Starbucks Capstone Project v.2

2022-01-21

Domain background

The project's **domain background** — the field of research where the project is derived;

Historical
Stabucks is a well-known, relatively new retail company offering primarily coffee-based products that have taken over the world, and in particular the United States, by storm. Its success in changing Americans' coffee preferences is remarkable and so is the company's growth. Already in 2008 it began its digital transformation by offering loyalty cards and mobile apps, as well as promotional offers. These promotional offers have led to the generation of databases from which inferences can be drawn in order to offer promotional offers to its customers based on their responses to previous offers and thus find out which customers are more likely to respond to new offers.
From the above expanation one can deduce that for Starbucks and other similar companies point of view, the objective is to minimize customer churn.

In this respect, there is a lot of reasearch made, but I've concentrated in this paper in particular:

Predicting Customer Churn: Extreme Gradient Boosting with Temporal Data by Bryan Gregory (https://arxiv.org/pdf/1802.03396v1.pdf)

Because the customer churn research using machine learning is not new, so this paper is current, and it is closely related to the subject we are treating here. Besides, this paper won the WSDM Cup 2018 (*)

Personal motivation
When I saw this project I was very motivated because some time ago I

was thinking about a project for restaurantes that took a picture of the client at the moment of paying the check and perform a Face Recognition (a bit controversial).

This way one can create a database of clients and recognize the client as soon as she enters the restaurant in the future.

From that moment on, the restaurant management know exactly what this customer ordered last time or in previous visits.

In this way, a food recommendation system can tell the manager what tastings to offer for free so that the customer is tempted with new tastes and will consider the experience to come back again.

I see a similarity with the case of Starbucks which solved the problem of identifying the customer thanks to the fidelity Starbucks card she is presenting (**).

But the sense of the experiment is the same as the one I thought in the past.

I think that knowing the customer can serve to offer a better personalized experience according to their tastes and thus increase the possibility of them becoming a repeat visitor. And this is true not only for Starbucks, but for many other retailers, so it's a generically applicable project with three possible winners:
-the customer, who is better served and receives offers to his liking in terms of price/quality/service.
-the retailer, who sees an increase in his sales.
-the Machine Learning Engineer who finds a way to channel his skills.

Taking a look at the Starbucks Workspace I see that they provide much more data than I ever consider to obtain for the restaurant project as, for example, what offers the client received in the past and if she accepted or not.

Also demographic data is also relevant and cannot be inferred easily from a faceId.

So this project seems promising to be able to accomplish a good work in forecasting best customer treatment.

Problem Statement

A **problem statement** — a problem being investigated for which a solution will be defined;

The problem to be solved is to determine whether each offer is accepted by the customer in question. In the past a customer may have received several offers or receives only one offer. The task is to classify (compute probability) whether or not the offer results in a change to the order (acceptance of the offer) based on the characteristics of the customer's data set.

Datasets and inputs

The **datasets and inputs** — data or inputs being used for the problem;

As stated in the Starbucks Project Workspace, we will have three data sets in json format.
Below can be appreciated the number of features and number of records for each file.

profile.json

Rewards program users (17000 users x 5 fields)

- gender: (categorical) M, F, O, or null
- age: (numeric) missing value encoded as 118
- id: (string/hash)
- became_member_on: (date) format YYYYMMDD
- income: (numeric)

portfolio.json

Offers sent during 30-day test period (10 offers x 6 fields)

- reward: (numeric) money awarded for the amount spent
- channels: (list) web, email, mobile, social
- difficulty: (numeric) money required to be spent to receive reward
- duration: (numeric) time for offer to be open, in days
- offer_type: (string) bogo, discount, informational
- id: (string/hash)

transcript.json

Event log (306648 events x 4 fields)

- person: (string/hash)
- event: (string) offer received, offer viewed, transaction, offer completed
- value: (dictionary) different values depending on event type
  - offer id: (string/hash) not associated with any "transaction"
  - amount: (numeric) money spent in "transaction"
  - reward: (numeric) money gained from "offer completed"
- time: (numeric) hours after start of test

These datasets are described in the link below (**)

-First, I will check if there are class imbalance. This will be important to decide about preprocessing and metrics.
I will follow this guide to deal with imbalance: *https://machinelearningmastery.com/what-is-imbalanced-classification/*
-I will upload that data to an s3 bucket and we will access to it to perform EDA to see class imbalance, or any correlations that can show up between variables and look for data integrity, looking for outliers, missing data, or any anomalies we can find on it to avoid the famous “garbaje in-garbage out”.
-Then I will perform training on the demographic data where the variable to be predicted will be offer_status:
three numerical values for offers offers received, offers viewed, and

offers completed.

Solution statement

A **solution statement** — the solution proposed for the problem given;

We can compute by using EDA, which offers are most accepted by groups like gender and age.
I we find something consistent this is an starting point. Then, given a particular customer, we can first associate her/him with a predetermined demographic group. Then we can apply the customer's own history about aceeptance/refusal of a given offer and combine both statistics.
I will make a try of using some kind of supervised method like XGBoost, that is so powerfull that wins most of the Kaggle competitions.
As I have gained some skills in the past lessons of the MLEND, I will use those tasks to be able to use s3 as data repository, perform EDA on that data, perform data curation in case I detect missing or outlier values, perform hyperparameter optimization as well as debugging to finally deploy endpoints to perform inference and perform an analysis to get some insight of the outputs obtained.

Benchmark model

A **benchmark model** — some simple or historical model or result to compare the defined solution to:

The obvious benchmark is a logistic regression with CrossEntropy as a metric and a softmax in the last layer to be able to perform multi class predictions.

Evaluation metrics

A set of **evaluation metrics** — functional representations for how the solution can be measured;

The most commonly used metrics for multi-classes are **ROC-**

AUC, Log-loss, F1 score, Average Accuracy.
Log-loss is the metric used in the mentioned paper “Domain Background” so it proved its efectiveness in that case.
Average Accuracy, as pointed by the reviewer could not be a good metric if the dataset classes are imbalanced. So once we perform the EDA, we will know if we should discard this possibility in faver or ROC-AUC metric, which deals with imbalance much better.
There is an interesting discussion about it here: *https://stats.stackexchange.com/questions/210700/how-to-choose-between-roc-auc-and-f1-score*

Project design

An outline of the **project design** — how the solution will be developed and results obtained.

Intended steps will be:

Step 1: data preprocessing

- Reading data files into a pandas dataframe.
- Describe data using pandas.
- See if scaling or normalization will be necessary.
- Look for outliers.
- Look for missing data
- Perform data cleaning with dropna and interpolation if necessary.
- Create histograms to see imbalances.
- Upload data to s3.

Step 2: create the model and training:

- Create a train_model.py file for the sake of creation of the model and training.
- Test more than one model from SKLearn (*RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier*)
- Perfdorm hyperparameter optimization using ranges of hyperparameters by GridSearch.
- Deploy the models for inference
- Make reflexions about the results obtained and report them.

References:
*(*) **WSDM Cup** is a machine learning competition-style event co-located with the leading WSDM (Web Search and Data Mining) conference.*

(**) Intro:
https://classroom.udacity.com/nanodegrees/nd189/parts/cd0549/modules/864d3e12-dc8d-47c6-b443-d01d6c7aedde/lessons/d6ec6005-e421-455c-9b79-ebc2df44e1ac/concepts/1d4b7aa2-2817-4448-a121-87bb45a782b1

(***) Workspace
https://classroom.udacity.com/nanodegrees/nd189/parts/cd0549/modules/864d3e12-dc8d-47c6-b443-d01d6c7aedde/lessons/d6ec6005-e421-455c-9b79-ebc2df44e1ac/concepts/c7f43c36-9fb1-4b15-aaa0-fe343f198dad

Proposal submission:
https://classroom.udacity.com/nanodegrees/nd189/parts/cd0549/modules/864d3e12-dc8d-47c6-b443-d01d6c7aedde/lessons/2bf7b504-cade-423f-acac-3a7b801b1f53/project