Coursework:

Churn Prediction for FoodCorp

Course: Machine Learning and Advanced Analytics

Student ID: 20171924

Academic year: 2019 / 20

A. Executive Summary

This report is mainly providing churn rate predicting system of FoodCorp as they considered customer churn issue is a significant concern across their stores. Since consulting company provided a methodology for determining a churn definition and some analysis, I conducted interpreting the initial work and decided to use day 33 as the definition of churn. Individually, when a customer visits a store after 33 days, then it considered as a churner. For features, I finalized with '*total values*', '*total quantity*', '*average between visits'* and 11 number of periods of the sum of values. Since the data type is temporal data, I chose reference day as 576 which is 17th October 2019, tumbling window size and output window size as 33 and window aggregation function and output aggregation function is both sums.

After feature engineering such as balancing output feature of training dataset and standardizing of traditional and temporal data, I mainly used f1 score, ROC curve and AUC score to decide the final prediction model, which is XGBoost algorithm. To understand more details about the statistically significant relationship between input features and the output feature, several feature importance methods are used. As a result, feature *'f1'* was the least important feature; thus, I decided to delete the *'f1'* feature in the final prediction system.

Predicting the whole dataset with the final prediction system, the number of churners is 486 (51.4%), while non-churners are 460 (48.6%). Two groups show a significant difference in terms of total values and total quantity as well as the sum of values periods, especially during '*f2'*, '*f3'* and '*f6'*.

Some insight and recommendations are given:

1. Churners, bounce back marketing by providing a discount coupon to visit more frequently.

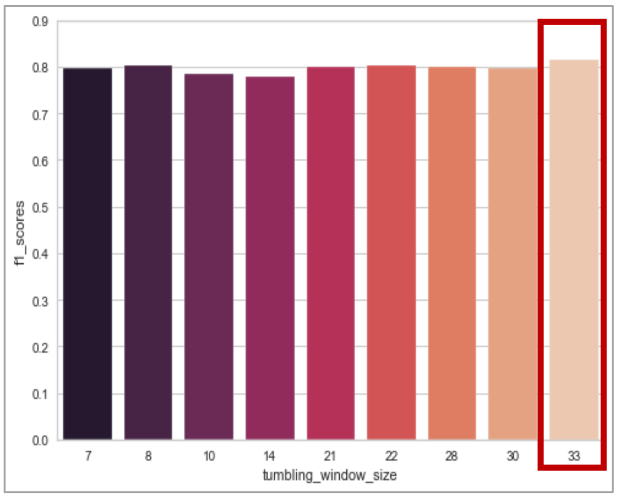
2. Churners, bounce back with upselling, encourage churners to re-visit and spend more.

3. Non-churners, loyalty program by offering unique goods and rewards.

B. Churn rates and churn definition

The churn prediction system predicted the current churn level as 51.4 per cent. The total number of the customer is 3,852 in the database, however, when predicting customer's churn rate, only active customers are included, which is 946 number of customers. The number of churners is 486 (51.4%), while non-churners are 460 (48.6%). The specific amount of churn definition requiring the number of active customer and churn customers to calculate. Since consulting company provided a methodology for determining a churn definition and some analysis, I conducted interpreting the initial work and decided to use day 33 as the definition of churn. Churn definition as 33 days can be construed as 59.88 per cent of customers visit less than this in median and Foodcorp can expect to target 19.03 per cent of active customers with a perfect classifier. With my prediction system, 24.5per cent is considered as an active customer, which 5.5 per cent is different.

Moreover, since the data type is temporal data, I chose reference day as 576 which is 17th October 2019, tumbling window size and output window size as 33 and window aggregation function and output aggregation function is both sums. Active customers can be calculated by people who purchased between reference day and 33 days before reference day, which is 17th October to 19th September 2019. In this model, I assume the number of churn definition should be same as tumbling window size, thus two features have the same number. With the final prediction system, I tried to check the optimal churn definition by comparing f1 score, and it founded out that 33 days is the optimal churn definition with my prediction system.



C. A technical part on the churn prediction system

C-1. Importing data from the database

The primary data type this coursework dealing is temporal data, which has time information. Since we need to import data from the database using SQL code in Jupyter Notebook, the value of reference day, tumbling window size, output window size and aggregation function of window and output have to be filled. From the consulting report, churn is defined as 33 days. Thus, the number of tumbling window size as well as the output window size is 33. Therefore, reference day is 17th October 2019, which is 576. All the number of a date in transformed into integer number from 1 to 609. Lastly, both window and an output aggregation function are using the sum function.

For features, I finally used *'total values'*, *'total quantity', 'average between visits'* and 11 periods of the sum of 33 day's values. Most intuitive and straightforward feature of understanding customer's behaviour would be their total spending and the total number of purchased units. Moreover, the feature '*avg_between'* is included to see each customer's frequency of visiting store. Lastly, for lagged features, there were three choices: the sum of values, the number of visits per week and average between visits. However, the sum of values was more interpretable than the number of visits per week in terms of judging churn definition of each customer. Average between visits couldn't be applied into SQL code, because of an error that doesn't match the length of rows with each customer.

C-2. Pre-processing

After importing data from the database, pre-processing is essential to continuing modelling. There are three steps: 1. changing output features into binary, 2. balancing the training dataset using SMOTE, 3. standardizing traditional and temporal data. When importing data using SQL code, I defined an output feature as the sum of values in order to transform into binary, churner or non-churner. In this coursework, I assigned *'0'* as churn and '*1*' as non-churn. Thus, when the output feature is more than zero, it changes to ‘*1*’, and unchanged values remain as ‘*0*’.

Next, balancing the number of each binary output features in the training dataset is necessary. To find the optimal SMOTE methods, I tried nine types of SMOTE methods including SMOTE, Borderline SMOTE, SVM SMOTE, ADASYN, Kmeans SMOTE, RandomOverSampler, SMOTENC, SMOTEENN and SMOTETOMEK with K-nearest neighbors algorithm and calculated f1 score. As a result, I decided to use SMOTE with default K, because the result of f1 score is similar and best fit for this temporal data.

The last step of pre-processing is standardization. In this part, it requires to divide into two groups based on the type of data and applies different methods of standardization. The first group is traditional data type features, *‘total values’, ‘total quantity’* and *‘avg_between’*. Since these three features show right skewness starting from value zero, log1p transformation is required to make the data more normal distribution. Moreover, rescaling to remove the units StandardScaler of sklearn is used. The second group is temporal data type features from ‘*f1’* to ‘*f11’*. Instead of using StandardScaler, each value needs to subtracted by the total sum of all the values and divided by the standard deviation.

C-3. Evaluation

In evaluation part, there are three steps: Step 1. finding default models that have higher f1 score than baseline model, step 2. tuning the hyper-parameters of a model that found in the previous step by Randomized Search, step 3. finding the best prediction model with five holdout sets. All these steps used training dataset and validation dataset.

1) Finding a model that has higher f1 score than baseline model’s with two holdout sets

	Model	F1 score
0	**LogisticRegression(solver = 'liblinear')**	**0.8**
1	KNeighborsClassifier()	0.713
2	LinearSVC(C=1, loss='hinge')	0.352
3	LinearSVC()	0.352
4	SVC(kernel='rbf', gamma=5, C=1)	0.713
5	SVC()	0.713
6	GaussianProcessClassifier(1.0 * RBF(1.0))	0
7	GaussianProcessClassifier()	0
8	DecisionTreeClassifier(max_depth=5)	0.691
9	DecisionTreeClassifier()	0.657
10	RandomForestClassifier(max_depth=5, n_estimators=10)	0.714
11	RandomForestClassifier()	0.734
12	GaussianNB()	0.713
13	AdaBoostClassifier(n_estimators=100)	0.48
14	**GradientBoostingClassifier(n_estimators=100)**	**0.752**
15	**xgb.XGBClassifier()**	**0.793**
16	*Baseline model: f1_score(valid[1], valid[0].f1)*	*0.723*
	* All the model’s random state is 42	

Total ten types of models with different hyper-parameters predicted validation set with a training dataset and compared by f1 score. The reason f1 score is used as the leading prediction score is f1 score is the harmonic mean of the precision and recall, the higher f1 score, better precision and recall. For comparison with the baseline model’s f1 score, baseline prediction is made by predicting the validation output feature with a validation input feature ‘*f1’*, which is the closest period to reference day. Since baseline f1 score is 0.723, three models have higher f1 score, logistic regression with ‘liblinear’ solver, gradient boosting classifier and XGBoosting classifier.

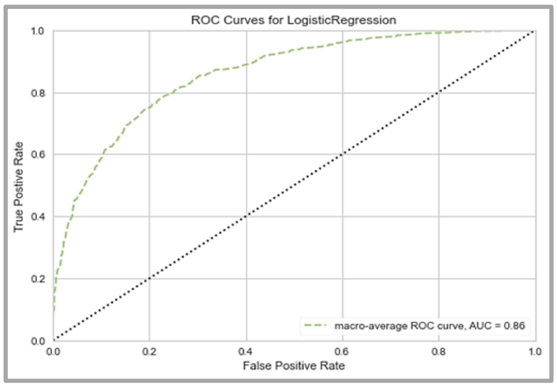
2) Tuning the hyper-parameter with Randomized Search CV

By comparing f1 scores of 16 models with baseline prediction model, Logistic Regression, Gradient Boosting and XGBoosting shows higher f1 score than baseline's f1 score. Now, tuning each of models is a demand to find optimized hyper-parameter. In this step, train dataset and validation dataset is used without holdout sets.

a. Logistic Regression

Model	Logistic Regression	
Basemodel f1		0.723
Before tune f1		0.800
After Randomized Search CV		
Train accuracy		0.820
Validation accuracy		0.770
Validation f1 score		0.786
Confusion Martix	y_pred Yes	y_pred No
y_true Yes	352	121
y_true No	71	292

After Randomized Search CV, the level of f1 score decreased by 0.014, however, this amount of decrement can be interpreted as solving the problem of overfitting. By regulating 'C' hyper-parameter as 0.00233 and 'penalty' as l2, the model's decision boundary complexity became decreased. In terms of the confusion matrix, the total rate of False Negative (FN) and False Positive (FP) is 22.97 per cent.

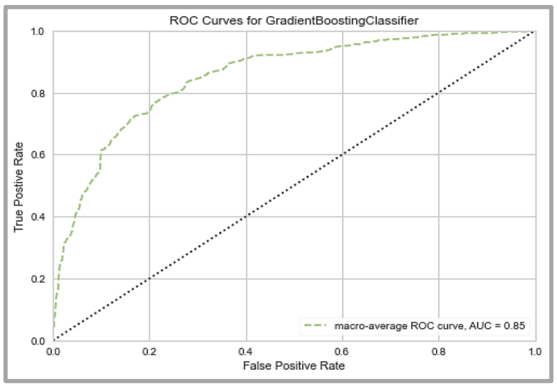


ROC curve of Logistic Regression shows the quite rounded shape, and it's AUC score is 0.774.

b. Gradient Boosting Machine

Model	Gradient Boosting	
Basemodel f1		0.723
Before tune f1		0.752
After Randomized Search CV		
Train accuracy		0.817
Validation accuracy		0.744
Validation f1 score		0.727
Confusion Martix	y_pred Yes	y_pred No
y_true Yes	285	188
y_true No	26	337

The decrement of f1 score after tuning hyper-parameter case also shows in Gradient Boosting Machine by 0.025. However, most importantly, the GBM's f1 score is lower than Logistic Regression's f1 score by 0.059. The gap between train and validation accuracy is similar to Logistic Regression. With GBM model, Randomized Search CV is used twice. Firstly, it used with a large range of hyper-parameter. Secondly, a narrow range of hyper-parameter with first Randomized Search CV's result. As a result, the final tuned hyper-parameter is following: subsample = 0.384, n estimators = 296, min samples split = 6, min samples leaf = 36, max depth = 10 and learning_rate = 0.077. In terms of the confusion matrix, the total rate of False Negative (FN) and False Positive (FP) is 25.6 per cent.

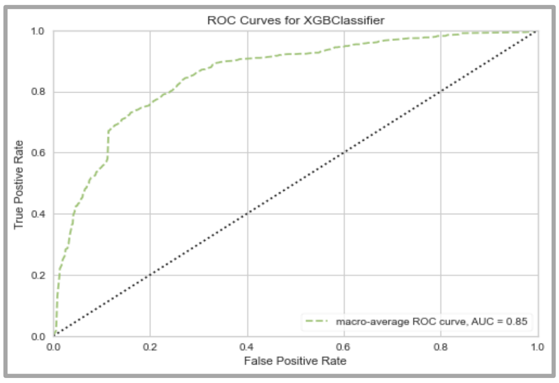


GBM's ROC curve show less rounded shape compare to Logistic Regression, and it's AUC score is 0.765.

c. XGBoost

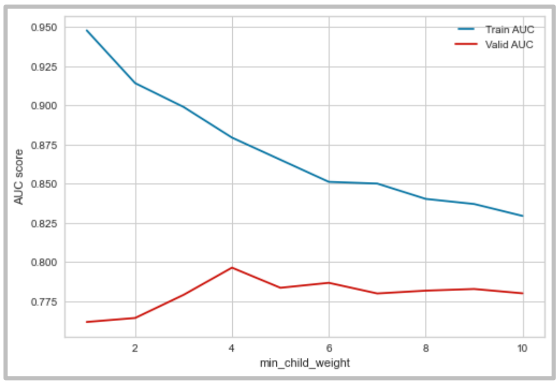
Model	XGBoost Classifier	
Basemodel f1		0.723
Before tune f1		0.793
After Randomized Search CV		
Train accuracy		0.840
Validation accuracy		0.786
Validation f1 score		0.811
Confusion Martix	y_pred Yes	y_pred No
y_true Yes	385	88
y_true No	91	272

Last modified model is XGBoost. Even though there is an f1 score decrement after tuning hyper-parameter, the degree of train and validation dataset accuracy and f1 score is highest among the three models. XGBoost model also used Randomized Search CV twice, wide range and narrow range of searching hyper-parameter. As a result, the final hyper-parameter is following: subsample = 0.208, n estimators = 862, min child weight = 8, max depth = 5, learning rate = 0.0171, gamma = 0.297, coal sample by tree = 0.782.



From the ROC curve of XGBoost, AUC score is 0.782, which is the highest among the three models.

Since XGBoost has the highest f1 score, I tried searching hyper-parameter more delicately by changing the value of one specific hyper-parameter while others are remaining as the result of Randomized Search CV. In this case, the AUC score of train and validation is used to choose the optimal value of each hyper-parameter. For instance, searching min child weight, trying the value of min child weight one to ten with other hyper-parameters that founded in Randomized Search CV and comparing train and validation dataset's AUC score.



This graph shows four is the optimal value of min child weight hyper-parameter. Train dataset's AUC scores continually decreasing as the number of value increases, on the other hands, validation dataset's AUC score increases until the value four and decreases over four. Total seven hyper-parameter continued train and validation AUC score comparison and the result would be following: subsample= 0.2, n estimators= 175, min child weight= 4, max depth= 3, learning rate= 0.5, gamma= 0.025, coal sample bytree= 0.5.

Model	XGBoost Classifier
After AUC score comparison	
Train accuracy	0.867
Validation accuracy	0.778
Validation f1 score	0.801

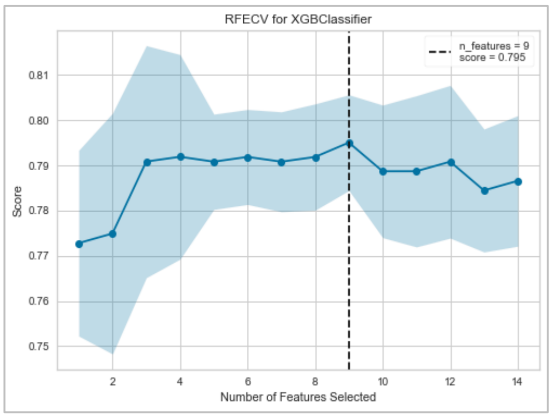
This is the result of using XGBoost with above hyper-parameters. Compare to the XGBoost model result using Randomized Search CV, the gap between train and validation accuracy is more significant, this could interpret as overfitting. Moreover, the last model's f1 score is lower than the previous model. In this regards, XGBoost that tuned hyper-parameter by Randomized Search CV shows the highest f1 score, which is the most critical score in this problem.

3) Finding the best model by five holdout sets with train and validation dataset

Model	Train accuracy	Validation accuracy	F1 score
Logistic Regreesion	0.794	0.781	0.792
Gradient Boosting	0.956	0.713	0.663
XGBoost	0.831	0.789	0.803

With train and validation dataset that pre-processed by SMOTE and standardization, I predicted validation set with three models with five holdout sets. As a result, XGBoost shows the highest f1 score, thus XGBoost classifier is the best prediction model in this problem.

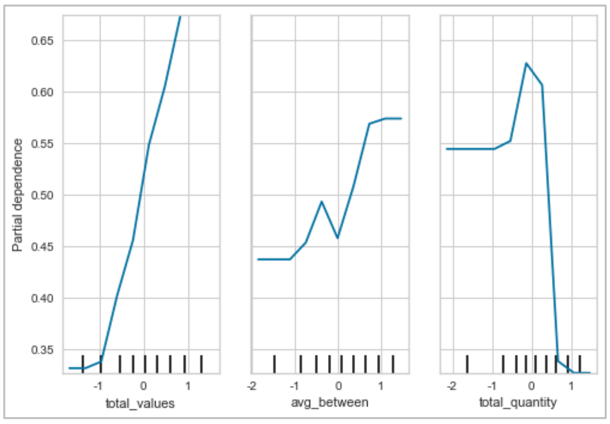
C-4. Feature importance and selection



Feature importance and selection is processed in XGBoost model with train and validation dataset without holdout sets. For feature importance and selection, Recursive feature elimination cross-validation (RFECV) and Recursive feature elimination (RFE) with Permutation Importance are used. From RFECV, it investigated that the nine number of feature is the best for this model. Furthermore, based on RFE that fits a XGBoost model and removes the least important features which embedded permutation importance, it founded the rank of features.

RFE Perm CV XGB	
Feature	Importance
total_values	0.0357
avg_between	0.0338
total_quantity	0.0095
f2	0.0013
f3	0.0004
f6	0.0004
f9	0.0000
f7	0.0000
f4	0.0000
f11	0.0000
f10	0.0000
f5	-0.0017
f1	-0.0025
f8	-0.0053

The graph beside shows the degree of the importance of each feature in this model. Feature '*total values*', '*avg_between'* and '*total quantity'* indicates the highest values, which means the most crucial feature in this model. On the other hands, from feature *'f11'* to *'f8'* is the least important feature in this model and will be deleted. To increase the performance of the model, feature selection is essential. After deleting five features, f1 score of XGBoost decreased by 0.012 which is not much difference then before.



To further understand the model, partial plot dependence is used, and it shows how a model's prediction depends on a single input. In this graph, the three most important feature *'total values', 'avg between'* and *'total quantity'* shows the relationship with the target variable, churn or non-churn. Since *'1'* is non-churn and *'0'* is churn, over -1 value of total values getting close to non-churn. In terms of the average between visits, more than 0 getting close to non-churn. Lastly, customers who buy more than 0.4 of total quantity, is possibly churner.

C-5. Summary

Our final prediction model is XGBoost classifier with hyper-parameter that founded from Randomized Search CV. After RFE with permutation importance, it discovered that feature *'total values'* is the essential feature and feature *'f8'* as the least important feature. From RFECV, I decided to maintain nine features, thus I eliminated feature *'f8', 'f1', 'f5', 'f10', 'f11'* in the whole dataset and obtained 0.764 of f1 scores with 0.826 train set accuracy, and 0.738 test set accuracy. This result brought out without holdout sets.

D. Insight part 1

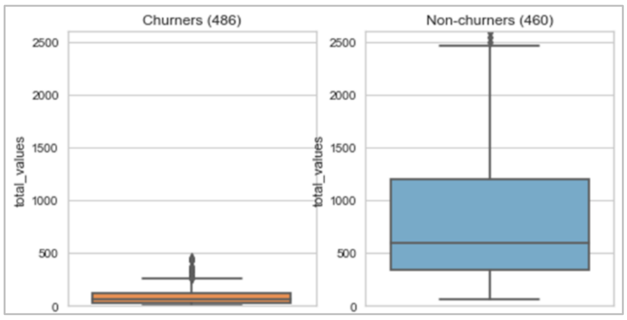
In insight part 1, a summary of the marketing strategy in both groups is mainly debated. For churners, bounce back offering marketing will be recommending, on the other hands, loyalty programs will be suitable for non-churners. In the churn prediction model, there are four kinds of features, total spending, spending in periods, entire purchased unit and average between visits. Marketing strategy for both groups is based on these four features. In this report, finding the churn rate and identifying churners is the main goal. The churn rate is 48.6 per cent, and they show distinguished behaviours comparing to non-churners. Churner's total spending and the total amount of purchasing unit is almost ten times lower than non-churners. A notable feature is ten periods of total expenditure. In all periods, 50 per cent of churner doesn't spend at all. In both groups, all periods of total spending's standard deviation are similar, thus both groups spending habit is constant. Lastly, churner visit stores in 51 days on average, contrarily, non-churner visits stores in 12 days on average. In this regards, both group's marketing strategy is mainly focusing on increasing the frequency of visiting stores with a differentiated approach.

Bounce back offers for churners is generating a re-order or further visit and can cause additional sales to the FoodCorp. Since churner's the degree of frequency is low, by marketing strategy increasing the frequent rate is essential. More specifically, by providing discount coupons to churners right after they purchased is to entice them to 'bounce back' to the store soon. Considering churners spending behaviour, small total spending and small purchasing units, a fairly aggressive offer that 15 per cent to 20 per cent discount would be suitable. However, the expiration date must be specified. With limited validity may encourage churners to visit stores less than 51 days on average. If the effect of 'Bounce back' marketing strategy deficient, adding an upselling strategy might help. Bounce back offer with upselling might interest them to re-visit stores, for instance, discount 15 per cent off your next purchase and receive a buy one get-one-free on any of product less than £10. With bounce-back offer and upselling, we can expect to encourage churners to re-visit less than 51 days on average.

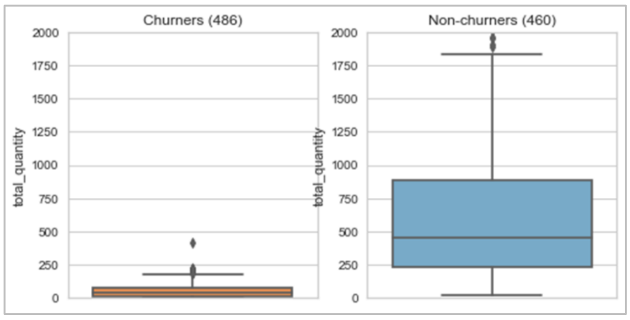
On top of that, maintaining the non-churner rate of 51.4 per cent is as vital as turning churners into a loyal customer, and it is less cost and high-efficiency strategy for the FoodCorp. A loyalty program that provides special members-only offers and rewards would keep the rate of non-churner and hopefully increase the number of non-churners. Since churners visit stores in 12 days on average and spend approximately £20 in each visit, offering exclusive members-only product delivers the sense of caring and attention to churners. Moreover, rewards to non-churners depending on the level of loyalty is required to compensate their love and affection to the FoodCorp. The rewards might be a gift certificate that they can use anytime and also can be passed over to others. The reason the discount coupon is not used for rewards is that they might feel discriminated when the rate of discount is different. A loyalty program that offering unique goods and gift card might encourage churners to become loyal customers and visit more often to spend more budgets.

E. Insight part 2

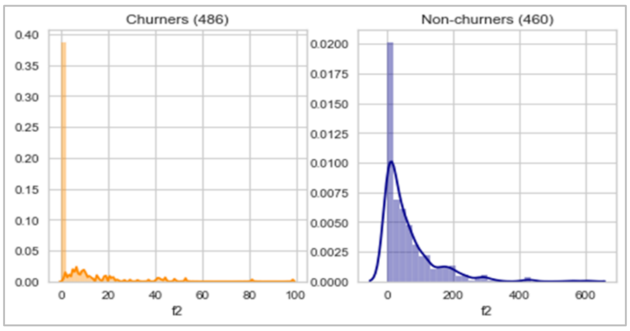
In insight part 2, more technical detailing about how insight part 1 were derived based on churned and non-churned customers' pen portraits is discussed. To predict customers whether they are churned or non-churned, XGBoost embedded probability is used. When the probability is over 0.5, considered them as non-churned. As a result, the total number of churner is 460 out of 946 active customers, thus the churn rate is 48.6 per cent. Pen portraits of churners and non-churners are based on features that considered as essential features from XGBoost embedded feature importance.



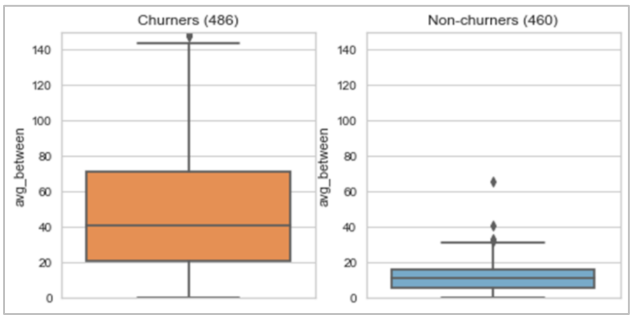
First comparison feature is *'total values'*, which is the most important feature. As it can see graph beside, the characteristic of the two groups is significant. Churners median spending is 58.68, while the maximum spending is 448.52. On the other hands, non-churners' median spending is 599.29, which is nearly ten times larger than churner's average expenditure.



Next comparison feature is *'total quantity'*, which is the second most important feature from XGBoost feature importance. As similar as 'total values' feature, it shows significant differences between churned and non-churned group. Churners median purchasing unit is 35.5, while non-churners are 451.50, which is more than 12 times.



In terms of total ten periods of the sum of values from feature *'f2'* to *'f11'*, feature *'f2'*, *'f3'* and *'f'6'* have become the most important feature in XGBoost classifier model. This bar chart besides demonstrates the distribution of feature *'f2'*, which is the total sum of values from 12th October (510 days) to 14th October 2019 (543 days). Both distributions are highly right skewness, and it contains a large number of zero values. This unique characteristic shows in all period of feature *'f2'* to *'f11'*. More specifically, churners' median spending in *'f2'* period is zero, however, non-churners spend 35.69 in the median. Interestingly, 50 per cent of churners didn't spend at all in whole periods. On the other hands, more than 75 per cent of non-churners purchased during most of the periods.



To understand customer's behaviours in terms of frequency, *'average between visits'* feature graph is required. The boxplot besides shows how frequent churners and non-churners visit stores. Churners visits stores 41days basis in the median, while non-churns visit stores 11days basis in the median. Thus, non-churners purchase nearly three times more frequently than churners.