Table of Contents

Directions to run the code 2

Install Packages 2

Load Data 2

Some important pre-processing steps done 2

Feature Engineering (very important part of exercise) 2

One-Hot Encoding of Nominal Variables 2

Feature Selection 3

Train-Test split of Dataset 3

Class imbalance problem 3

Modelling 3

Prediction Results 3

Directions to run the code

User can run the main coding file named customer_return_main_code.ipynb in Anaconda or any other environment. You should have python ML packages installed in your system as mentioned in install packages. You should have datafiles in data folder (as I kept them in my repository). You can open Jupyter Notebook and execute cell by cell.

Install Packages

Following packages must be already installed into your system/Anaconda environment:

1. numpy
2. pandas
3. matplotlib
4. datetime
5. itertools
6. xgboost
7. sklearn
8. mlxtend

Load Data

Data files (both order data and labelled data) must be present in data folder in csv format. I have already done that so ideally you don't have to do anything.

Some important pre-processing steps done

1. Missing values imputation
2. Outlier detection and removal
3. Deciding time range of dataset
    a. Time range of dataset is taken from 01/03/2015 - 27/02/2021 based on significant number of orders. Starting date of data is 17/05/2012 but in duration 17/05/2012 – 01/03/2015, we didn't receive significant number of orders.

Feature Engineering (very important part of exercise)

I have rolled up the data to customer level and created several features which are very important to describe our customer behaviour or target variable.

Features created

1. Total number of orders by each customers (Frequency (F))
2. Total Order Value of Each Customer
3. Total Order Value in last 180 days
4. Number of unique restaurants customer ordered from
5. Recency
6. Customer maturity
7. Percentage of total order value spent in 2nd half by customer
8. Maximum Value of Orders
9. Mean Value of Orders
10. Median Value of Orders
11. Customer segmentation into Single Order, Normal, Attrition, At-Risk, Lost
12. New Features as Pairwise Products of Numerical Variables
13. New Features as square of Numerical Variables

One-Hot Encoding of Nominal Variables

I have done 0-1 (one-hot) encoding of nomical categorical variables to use them in prediction model.

Feature Selection

Step-wise selection and variable importance from XGBoost method are implemented to know feature importance. I have taken variables tagged as important by both algorithms

Train-Test split of Dataset

Whole dataset was split into training set and test set in ratio 0.7/0.3

Class imbalance problem

I have performed under-sampling of class 0 (non-returning customer) data as originally dataset is imbalanced (77% non-returning customers and 23% returning customers)

Modelling

Explanation of model development is mentioned below

Model	Why I developed this model?
XGBoost	XGBoost is extreme gradient boosting model. It creates small decision trees (4-8 nodes) and combine them in such a way that n^{th} tree tries to correctly predict errors made by $n\text{-}1^{th}$ tree, thus this ensemble model is very powerful algorithm. I used this model as it is fast, have minimal impact of outliers, very good performance, less prone to overfitting.
Logistic Regression	I created LR model as it's simple, easy to train, takes less time and we can interpret model coefficients as indicator of feature importance
Random Forest	I created RF model as it works well with non-linear data, robust to outliers, good prediction accuracy and lower risk of overfitting
SVM Classifier	I created SVM because when classes are well separated, SVM works really well, but SVM takes lots of time in training, hence I had to comment it out. You can try it you have enough time.
KNN Classifier	I created KNN as it's fast to train as it doesn't have training period as the data itself is a model which will be the reference for future prediction. It's very easy to understand to business management.
Voting Classifier	Voting classifier is a stable classifier with low variance and overfitting. If independent classifiers (which are past of voting classifier) are good predictors and voting classifier really works well.

Prediction Results

models	accuracy	sensitivity	specificity	auc	f1_score
XGBoost	0.778	0.678	0.808	0.743	0.581
Logistic Regression	0.780	0.671	0.812	0.741	0.580
Random Forest	0.720	0.693	0.728	0.711	0.529
KNN Classifier	0.763	0.662	0.793	0.728	0.560
Voting Classifier	0.784	0.670	0.818	0.744	0.585