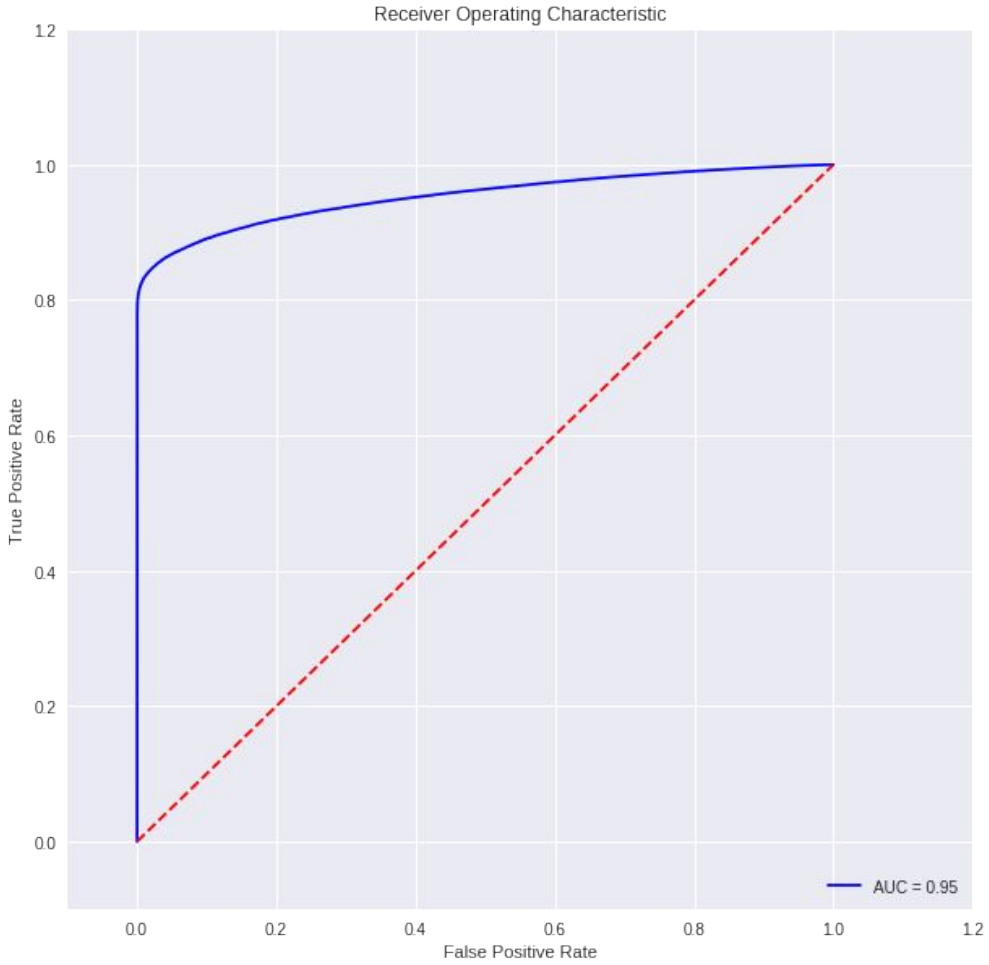
Learning rate: 0.1
min_child_weight: 5
max_depth: 5
objective: binary:logistic (logistic loss function)
lambda: 1 (L2 regularization factor)
tree_method: "auto" (achieve fast algorithm)

- **Learning Task Parameters:** Guide the optimization performed

objective = "binary:logistic"

Result of XGBOOST

ROC & Test Error



Result of XGBOOST

Train Accuracy: 95.4%

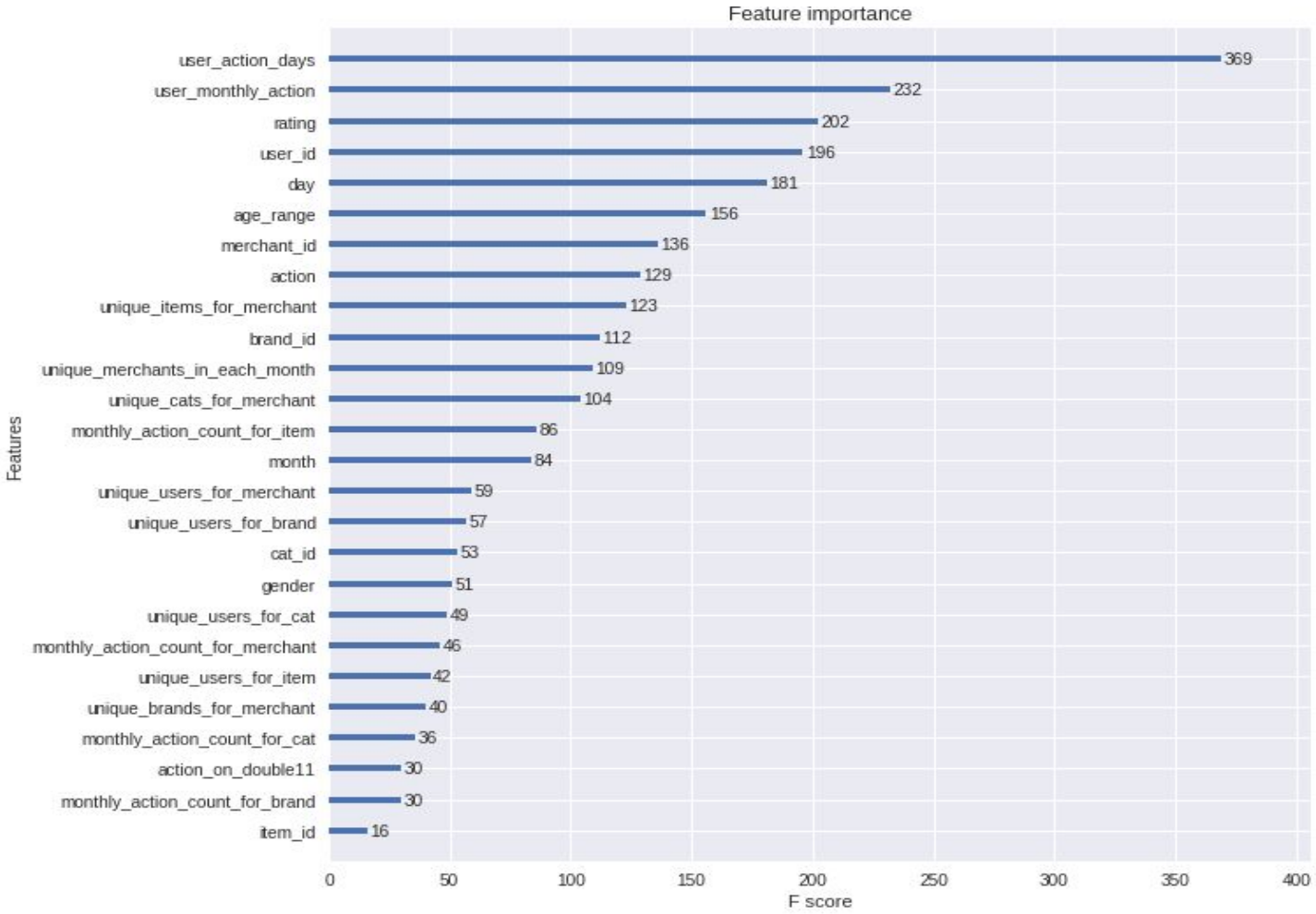
Test Accuracy: 94.14%

			Actual	
			TRUE	FALSE
			1	0
Prediction	True:	1	348794	2679
	False:	0	38556	313307

Truth data

Classifier results	Class 1	Class 2	Classification overall	Producer Accuracy (Precision)
Class 1	348794	2679	351473	99.238%
Class 2	38556	0313307	351863	89.042%
Truth overall	387350	315986	703336	
User Accuracy (Recall)	90.046%	99.152%		

XGBOOST: Feature Importance



Conclusion:

Model	Training Error	FP	AUC
Logistic Regression	9.33%	9.67%	90.67%
SVM	50.01%	0%	50%
Random Forest	7.83%	8.72%	92.31%
XGBoost	4.6%	9.54%	95%

If we pursue lower total error, XGBoost wins! Considering FP, SVM model may be a better choice.

Reference

https://www.jair.org/media/953/live-953-2037-jair.pdf

spark MLlib

https://spark.apache.org/docs/latest/mllib-guide.html

spark sql and dataframe:

https://spark.apache.org/docs/latest/sql-programming-guide.html

XGBoost:

http://www.kdd.org/kdd2016/papers/files/rfp0697-chenAemb.pdf

https://github.com/dmlc/xgboost

http://xgboost.readthedocs.io/en/latest/model.html

Logistic regression

https://en.wikipedia.org/wiki/Logistic_regression

Random Forest

https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.html

SQLite:

https://www.sqlite.org/docs.html