Data Mining: Repeat Buyers

Weisi Li
IT University of Copenhagen
Software Development and Technology
Copenhagen, Denmark
Email: liwb@itu.dk

Maria Guardiola Munoz
IT University of Copenhagen
Software Development and Technology
Copenhagen, Denmark
Email: mgmu@itu.dk

Adrian-Andrei Cobzaru
IT University of Copenhagen
Software Development and Technology
Copenhagen, Denmark
Email: adco@itu.dk

Jiun-Hung Guo
University of Copenhagen
IT and Cognition
Copenhagen, Denmark
Email: jigu@itu.dk

1. Introduction

On particular dates (such as ”Boxing-day” sales, ”Black-Friday”, ”Valentine’s Day” and ”Double 11 (November 11th)”), most merchants run big promotions in order to attract a large number of new buyers, many of who are only one-time deal hunters and targeting them is therefore of little value.

In order to reduce their promotion cost and enhance the return on investment (ROI), the merchants need to identify who can be converted into repeated buyers (i.e. new customers that would buy items from the same merchant again within six months).

In this report we use data from the ”Repeat Buyers Prediction-Challenge the Baseline” competition, based on the sales data from 6 months before and on the ”Double 11” day from Tmall.com (the second largest e-commerce website in China) to try to identify which of these new buyers can be converted into repeated buyers [1].

2. Problem description

We are provided with a set of merchants and their corresponding new buyers acquired during the promotion of the ”Double 11” day. The data set contains anonymized users’ shopping logs for the six months up to November 11th.

The data is presented in two different formats. In format one the data is divided in four files: user_log, user_info, train and test, while format two only consists of two files: train and test, where all information about the user and the behaviour logs are contained in both files. We chose format one, since it is more manageable and user-friendly for feature engineering [1].

Figure 1 shows the structure of the provided raw data.

The user activity log data, or user_log, contains the following seven data fields: user_id (a unique id for the shopper), item_id (a unique id for each product), cat_id (a unique id for the category that the item belongs to), seller_id (a unique id for the merchant that sells that specific product), brand_id (a unique id for the brand of the item), time_stamp (the date the action took place - format: mmdd) and action_type (action performed by the shopper, it is an enumerated type $\{0, 1, 2, 3\}$, where 0 is for click, 1 is for add-to-cart, 2 is for purchase and 3 is for add-to-favourite).

The user information , or user_info, contains the following three data fields: user_id (same as in user activity log data), age_range (the shopper’s age range: 1 for $<$ 18; 2 for [18,24]; 3 for [25,29]; 4 for [30,34]; 5 for [35,39]; 6 for [40,49]; 7 and 8 for $>=$ 50; 0 and NULL for unknown) and gender (the shopper’s gender: 0 for female, 1 for male, 2 and NULL for unknown).

The training and testing data contain three data fields: user_id (same as in the two above), merchant_id (same as seller_id) and label/prob (indicates if the user is a repeat buyer, it is an enumerated type 0, 1, where 1 means repeat buyer, 0 is for non-repeat buyer). In the testing data, the prob field is empty, as it is the one we have to infer.

user activity log data (6 months)

user_id	item_id	cat_id	seller_id	brand_id	time_stamp	action_type
328862	323294	833	2882	2661	829	0
328862	844400	1271	2882	2661	829	0
...	...	...	...	...	...	...

user information

user_id	age_range	gender
376517	6	1
234512	5	0
...	...	...

training data

user_id	merchant_id	label
34176	3906	0
34176	121	0
...	...	...

testing data

user_id	merchant_id	prob
163968	4605	
360576	1581	
...	...	

Figure 1. Raw data

Both the user information and the user activity log data tables contain the same amount of users, and the test and train tables contain roughly half of them, as depicted in Figure 2, where it is also shown that the number of positives (the number of repeat buyers) in the training data is 3170 or about 7.5%, which indicates that most of these new buyers are one-time deal hunters as hypothesised in the introduction.

Data statistics

users in info/log	users in test	users in train	positives	% of positives
84834	42554	42282	3170	~ 7.50%

Figure 2. Training data statistics

3. How to Run the Project

The project uses Python notebook and runs on Google Colab. Experimental data is saved on Google Drive. Here, Colab can read the shared Google file format data; the maximum size of a Google file is 20MB. However, the experimental data is up to 1.9GB, and it takes a very high cost of doing file division. Therefore, we shared the entire Google Drive project folder. To implement, we need to save the folder to our local Drive (right-click "Add to my drive") and read the experimental data from a local Google Drive.

Shared folder link:
https://drive.google.com/drive/folders/1WYQO_yFMtkq2fw59UhW1ZqD_THLUPjNh?usp=sharing

4. Data Preprocessing

In this section we will explain how we prepared the data for the analysis, that is, the tools and methods we used to preprocess our data.

First, we remove the rows with NaN values using the dropna() method from pandas.DataFrame [2] for both user_info and user_log.
We also change the name of the data field seller_id in user_log to merchant_id, so it matches the columns for the merchants ids in the other tables.
For the sake of convenience, we also separate the time_stamp column from user_log into two columns, one for the month: "time_mm" and one for the day: "time_dd".

Next, we want to fix the values that are missing from the data. Firstly we check all of the tables for missing values, the result is shown on Table 1.
Both the gender and age_range columns have an option for the data being unknown, in the case of gender it is labeled 2 and for age_range it is 0. We decide to change all of the missing values to unknown values, so we can proceed to use the rows in the analysis.

TABLE 1. MISSING VALUES

Data field	Missing values
user_info[age_range]	420
user_info[gender]	1296
user_log[brand_id]	18209

The missing values for brand_id are a little more tricky, but we realise that even though a brand can encompass multiple items, an item can only belong to one brand. Knowing this, we can create a dictionary that relates every item_id to its correspondent brand_id, and see if any of the missing values matches any of the items in the dictionary. This way, we find the missing brand_id values for 1220 items.
We are still missing the brands of 16898 items, so we will have to bear that in mind when we get to the analysis.
The DataFrames used for this project are fairly large, especially user_log having a memory usage of 836.1 MB. So we decide to change the datatype of some values from int64 and float64 to int8, int16 and int64, saving a total of 189.1 MB of memory usage.

5. Feature Extraction

In this chapter we will explain how we extracted the features for future analysis and which ones we will use.

5.1. RFE[5]

RFM is a segmentation method used for analyzing customer value, and RFE is an improvement of it. It stands for Recency, Frequency and Engagement (instead of RFM's Monetary value):

- Recency: How long ago did a customer purchase?
- Frequency: How often does the customer purchase?
- Engagement: How does the customer engage?

It is probable that the effects of the promotions will be bigger if the customer has a recent purchase (recency) and/or if the customer has many purchases in a short time (frequency).

We calculate recency from the time_stamp column of user_log of each row, comparing it to the most recent time_stamp (1111, November 11th). We calculate the frequency by counting the number of action_type per costumer. We also look at the customer's web browsing behaviour (engagement), by counting the amount of times that the costumer interacts with the merchant (action_type). Each action_type has a weight, as shown on Table 2.

TABLE 2. WEIGHTS FOR THE ACTION TYPES

action_type	0	1	2	3
weight	1	2	3	4

We then sort the results of each Recency, Frequency and Engagement column and divide them in four quartiles, giving them the values {1,2,3,4}. The final RFE value is the sum of these values, i.e. the highest valued costumers have an RFE score of 12, while the lowest rated have a score of 3. A sample of the results from the RFE segmentation can be seen in Table 3.

TABLE 3. RFE RESULTS

user_id	merchant_id	R	F	E	RFE Sum
196197	604	1	1	1	3
291521	4268	1	1	1	3
291521	3828	1	1	1	3
...	...	...	...	..	...
254263	4806	4	4	4	12
254263	1267	4	4	4	12
254263	1102	4	4	4	12

5.2. Features

We can divide the extracted features into five groups: User, Merchant, User-Merchant, Category and Brand.

5.2.1. User. In the user group, we use the results from the RFE customer segmentation.
We also extract other features:
- the number of each action type that each user performs, for example f_u_act_0 is the feature that holds the number of actions of type 0 a user makes,
- and the number of times (from time_stamp) a user buys an item or adds an item to favorites, e.g. f_u_f_ts is the feature that holds the number of times (ts is time_stamp) a user adds an item to favourites.
In total we have six features in the user group: f_u_act_0, f_u_act_1, f_u_act_2, f_u_act_3, f_u_f_ts and f_u_p_ts.

5.2.2. Merchant. In the merchant group, we have extracted twenty-five different features, an overview can be found on Table 4

TABLE 4. FEATURES PER MERCHANT

f_m_user	The number of users
f_m_item	The number of items
f_m_cat	The number of categories
f_m_brand	The number of brands
f_m_act_0-3	The number of action type 0-3
f_m_user_p>2	The number of users with more than two purchases
f_m_tmm_p_max	Maximum number of sales per month
f_m_tmm_p_min	Minimum number of sales per month
f_m_tmm_p_std	Standard deviation of number of sales per month
f_m_tmm_p_mean	Average number of sales per month
g_0 - g_2	The number of users per gender
a_0 - a_8	The number of users per age range

5.2.3. User-Merchant. For the user-merchant group we have seven different extracted features, that are the following (all of the descriptions are per costumer per merchant): f_um_item_p (number of items), f_um_cat_p (number of categories), f_um_brand_p (number of brands), f_um_act_0-3 (number of action types: 0-3).

5.2.4. Category. In the category group we find five categories: f_c_action (feature containing all actions per category), f_c_user (users per category), f_c_merchant (merchants per category), f_c_item (items per category) and f_c_brand (brands per category).

5.2.5. Brand. In the brand group, we also find five categories, which are the same as in the category group, but with a category instead of a brand feature: f_b_action, f_b_user, f_b_merchant, f_item and f_b_cat (categories per brand).

6. Feature Analysis

In this section we explain the analysis we conducted on the extracted features.

6.1. Correlation and Rank

We extracted 49 features from a total of 5 information combinations, and calculated feature correlation and importance maps.

In the correlation Figure 3, the closer to the red, the higher the correlation between features. It can be seen that only about 10 types of features have low correlations, which is also proved by PCA results in section 6.2.2.

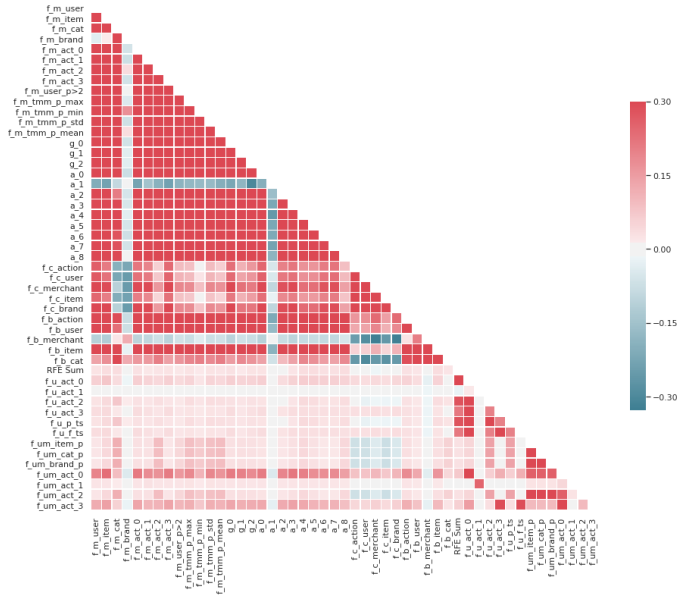

Figure 3. Feature Correlation

According to the introduction of Feature Importance and Feature Selection of Jason[6], we use Decision-Tree-based gradient boost ideas to extract feature importance, as shown in Figure 4. The score of each feature indicates how useful or valuable it is for building an enhanced decision tree. The more a feature is used in making a key decision, the more important it is.

Our ranking of feature importance shows that there are 12 features with an importance score of 20 or greater. The number of action_type 0, or click action, for each user-merchant is the most important, and there is no clear rule of which type of feature is more important than others.

From the correlation and importance analysis, we believe that feature is the key to affect the prediction accuracy, and extracting as many features as possible could mine information better from user actions and improve the prediction result.

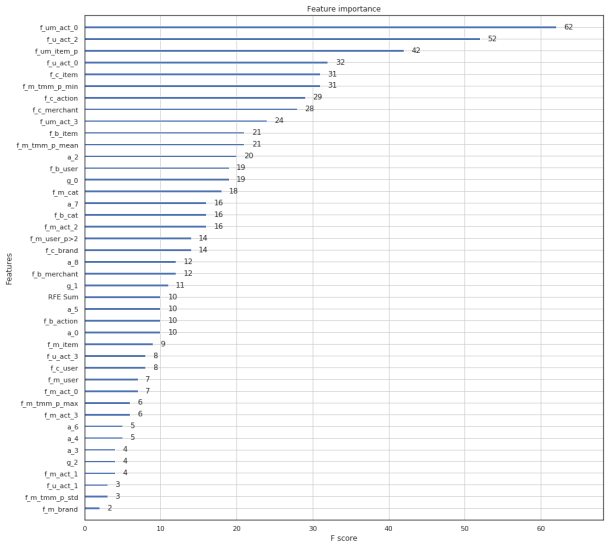

Figure 4. Feature Importance

6.2. Dimensionality Reduction

The project implements three different ways to reduce the high dimension, non-linear based AutoEncoder, linear based PCA and clustering. The results are saved separately for training comparison needs.

6.2.1. AutoEncoder. AutoEncoder is a non-linear, artificial neural network based algorithm[4]. It uses unsupervised learning to encode and decode data. In image analysis, AutoEncoder is usually used for image compression. The compressed image retains the important original image information while removing noise points and reducing the number of color bits. Similarly, in data processing, AutoEncoder can remove outlier data as well as reducing the dimension.

We perform a three-layer AutoEncoder. The original input features number is 49; therefore, in encoder processing, the settings of neurons in each layer are 40, 20, and 10 in order. We perform the decoding layer by layer, forcing AutoEncoder to select the most useful information and reduce the noise of the data. Finally, the decoded result is encoded again by 10 neurons to obtain new features for dimensionality reduction.

6.2.2. PCA. (Principal component analysis) is a dimensionality reduction method. If the dataset consists of n dimensions, PCA looks for k n-dimensional orthogonal vectors to represent the data, where $k \leq n$. In other words, it projects the original data onto a much smaller space, which results in dimensionality reduction. To perform this, the data is first normalized. Then k orthonormal vectors, referred to as *principal components*, are computed. These vectors are unit vectors that point in a direction perpendicular to the others and the normalized data can be represented as a linear combination of the principal components. The principal components are then sorted and we select the most important components that cover most of the variance of the data. Compared to AutoEncoder, PCA is a linear algorithm[3].

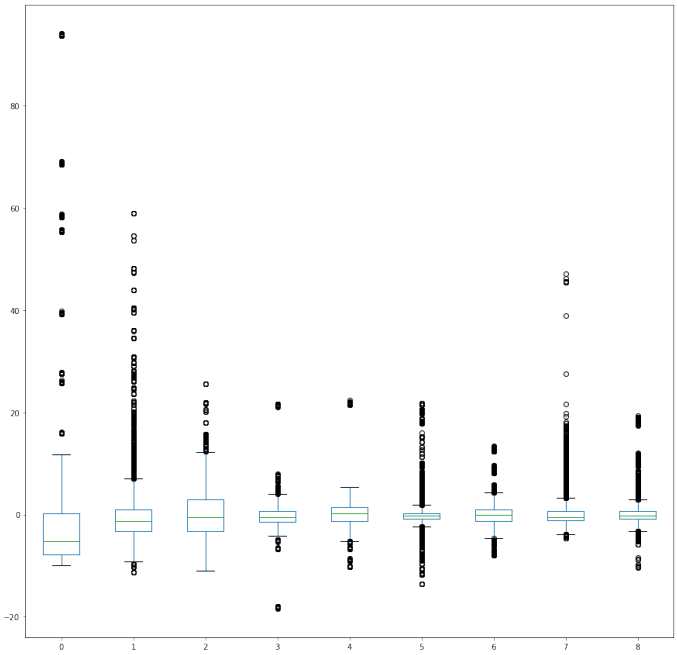

Figure 5. PCA Features Box Plot

We performed PCA on our training feature data. We chose the first 8 principal components, which cover 95% of the variance of the data. Then we made a box plot of the principal components to observe the outliers. As we can see on Figure 5, components 1, 5, 7 and 8 have a large amount of data points outside of the whiskers, some of which can been seen as a cluster. Therefore, it is not a good idea to remove all the outliers, as we might lose some important information. In this case, we only slightly trimmed the outliers.

6.2.3. Kmeans. K-Means is a centroid-based clustering technique. We use the centroids of each cluster to represent the corresponding cluster. The centroid of a cluster is its center point. To compute the centroids, the algorithm first initializes with k randomly selected objects, where k is the number of centroids we need. For the remaining objects, it assigns each of them to the closest cluster based on the

Euclidean distance between the object and the centroids. Then it computes the new centroids of each cluster and reassigns the objects to the new centroids until there is no update or the algorithm reaches the maximum number of iterations [3]. In order to find the best parameter k, we applied the elbow method. This computes the distortion score of the given cluster number k, which is the sum of squared distances from each object to its assigned centroid. The distortion score goes down as k increases. Thus the line chart of distortion score and k resembles an arm and the point of inflection on the curve ('the elbow') indicates the best k value for the model.

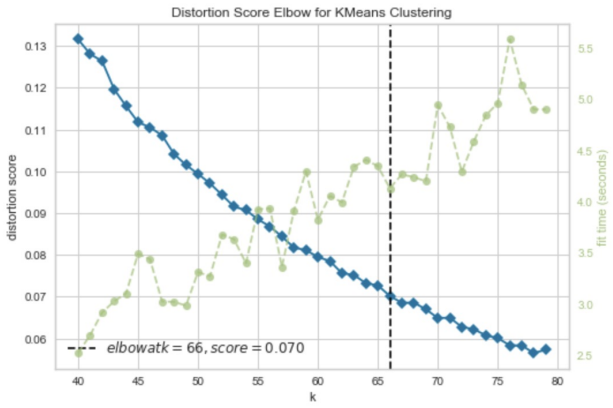

Figure 6. User-merchant Kmeans Elbow

K-Means showed a higher running time performance than other clustering methods. Therefore, we divided the training features into 5 feature groups. Figure 6 shows the distortion score and fit time of different k values for user-merchant features. We can see that it is not easy to tell the 'elbow' point with the naked eye, but the computed k value is 66.

7. Model Training

In this section we talk about the different algorithms we have used for model training. The training features were split into random train and test subsets with a ratio of 80% - 20%.

7.1. Support Vector Machine(SVM)

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection [8].
The algorithm outputs an optimal hyperplane which categorizes new examples. In two dimentional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.
SVM uses a nonlinear mapping to transform the original training data into a higher dimension. Within this new dimension, it searches for the linear optimal separating hyperplane[3].

We use the GridSearchCV, an exhaustive search over specified parameter values for an estimator, where the estimator is the model we define with svm.SVC (Support Vector Classification). The parameters can be f.eks. kernel, C and gamma, we tuned the kernel parameters, using 'linear' and 'rbf' (Radial Basis Function) as kernels.
We then use the three dimentionality reduced versions of the data (AutoEncoder, PCA and KMeans, see chapter 6.2) to fit the model and predict the ROC AUC score (more about the ROC AUC score in chapter 7.4).

7.2. Multi-layer Perceptron

Multi-layer Perceptron is a feedforward artificial neural network. Perceptron is a supervised linear classification algorithm. It takes weighted inputs and maps them to an output by a threshold function. Different from linear perceptron, a multi-layer perceptron has several layers of neurons and a non-linear activation function. Thus it is able to perform classification task on data that is not linearly separable.

We also applied GridSearch to search for the optimal parameters. There are 4 different activation functions for the hidden layer: *indentity*, *logistic*, *tanh* and *relu*. In the recent years, ReLU (rectified linear unit) has been more frequently used among the others for deep learning and it was also the function we got through GridSearch. ReLU returns f(x) = max(0, x). There are 3 different solvers for weight optimization: *lbfgs*, *sgd* and *adam*. We tried two of them and got *adam* as our solver through GridSearch, as it usually works well on relatively large datasets.

7.3. Decision Tree

Decision tree is a supervised classification method. A decision tree is a flowchart-like tree structure, where each internal node (non-leaf node) denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (or terminal node) holds a class label [3]. The decision tree method performs the classification task by making decisions about one attribute at a time. We start from the root node and proceed to a leaf node to get class label.

7.4. Evaluation

The positive and negative proportions of the training samples are very imbalanced, with only 6% of the positive samples. So, we are using a confusion matrix to describe the accuracy, due to the default classification boundary being 0.5 in [0,1], then about 94% will be classified as a negative sample and 0% positive detected. Although the confusion matrix shows high accuracy, the precision difference between positive and negative samples is too big, which means that the model is terrible.

Therefore, we choose to use the ROC (Receiver Operating Characteristics) AUC (Area Under The Curve) curve. ROC-AUC is also used to evaluate a binary classifier. Sarang[7] write in a ROC-AUC introduction that, *" ROC is a probability curve and AUC represents degree or measure of separability. It tells how much model is capable of distinguishing between classes. Higher the AUC, better the model is."* The evaluation results are shown in Table5.

TABLE 5. ROC AUC SCORE OF DIFFERENT MODEL AND TRAIN DATA

	Origin	**AE**	**PCA**	**Kmeans**
SVM	pass	0.530	0.517	0.478
MLPC	0.64	0.5	0.652	0.56
DT	0.515	0.517	0.503	0.505
GBDT	0.657	0.619	0.624	0.648

We used multiple models and datasets for evaluation comparison. We found using the results of feature extraction directly is generally more accurate than performing dimension reduction. Here, because PCA chose to obtain more than 95% of the information, the results are also good. The Kmeans clustering results only perform well in the Gradient Boosting algorithm. Using Kmeans in high-dimensional large data sets may cause an unstable clustering, so it affects the prediction.

The best score of the Multi-layer Perceptron neural network is 0.65, and the effect of Decision Tree is the worst. SVM also has weak results due to uneven samples and dimension reduced data. Gradient Boosting is an improved algorithm that combines weak models, typically Decision Tree, while taking into account the gradient calculation of the cost function. We can see that the Gradient Boosting score is also good, up to 0.65. Alibaba's official baseline is 0.7, and there is no group beyond the baseline in the top 400 teams.

8. Reflection and Application

Regarding our choices for the used algorithms and their implementations, we are quite confident that most of them are relevant for this project (more about the quality of the obtained results in the evaluation, section 7.4). Feature extraction represents one of the indispensable tools used for this case, as a significant number of new features were revealed. Our 0.65 score of the Multi-layer Perceptron could slightly be improved, especially if we want better chances at winning this competition. However, we are pleased with the quality of our results and we strongly believe that this project is pertinent as a submission for this course.

those are ... are...something we can improve.... Feel free to ignore... But remember them in exam: "How to improve your model, since it hasn't get the baseline?"

As stated in our case description, the main application of this project is to mine and analyze the provided buyers' data, in order to determine which of them is susceptible to converted into loyal customers (or "repeat buyers") for different sellers. Knowing this sort of information is a crucial aspect for any merchant in the era of digital marketing, as better targeting could significantly improve the return on investment for the capital spent in online advertising.

An important aspect worth to consider when mining data collected from regular users is privacy. The data given for this public competition does not contain any sensitive personal information, such as names, addresses, or payment information, as every user is referenced by a numerical identifier (user_id). Moreover, the age of the users has been normalized into different groups by the data provider and some of the entries have unknowns values for the age range and/or gender. Based on these considerations, all the concerns regarding privacy issues are left to the owner of these data points, as they are likely to posses personal information linked to every user_id. Nonetheless, there are differences regarding the regulations imposed by different authorities, which depend on region. As this data comes from China, its owner might be constrained by different law procedures than what we know as GDPR (General Data Protection Regulation) in the European Union, such as the extent to which the regulations are legally binding or who has the true "ownership" of the user's data. [9]

References

[1] Alibaba Cloud TIANCHI, Competitions, Repeat Buyers Prediction-Challenge the Baseline, https://tianchi.aliyun.com/competition/entrance/231576/information

[2] Pyhon, pandas 0.25.3 documentation, DataFrame https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html

[3] Han, J., Pei, J., & Kamber, M. (2011). Data mining: concepts and techniques. Elsevier.

[4] Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. AIChE journal, 37(2), 233-243.

[5] Chandan Rajah, Engineering Manager, London Business School. *Strategic Customer Segmentation - A Universal Approach* (March 30, 2016) https://www.linkedin.com/pulse/strategic-customer-segmentation-universal-approach-chandan-rajah

[6] Jason Brownlee (2016), Feature Importance and Feature Selection With XGBoost in Python, Ref: https://machinelearningmastery.com/feature-importance-and-feature-selection-with-xgboost-in-python/

[7] Sarang Narkhede (2018), Understanding AUC - ROC Curve, Ref: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

[8] *Scikit-learn: Machine Learning in Python*, Pedregosa, F. and Varoquaux, G. et.al, Journal of Machine Learning Research, v.12, p.2825-2830, (2011)

[9] Michael Gentle (2018), China's data-privacy law vs. GDPR, Ref: https://medium.com/the-balance-of-privacy/chinas-data-privacy-law-vs-gdpr-566fde8c213c