Machine learning engineer nanodegree program

Capstone project report

1. Problem definition

A key aspect in the marketing strategy of any company is to understand the value and the needs of its customers. Determine which customers the company should focus and how much to spend with promotions, marketing events, campaigns and other related activities is a fundamental factor to its success.

In this context, the problem addressed in this project consists of:

(1) Understand what customers are more valuable to the business.

(2) How much the company should spend with marketing and promotions for each customer.

(3) Which customers should be targeted by marketing campaigns.

To answer all these questions we will be implementing a customer lifetime revenue (LTR) model. The goal of this model is to predict the future revenue for a given customer using its transaction history. Due to our limited dataset, the prediction period for this project will be 90 days, so our objective is the 90 days future revenue for each customer in our base.

2. Solution details

2.1 Models

We are actually going to build two models**: one machine learning model** that will be deployed using SageMaker **and a baseline parametric model**. This second model is a specific statistical model for predicting lifetime value and related problems.

2.2 Data

The data used in this project is from the Acquire Value Shoppers dataset available at kaggle. We will be using only the transaction history since offers are not relevant to this problem.

2.3 Performance evaluation (metric)

There are many metrics to evaluate regression problems and the two most popular are probably the Mean Squared Error (MSE) and Root Mean Squared Error (RMSE). Another interesting metric is the Mean Absolute Percentage Error (MAPE), since this metric is easy to explain and to relate with the business problem in hand. However, for this particular problem we need to consider a few important aspects:

- There can be many customers with zero purchase amount (churn)

- There is a long right tail for the target variable, including many outliers (see next session)

- We are more interested in a relative error than the absolute error

- In general, for marketing decisions, underestimating the LTR is worse than overestimating it

The first point is a problem when using MAPE since the denominator will be zero. Both MSE and RMSE, as well as other metrics like Mean Absolute Error (MAE) doesn't have this problem, but are heavily affected by outliers. Therefore, the best metric considering all these points is the **Root Mean Squared Log Error (RMSLE)** that gives us a relative error without much impact from outliers. It also has the property mentioned in the last point and is easy to calculate.

2.4 Structure

The complete solution is divided in the following notebooks:

1. Exploratory Analysis: Exploratory analysis for the raw transactions and aggregated in time and by customer. This notebook is the focus of the next section.
2. Preprocessing: Preprocessing the data according to the insights from the exploratory analysis. We also split the data in a development sample and an out-of-time sample.

3. Feature Engineering: Extract features by aggregating transactions at the customer level using the PySparkProcessor class from SageMaker. A custom script with the PySpark code is available at ‘spark/feature_eng.py’.
4. Feature Analysis: Check the resulting features and the target distribution with some visualizations and descriptive statistics.
5. Baseline: A baseline for this problem using two statistical models called BG-NBD and Gamma-Gamma model.
6. Model and results: Build a XGBoost model using a custom script available at the ‘xgb/xgb_train.py’ file. The model is trained, deployed and used to batch transform the data. The results are evaluated and compared with the baseline.

3. Data analysis

The transactions dataset has eleven columns as presented here:

	id	chain	dept	category	company	brand	date	productsize	productmeasure	purchasequantity	purchaseamount
0	16802992	18	97	9753	10000	0	20120309	1.0	CT	1	0.00
1	16802992	18	99	9908	102113020	15704	20120309	6.0	OZ	1	2.00
2	16802992	18	99	9908	102113020	15704	20120309	8.0	OZ	1	2.00
3	16802992	18	58	5827	107024777	6176	20120316	16.0	OZ	1	5.99
4	16802992	18	73	7304	104110040	4082	20120323	20.0	CT	1	6.29

Figure 1 – transactions dataset overview

The description for each column is provided above:

id – customer unique id

chain – store chain

dept - An aggregate grouping of the Category (e.g. water)

category - The product category (e.g. sparkling water)

company - An id of the company that sells the item

brand - An id of the brand to which the item belongs

date - The date of purchase

productsize - The amount of the product purchase (e.g. 16 oz of water)

productmeasure - The units of the product purchase (e.g. ounces)

purchasequantity - The number of units purchased

purchaseamount - The dollar amount of the purchase

The only column with missing values is the product measure

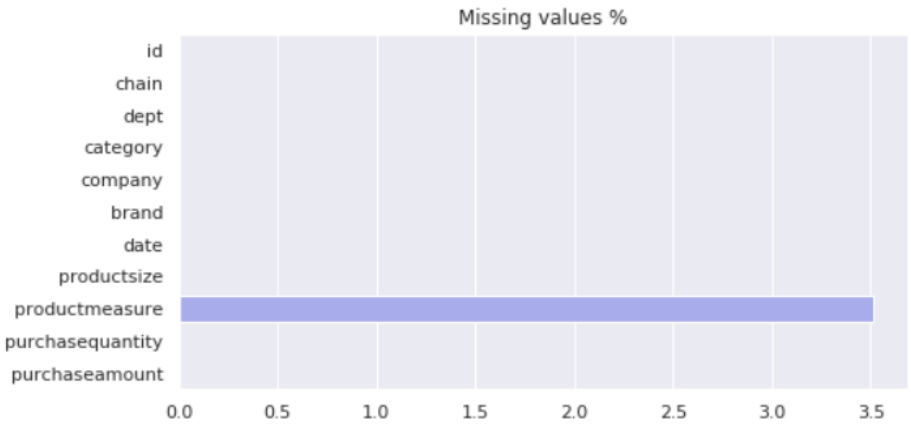

Figure 2 – missing values percentage

During this step, in the *eda* notebook, we perform a univariate and bivariate analysis for each feature. An example of this analysis is presented in Figure 3. A few conclusions from this first step:

- The only column with missing values is productmeasure which is also related to productsize == 0:
- Products with missing measure are mostly from category 0 (missing category) or 9609.
- There are negative transaction amounts, but as described in the data page, these values represent returns or canceled purchases.
- First two digits of the category indicate the dept.
- Purchase amount has a skewed distribution since products may have very different prices.

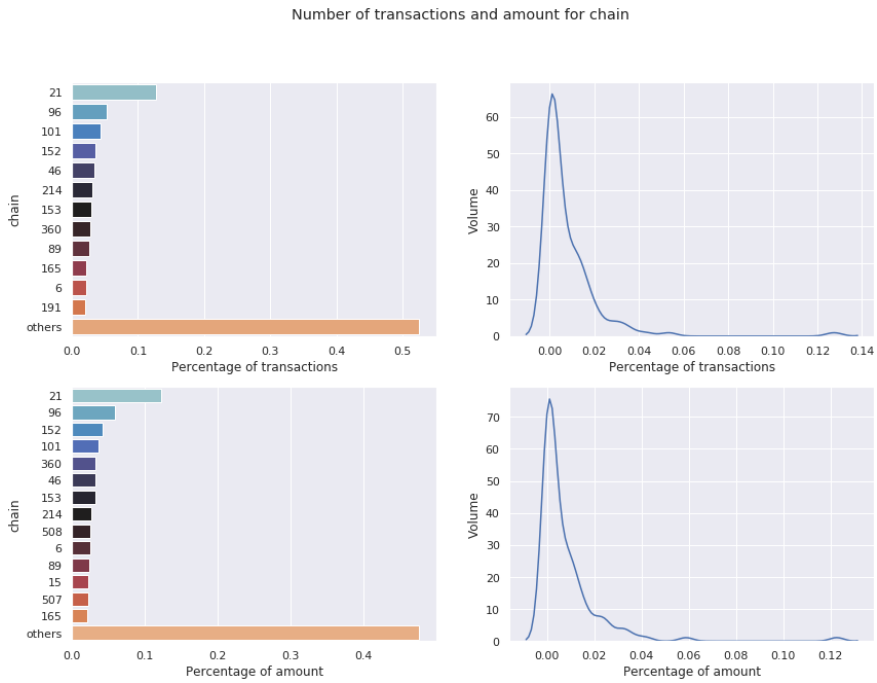

Figure 3 – univariate and bivariate analysis example

The next step was looking at how the transactions are distributed in time. Figure 4 shows the distribution for the number of daily transactions and daily purchase amount, as well as the values over time. We can see some spikes for special dates like Christmas and Thanksgiving.

Figure 4 – number of transaction and amount over time

There is a sharp decline in sales after march 2013 without any explanation. This is probably due to the data collection process. To keep the dataset consistent we will cut the last few months during the preprocessing step.

4. Implementation

4.1 Preprocessing

The first step in the implementation is the preprocessing notebook, which perform the following tasks:

- Cutoff in March 2013 (see Figure 4)

- Removing returns (transactions with negative amount)

- Remove some columns that won't be used

- Splitting the data in development (training) and out-of-time (test) samples

- Create the target for each sample

- Save preprocessed datasets to S3

4.2 Feature Engineering

The next step is to perform feature engineering by aggregating the transactions at the customer level. For this task we used PySparkProcessor class which set up and runs a PySpark job defined in the 'spark/feature_eng.py' script. From this step we get 30 features for our models. An example with a few features is show in Figure 5.

id	num_unique_date	max_transaction_amount	sum_amount	avg_transaction_amount	days_since_last_transaction	days_since_first_transaction	avg_daily_amount
12262064	120	65.959999	2295.729980	4.114212	1	213	10.778075
12277270	93	24.980000	3601.310059	5.789887	2	210	17.149096
12332190	49	164.899994	1416.559937	5.169927	0	210	6.745523
12524696	70	21.990000	1581.670044	3.886167	3	213	7.425681
12682470	73	36.990002	1578.010010	4.347135	3	213	7.408498

Figure 5 – feature engineering with Spark

4.3 Baseline model

Our baseline for this problem will be two parametric models called BG-NBD (Beta Geometric Negative Binomial Distribution) and Gamma-Gamma model.

The BG-NBD model estimates the purchase frequency and churn rate according to five assumptions:

1. While active, the number of transactions made by a customer follows a Poisson process. This is equivalent to assuming that the time between transactions is distributed exponential with transaction rate λ.

2. Heterogeneity in λ follows a gamma distribution.

3. After any transaction, a customer becomes inactive with probability p.

4. Heterogeneity in p follows a beta distribution

5. The transaction rate λ and the dropout probability p vary independently across customers.

In order to fit this model, four parameters are estimated using MLE (Maximum Likelihood Estimation). A detailed description of the method is available in the original paper.

After obtaining the expected frequency and churn rate for each customer, the Gamma-Gamma model is used to estimate the monetary value for each transaction. The combination of both models can be used to obtain the expected revenue for the next 3 months.

4.4 Machine learning model

In the last notebook, we implement an XGBoost regression model with log squared error objective. The RMSLE metric is not supported by the default SageMaker built-in algorithm. Therefore, we need to pass a custom script as entry point to the *sagemaker.xgboost.estimator.XGBoost* class in order to train the model; this script is available at 'xgb/xgb_train.py'. We also perform some Hyperparameter tuning using the *HyperparameterTuner* class to search for the best hyperparameters for our model. To avoid overfitting the out-of-time sample, we split the development set in train and

validation sample. The validation sample is a 10% random sample from our customer-aggregated features. Using the out-of-sample (test) data to search for hyperparameters can impose a *selection bias* to our model, causing overfitting.

After training the model, we use the batch transform function to evaluate the performance of our model in the train, validation and test set. Finally, we deploy the model as an endpoint that accepts csv data as input for online predictions.

5. Results and conclusions

In this project we have trained and deployed a machine learning model using AWS SageMaker and S3. The performance of the machine learning model was compared with a statistical baseline and the results are presented in Figure 6.

Sample	Model	RMSLE	RMSE
train	Statistical model	0.6403	386.17
	XGBoost	0.7325	688.47
test	Statistical model	0.6210	353.60
	XGBoost	0.7153	630.36

Figure 6 – Score for each model and sample

The results for the statistical approach were better than the machine learning model. However, the XGBoost model depends a lot on the features extracted from the transaction history. In this project we performed some very basic feature engineering, so there is a lot of space to improve the machine learning model. The XGBoost model predictions were highly concentrated around the average transaction value for the next 90 days (Figure 7 and 8).

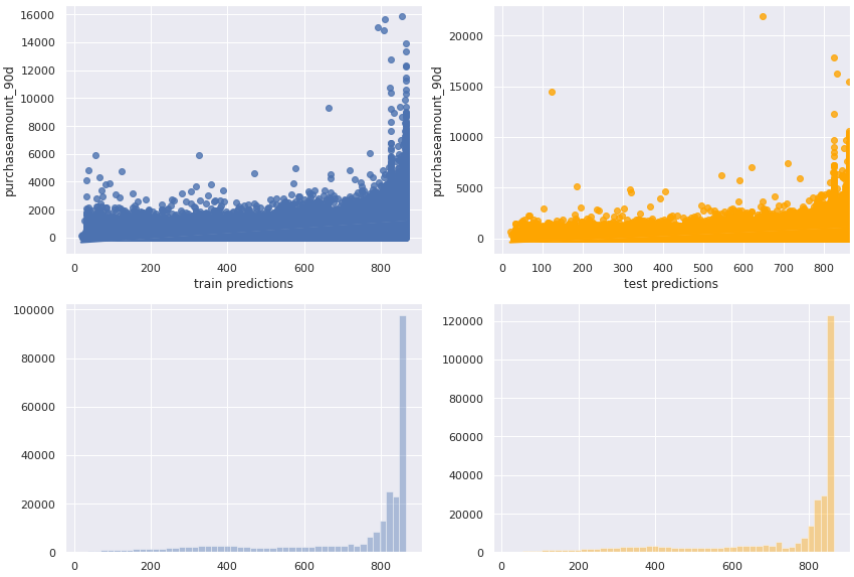

Figure 7 – Predictions vs actual values (top) and predictions distribution for the XGBoost model

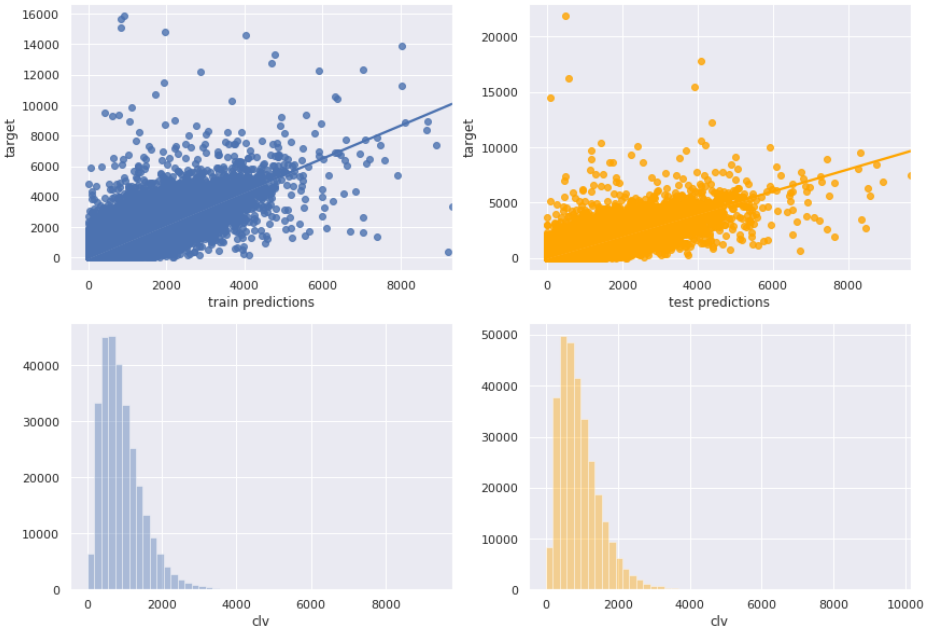

Figure 8 – Predictions vs actual values (top) and predictions distribution for the statistical model

This shows the relatively low predictive power of the features and a high bias/underfitting for this model. Therefore, the main step to improve the current model is a more careful feature engineering.