Capstone Project

Rodolfo Jerônimo Teles

18th August 2021

1 Definition

1. Project Overview

The project chosen to be the capstone project was one of the projects recommended by Udacity as the last project of the Machine Learning Engineer Nanodegree.

The domain background chose to this project was the regression problem. Specifically, this project intends to predict future values, according to historical patterns which will be learned by the machine learning algorithm during training process. This study field is known as time series forecasting.

A field behind time series forecasting is time series analysis which studies methods for analysing data aiming extract relevant statistics and insights of the data. Thus, time series forecasting can be defined as the use of methods to predict values based on previously observed data.

Some of the applications of time series forecasting are: weather forecasting, retail sales forecasting and recommendation systems. In this project, time series forecasting will be used to predict Starbucks transactions after users see and complete offers sent through the app.

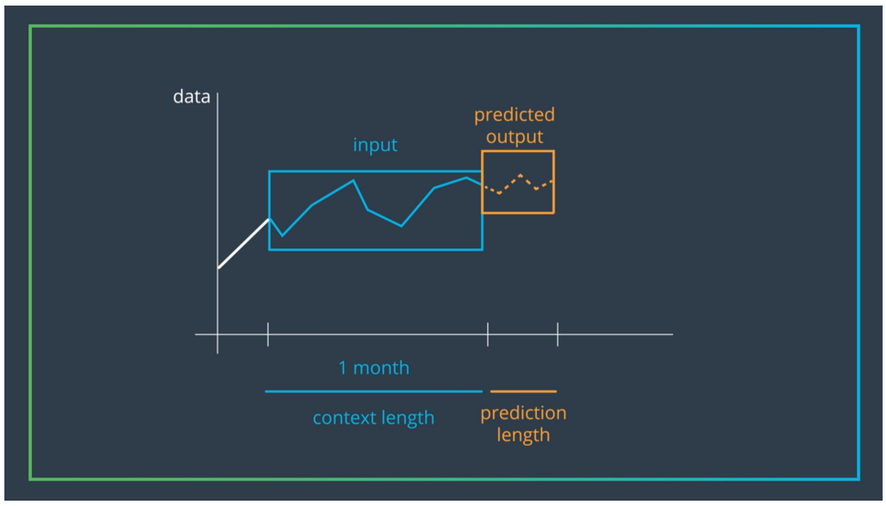


Figure 1: Time series forecasting problem illustration. [8]

The data to be used was provided by Udacity and contains simulated data that mimics customer behavior on the Starbucks rewards mobile app. The available data are:

(a) Portfolio: contains Starbucks offers data. Data available:

- id (string) - offer id;
- channels (list of strings);
- difficulty (int) - minimum required spend to complete an offer;
- duration (int) - time for offer to be open, in hours;
- offer_type (string) - type of offer i.e. BOGO, discount, informational;
- reward (int) - reward given for completing an offer;

(b) Profile: contains users demographics anonymized data. Available features:

- age (int) - age of the customer;
- became_member_on (int) - date when customer created an app account;
- gender (str) - gender of the customer (note some entries contain 'O' for other rather than M or F);
- id (str) - customer id;
- income (float) - customer's income.

(c) Transcript: contains data about users actions related on offers like: users interactions, offer received and offer viewed.

- event (str) - record description (i.e. users interactions, offer received, offer viewed, etc.);
- person (str) - customer id;
- time (int) - time in hours since start of test. The data begins at time t=0;
- value - (dict of strings) - either an offer id or transaction amount depending on the record.

2. Problem Statement

The Starbucks was founded by Jerry Baldwin, Gordon Bowker, and Zev Siegl in 1971 in Seattle (USA) [1] and today is the largest coffeehouse chain in the world. The first app launched by Starbucks was in September 23, 2009 in the App Store. The first version of MyStarbucks application offered a store locator, nutrition information and an interactive drink builder.[5]

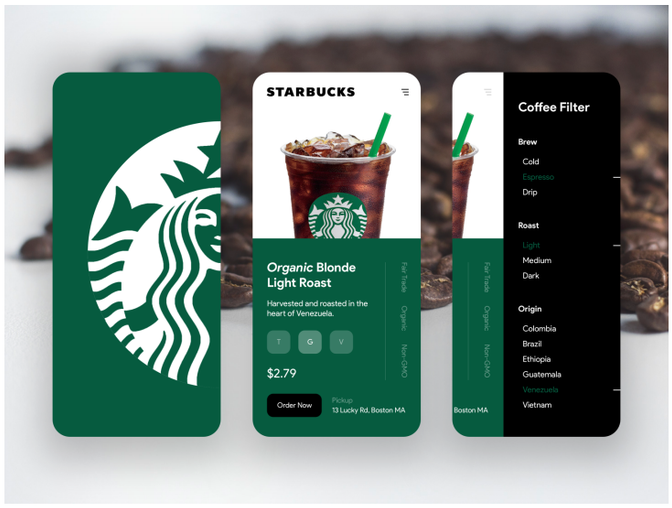


Figure 2: Starbucks Rewards App

The machine learning model which will be used to predict how much a user will spend next day will be deployed using Amazon SageMaker. To be consumed, the model will use a web-app that will allow a user pass his ID, using it as a trigger to Amazon API Gateway which will trigger a lambda function to pre-process the input data and send pre-processed input data to the model. The main tasks involved to create the recommendation system are the following:

(a) Exploring and cleaning the data;

(b) Choosing a model;

(c) Training an estimator that can predict how much someone will spent according to the offer received;

(d) Building an API using Amazon API Gateway and make the estimator run on a web-app;

(e) Make the web-app predicts a how much a user will spend in each offer to the next day.

3. Metrics

Evaluate model performance is a essential step to get the best predictions as possible to achieve. Thus, to evaluate how well the model predicts the users purchases, it will be used the mean squared logarithmic error (MSLE) which is defined by:

$$L(y, \hat{y}) = \frac{1}{N}\sum_{i=0}^{N}(log(y_i + 1) - log(\hat{y}_i + 1))^2 \tag{1}$$

Another relevant metric to evaluate the model performance which will be used in this project is the Root Mean Squared Error (RMSE) that is defined by:

$$L(y, \hat{y}) = \sum_{i=1}^{N}\sqrt{\frac{(\hat{y}_i - y_i)^2}{N}} \tag{2}$$

2 Analysis

(a) Data Exploration

During data exploration phase, it was explored some basics statistics about the data provided.

i. Portfolio: contains Starbucks offers data.

	reward	channels	difficulty	duration	offer_type	id
0	10	[email, mobile, social]	10	7	bogo	ae264e3637204a6fb9bb56bc8210ddfd
1	10	[web, email, mobile, social]	10	5	bogo	4d5c57ea9a6940dd891ad53e9dbe8da0
2	0	[web, email, mobile]	0	4	informational	3f207df678b143eea3cee63160fa8bed
3	5	[web, email, mobile]	5	7	bogo	9b98b8c7a33c4b65b9aebfe6a799e6d9
4	5	[web, email]	20	10	discount	0b1e1539f2cc45b7b9fa7c272da2e1d7
5	3	[web, email, mobile, social]	7	7	discount	2298d6c36e964ae4a3e7e9706d1fb8c2
6	2	[web, email, mobile, social]	10	10	discount	fafdcd668e3743c1bb461111dcafc2a4
7	0	[email, mobile, social]	0	3	informational	5a8bc65990b245e5a138643cd4eb9837
8	5	[web, email, mobile, social]	5	5	bogo	f19421c1d4aa40978ebb69ca19b0e20d
9	2	[web, email, mobile]	10	7	discount	2906b810c7d4411798c6938adc9daaa5

Figure 3: A sample of the data set "portfolio".

```
---------- Dataset:  portfolio
reward         int64
channels      object
difficulty     int64
duration       int64
offer_type    object
id            object
dtype: object
          reward  difficulty   duration
count  10.000000   10.000000  10.000000
mean    4.200000    7.700000   6.500000
std     3.583915    5.831905   2.321398
min     0.000000    0.000000   3.000000
25%     2.000000    5.000000   5.000000
50%     4.000000    8.500000   7.000000
75%     5.000000   10.000000   7.000000
max    10.000000   20.000000  10.000000
(10, 6)
```

Figure 4: Basic statistics of the data set "portfolio".

It could be seen that in the portfolio data set there was ten offer types with specific difficulty levels, offer duration, distribution channels and rewards. During exploratory analysis, it was seen there were only three different offer types: bogo, informational and discount. To discount and bogo offer types there were four variants of each and two variants of the informational offer type.
Looking at the data set it also could be observed there were have four channels types: web, email, mobile and social. Thus, an offer could last a maximum of ten days and maximum reward is $10 dollars to bogo offer variant.

ii. Profile: contains users demographics anonymized data.

	gender	age	id	became_member_on	income
0	None	118	68be06ca386d4c31939f3a4f0e3dd783	20170212	NaN
1	F	55	0610b486422d4921ae7d2bf64640c50b	20170715	112000.0
2	None	118	38fe809add3b4fcf9315a9694bb96ff5	20180712	NaN
3	F	75	78afa995795e4d85b5d9ceeca43f5fef	20170509	100000.0
4	None	118	a03223e636434f42ac4c3df47e8bac43	20170804	NaN

Figure 5: A sample of the data set "profile".

```
---------- Dataset:  profile
gender               object
age                   int64
id                   object
became_member_on      int64
income              float64
dtype: object
                age  became_member_on         income
count  17000.000000      1.700000e+04   14825.000000
mean      62.531412      2.016703e+07   65404.991568
std       26.738580      1.167750e+04   21598.299410
min       18.000000      2.013073e+07   30000.000000
25%       45.000000      2.016053e+07   49000.000000
50%       58.000000      2.017080e+07   64000.000000
75%       73.000000      2.017123e+07   80000.000000
max      118.000000      2.018073e+07  120000.000000
(17000, 5)
```

Figure 6: Basic statistics of the data set "profile".

During data exploration, it was found 2175 records where the columns 'gender' and 'income' are "not a mumber" (NaN) values in the profile data set. It also was seen whenever there was a "not a number" (NaN) value in one of the columns 'gender' or 'income' there was an 'age' equals to 118, which means is a default value when the user doesn't inform 'gender', 'income' and 'age'.
As we got 12.8% of the records with "not a mumber" (NaN) values, a relevant number, let's fill the rows with a new category 'Uninformed' to 'gender' column and use the mean to fill NaN values in 'income' column.

(b) Transcript: contains data about users actions related on offers like: users interactions, offer received and offer viewed.

	person	event	value	time
0	78afa995795e4d85b5d9ceeca43f5fef	offer received	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}	0
1	a03223e636434f42ac4c3df47e8bac43	offer received	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}	0
2	e2127556f4f64592b11af22de27a7932	offer received	{'offer id': '2906b810c7d4411798c6938adc9daaa5'}	0
3	8ec6ce2a7e7949b1bf142def7d0e0586	offer received	{'offer id': 'fafdcd668e3743c1bb461111dcafc2a4'}	0
4	68617ca6246f4fbc85e91a2a49552598	offer received	{'offer id': '4d5c57ea9a6940dd891ad53e9dbe8da0'}	0

Figure 7: A sample of the data set "transcript".

```
---------- Dataset:  transcript
person    object
event     object
value     object
time       int64
dtype: object
                time
count  306534.000000
mean      366.382940
std       200.326314
min         0.000000
25%       186.000000
50%       408.000000
75%       528.000000
max       714.000000
(306534, 4)
```

Figure 8: Basic statistics of the data set "transcript".

In the transcript data set, it was found four types of user interaction with offers: offer received, offer

viewed, offer completed and transaction. To solve the problem of this project, it is expected to use 'transaction' records to get how much and when a user spent.

Aiming understand the users behaviour which received and viewed an offer and after that made a transaction during the offer duration period, it was needed to filter the transcript data set to only use transactions data.

After initial data processing, the transcript data set was filtered just to bring transactions data. Therefore, it was possible to plot a boxplot and histogram of the transactions from completed offers.

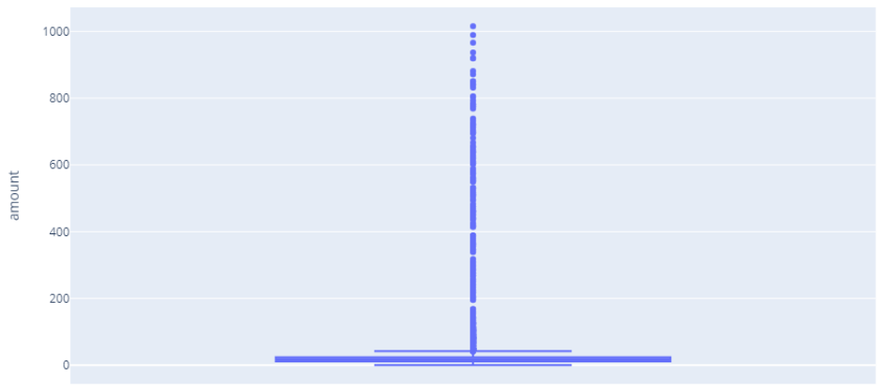


Figure 9: Boxplot without outliers replacement by upper fence value of the transactions amount values from completed offers.

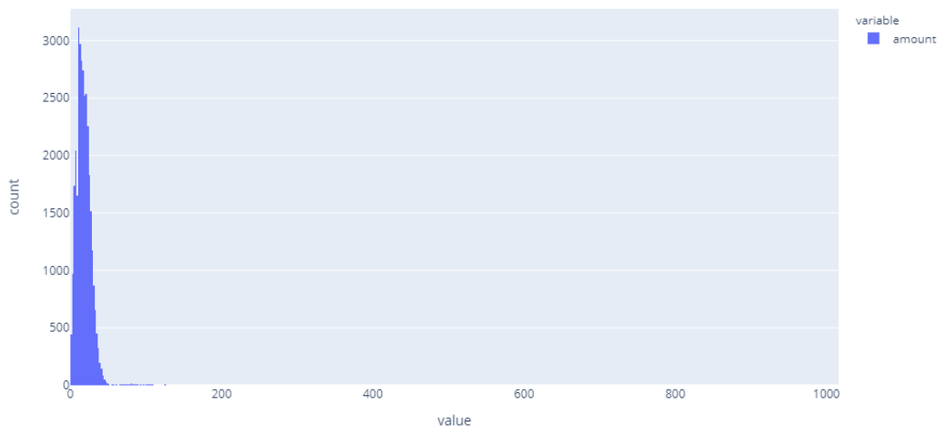


Figure 10: Histogram without outliers replacement by upper fence value of the transactions amount values from completed offers.

(c) Exploratory Visualization

During the exploratory visualization phase, the following charts were plotted:

i. A sample of hourly mean transactions
With the amount transactions values sampled hourly basis it could be seen that the mean values assumed a range between about 14 and 30 dollars. It couldn't be possible use the data sampled hourly basis to train the model, once hourly it just have 120 data records.

ii. Net revenue over time.

iii. Total sum of transactions with rewards over time.
At the charts above it could been seen every seven days in the first two weeks there was a clearly defined pattern, where the first day of the week is the day with highest revenue and the following days have a decreasing revenue. The following weeks have another pattern starting the first three days of the week in a decreasing revenue, after that we got a peak value with decreasing values in the following days.
The rewards values data followed about the same pattern as amount values what could indicate that there is a positive correlation between transactions amount values of completed offers and rewards values.

iv. Offer types count by offer difficult.
The net chart above shows there was more transactions to 'discount' offers than 'bogo' ones, what could point to discount offers are more easy to be accepted than 'bogo' type. Beyond that, to difficulty level 10 and discount offer type it had the most part of the data. While to 'bogo' offer type offers of difficulty level 5 were the major amount.

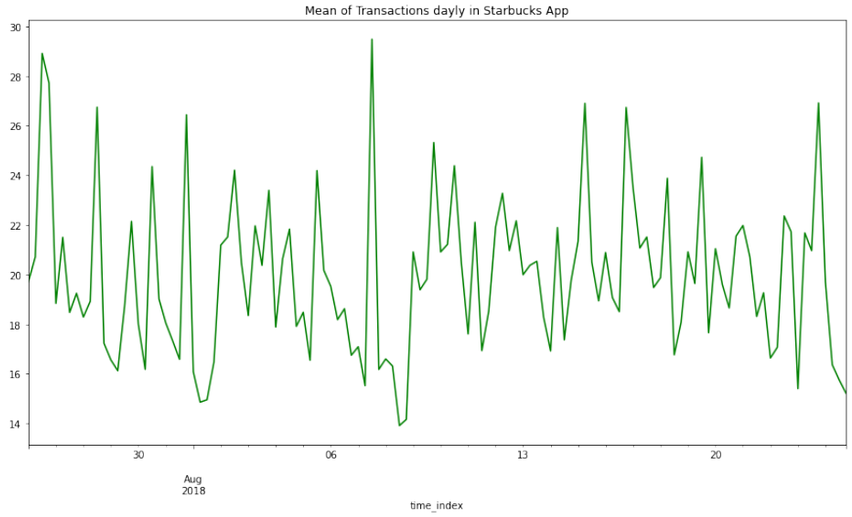


Figure 11: Mean of transactions hourly in Starbucks app.

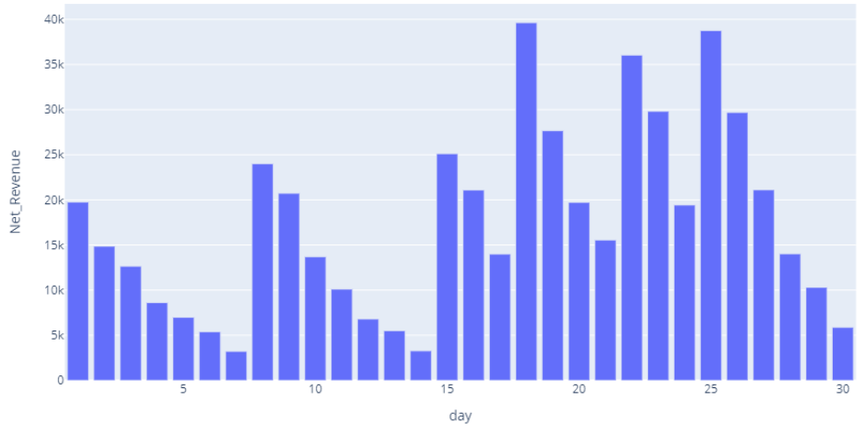


Figure 12: Net revenue of transactions over time.

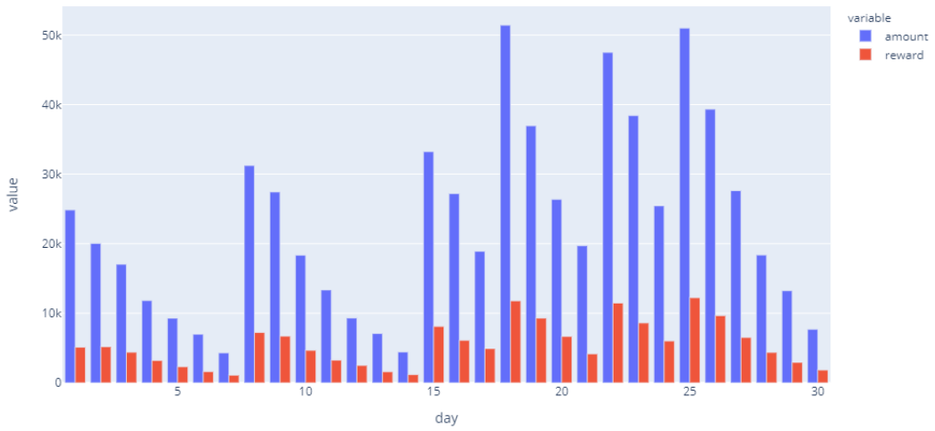


Figure 13: Total sum of transactions with rewards over time.

(d) Algorithms and Techniques

Initialy, it was planned to use the model Amazon SageMaker DeepAR once the model was used during the Energy Consumption forecasting project [4], got a good performance and has a strong similarity with predict transactions amount values. However, during data processing and the attempts to train the SageMaker DeepAR model it was clear that the model would not be useful to solve the proposed problem of predicting how much a user would spend the next day according to an offer, once it got two limitations:

i. Using the data with granularity person, hour of the day, offer would bring a small and different numbers of time series to each person in the training data set, what would compromise the predictions assertiveness.

ii. Using the data with granularity hour and offer, i.e, re-sample the training data to get mean transaction amount value in hourly basis, it would bring a less variant set of values, what would be good

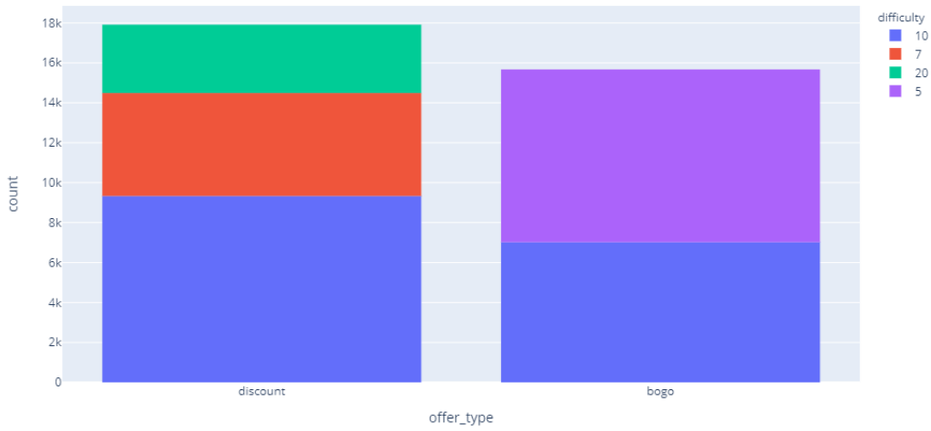


Figure 14: Offer types counts by offer difficult.

for the model performance, but with just 120 data points wouldn't be enough to train and test the model.

Due to those limitations and to don't change the proposed goal and solution, it was decided to change the model chosen to the XGBoost model.

According to official the library documentation [9], XGBoost implements machine learning algorithms under the Gradient Boosting framework and is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It provides a parallel tree boosting that solve many data science problems in a fast and accurate way.

Extreme Gradient Boosting (XGBoost) originated from the term "Gradient Boosting" from the Greedy Function Approximation article: A Gradient Boosting Machine [3]. Gradient boosting is a supervised learning algorithm that tries to predict a target variable by combining an ensemble of estimates from a set of simpler and weaker models [2].

The model of XGBoost implements a decision tree ensemble, which consists of a set of classification and regression trees (CART).

To use XGBoost correctly, it was needed clean the data, treat null values, encode categorical features (person id, offer id, offer type and gender, split the data) and upload processed data to an Amazon S3 bucket.

(e) Benchmark

One of the models used as benchmark during training phase was obtained from Amazon SageMaker XGBoost using a specific image container with

```
get_image_uri(session.boto_region_name, 'xgboost')
```

and used the following hyperparameters:

i. max_depth=20;
ii. eta=0.2;
iii. gamma=4;
iv. min_child_weight=5;
v. subsample=0.6;
vi. objective="reg:linear";
vii. early_stopping_rounds=10;
viii. num_round=300.

To this set of hyperparameters the obtained metrics were:

i. Root Mean Squared Error in train set: 7.73222
ii. Root Mean Squared Error in test set: 8.97852

3 Methodology

(a) Data Processing

i. Cleaning
The cleaning step in this project had a relevant function as it was found 2175 rows with NaN values in the profile data set in the "gender" and "income" columns. In those cases, whenever there was a NaN value in one of "gender" or "income" columns there was an age equals to 118 years old, what takes it to conclude that person didn't complete or didn't her or him register. To fill those cases, it was created a new "gender" class "Uninformed" and it was calculated mean income to use as replace value in "income" column which was $65,404.99 dollars.

ii. Processing
The first processing step was filter the transcript data set just to bring transaction data. After filtered, it was necessary to bring the offer id as a new column, once the transactions records didn't

have the the offer id in the "value" column. Thus, aiming a merge with profile and portfolio data sets, it was created a day column from column "time" divided by 24 and made an inner merge of the transcript filtered data set with completed offers data set which had offer id in the 'value' column. So, to do that merge it was used the columns "time", "day" and "person" as keys to create a data set with transactions values and 'offer_id'. This way, it was possible make another inner merge with portfolio data set using 'offer_id' and 'rewards' columns as keys.
Analysing the box plot it could be observed how many outliers the data have. One way to treat outliers is replace them by upper fence value, which in the case of transactions amount values from completed offers was "42.1". Therefore, the data was processing again and it was obtained the following box plot and histogram:

Figure 15: Box plot after outliers replacement by upper fence value of the transactions amount values from completed offers.

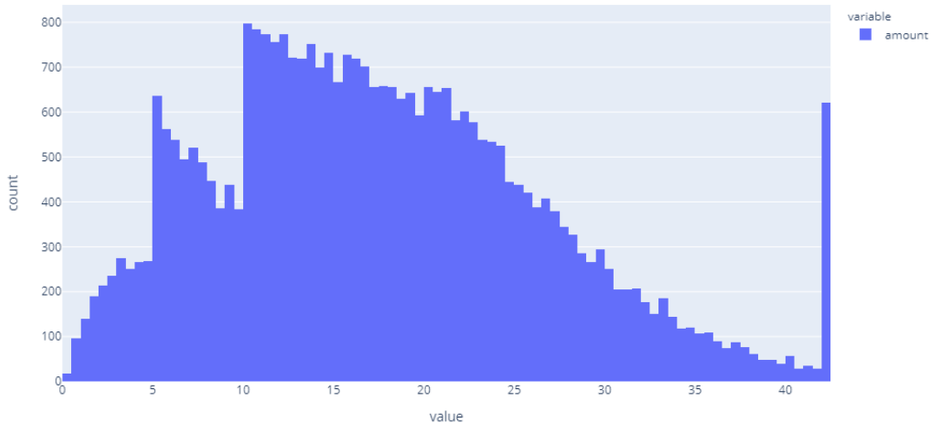


Figure 16: Histogram after outliers replacement by upper fence value of the transactions amount values from completed offers.

Even though it was obtained a peak value in the upper fence value, the distribution and the box plot had a clean and better result to use in the model.

(b) Implementation
The final architecture of the system can be viewed bellow:

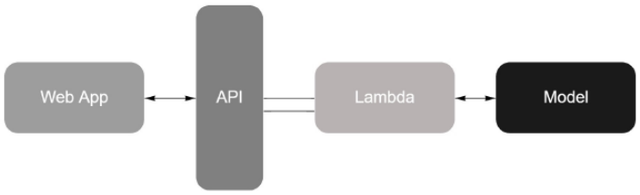


Figure 17: Final solution architecture.

After data processing step, the final input data used to train the model contains the following features:

i. Time of the day the transaction
ii. Person ID who completed the offer

iii. Person gender who completed the offer
iv. Offer ID of the transaction
v. Offer type of the transaction

The final model used was obtained from Amazon SageMaker XGBoost library and used the following hyperparameters:

i. max_depth=20;
ii. eta=0.2;
iii. gamma=4;
iv. min_child_weight=5;
v. subsample=0.6;
vi. objective='reg:linear';
vii. early_stopping_rounds=10;
viii. num_round=500.

To the final set of hyperparameters used the obtained metrics were:

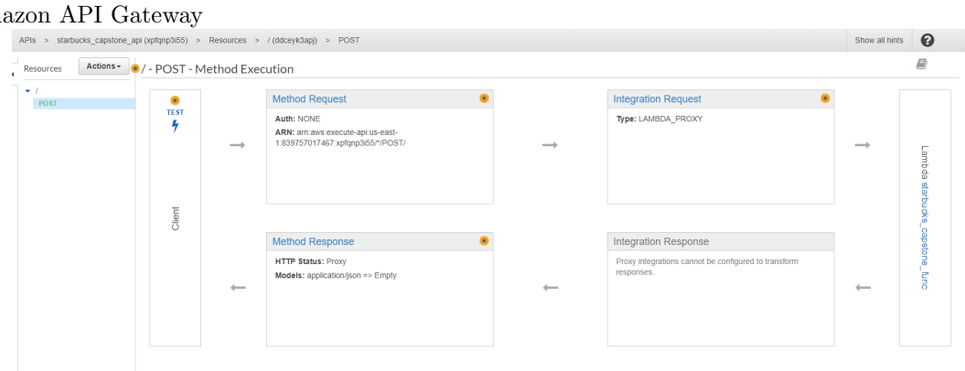
i. train-rmse: 7.73222
ii. validation-rmse: 8.97852

Following the planned architecture, it was necessary to build a Amazon Lambda function, an Amazon API Gateway and a Web App.

i. Amazon Lambda function It was used an Amazon Lambda function created during the project "Creating a Sentiment Analysis Web App" at the Machine Learning Engineer Nanodegree as reference. [7]

```
# We need to use the low-level library to interact with
# SageMaker since the SageMaker API
# is not available natively through Lambda.
import boto3

def lambda_handler(event, context):

    # The SageMaker runtime is what allows us to invoke
    # the endpoint that we've created.
    runtime = boto3.Session().client('sagemaker-runtime')

    # Now we use the SageMaker runtime to invoke our
    # endpoint, sending the review we were given

    endpoint_created = 'xgboost-2021-08-11-00-16-11-486'
    # The name of the endpoint we created
    response = runtime.invoke_endpoint(EndpointName = endpoint_created,
                                       ContentType = 'text/csv',
                                       Body = event['body'])

    # The response is an HTTP response whose
    # body contains the result of our inference
    result = response['Body'].read().decode('utf-8')

    return {
        'statusCode' : 200,
        'headers' : { 'Content-Type' : 'text/plain',
                      'Access-Control-Allow-Origin' : '*' },
        'body' : result
    }
```

ii. Amazon API Gateway



API Endpoint created: https://xpfqnp3i55.execute-api.us-east-1.amazonaws.com/PROD

iii. Web App
The web app was built using the library Streamlit [6] which is a powerful tool to develop shareable web apps through Python scripts. It was needed to create some preprocessing code to transform viewable data from the app into input data to the model. The created functions can be found in the link:
2.Starbucks_Capstone_notebook-Training_an_Estimator.ipynb
The web app code can be found here: App.py

4 Results

(a) Model Evaluation and Validation

The final model was chosen due the highly efficient, flexible and portable of the XGBoost model. Once it solves many data science problems in a fast and accurate way, XGBoost is a valid choice to predict the transactions amount value of the completed offers from Starbucks products.

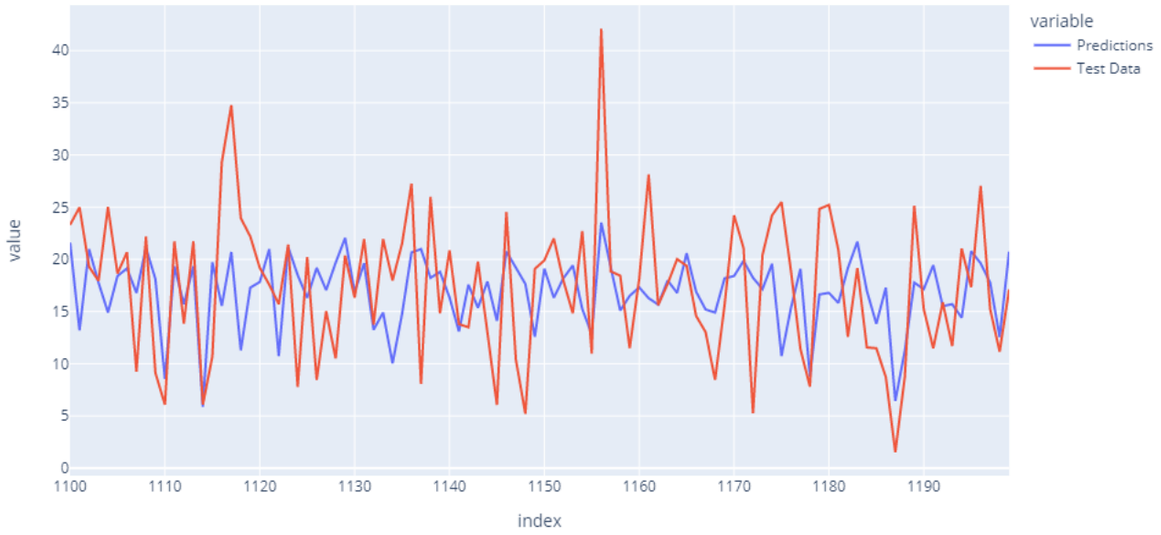


Figure 18: Final model performance compared to test set.

Due to outliers treatment, replacing them to upper fence value (42.1) it can be seen that the model didn't perform well when it tries to predict higher transaction values in the test set, once it was not created a preprocess step to treat outliers in the test set. On the other hand, graphically values near the mean value (20.49) had a good approximation in that sample in the test set.

The results of the model being consumed in the web app can be found bellow:

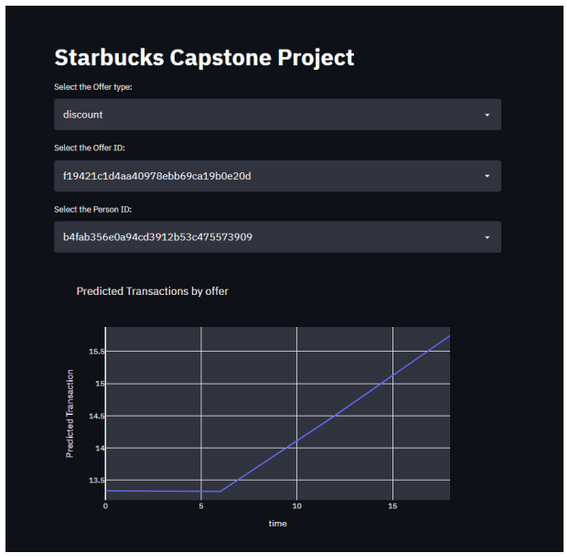


Figure 19: Case 1. Final solution in the web app built.

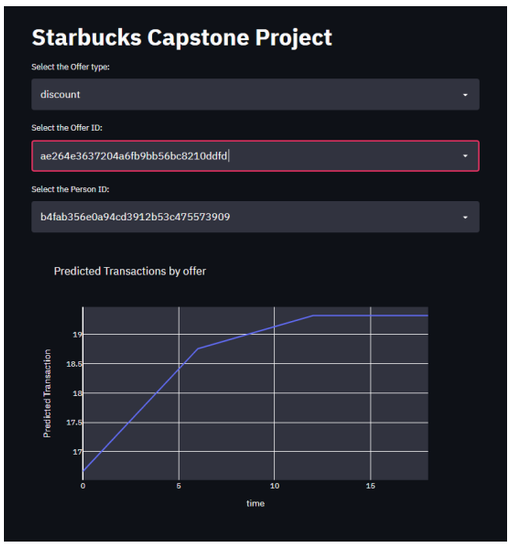


Figure 20: Case 2. Final solution in the web app built.

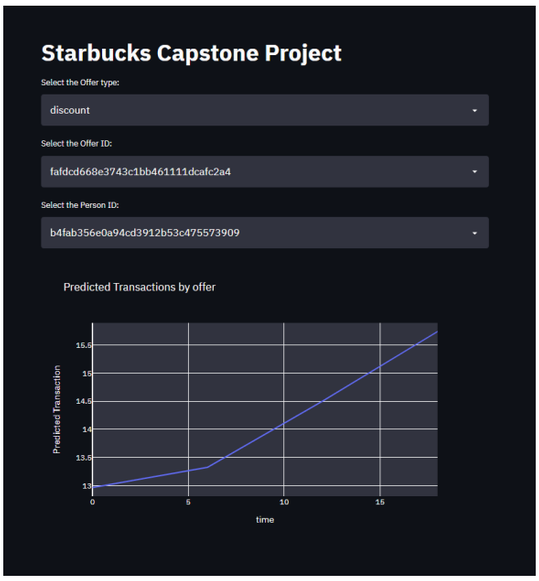


Figure 21: Case 3. Final solution in the web app built.

5 Conclusion

At this project, it was possible create a web app that allows an user to set an offer type, an offer id and a customer id to predict how much the customer will spend on Starbucks products based on historical users transactions data.

Acquiring a data set with more transactions from completed offers, some improvements the model could receive is related to add day of the month and month of the year to the model, so that the model can understand the changes in customers behavior during in the month and in the year.

Treating outliers before make predictions could be another huge improvement and significantly increase the perform of the model to higher values than 42.1 (upper fence value).

References

[1] Britannica. Howard schultz, american businessman.

[2] Amazon SageMaker documentation. Xgboost algorithm.

[3] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine.

[4] Udacity Machine Learning Engineer Nanodegree. Time series forecasting, energy consumption solution.

[5] RetailDive. Starbucks rolls out largest mobile payments, loyalty play in us.

[6] Streamlit. 2021 streamlit, the fastest way to build and share data apps.

[7] Udacity. 3.6. creating a sentiment analysis web app. machine learning in production.

[8] Udacity. 4.5.1. time-series forecasting. machine learning engineer nanodegree program. time-series forecasting illustration.

[9] XGBoost. Xgboost documentation.