Project Report

ChihYung(Raymond) Wu
2/23/2020

Definition

a. Project Overview

My project is based on the Udacity's Starbucks Project. As previously a business-major student, I found the integration of technology and business quite intriguing and always provide useful insights for business development. As a result, I would like to take advantage of the capstone project opportunity to develop a method to apply machine learning to business, especially in the field of Marketing in this case.

In this case, we will be investigating into various Starbucks marketing campaign info and its simulated customers' info in order to figure out the pattern for the most popular and preferred marketing channels and marketing offers.

b. Problem Statement

Before developing a strategy to resolve our problem, we first need to make sure we are asking an appropriate question. As this project falls in the genre of marketing, the ultimate goal for this project is to help Starbucks maximize its rate of return on marketing expenditure by classifying each customer properly and target each customer group with the right ads in the right channels. Furthermore, via exploring the solution to the suggested problem, we also want to figure out what features carry heavy weights for classifying various users.

Overall, the questions we will be trying to answer in this project will be as follows:

- How offer types vary from each other based on different pairs of features?
- Check out the distribution of users who have consumed offers
- To see if users can be classified by offer ID
- To see if users can be classified by offer Type

c. Metrics

As we will be exploring the relation between each feature specified in the datasets, we will ultimately figure out a method to group each user. As a result, this will be considered an Unsupervised Machine Learning project.

To evaluate the accuracy and efficiency of an unsupervised machine learning model we will adopt the SSE (Sum of Squared Error) metrics to make sure that the data points can be clearly distinguished among groups by finding the clear ankle point when using the elbow method in the SSE chart.

Analysis

a. Data Exploration

The datasets we will be using for this project will be coming from the three files provided by Udacity and their info are listed as below.

- portfolio.json - containing offer ids and meta data about each offer (duration, type, etc.)
- profile.json - demographic data for each customer
- transcript.json - records for transactions, offers received, offers viewed, and offers completed

The fields/features in each of these 3 files are listed as below:

portfolio.json

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

profile.json

- 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

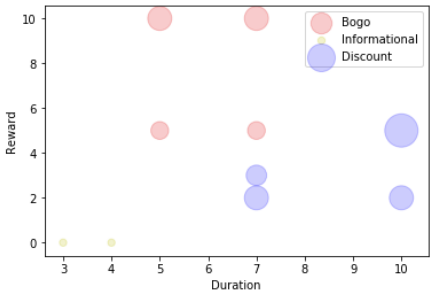
transcript.json

- event (str) - record description (ie transaction, 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

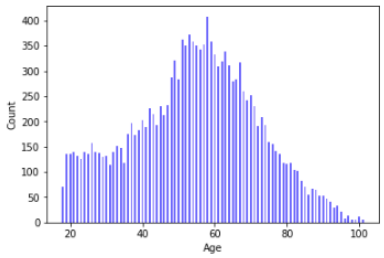
b. Exploratory Visualization

As one of our goals for this project is to figure out which offer type is suitable for each user, it certainly will be helpful for us to discover the distribution of these offer types among various features. As result, below is the distribution of the offer types between features such as Reward, Duration, and Difficulty.

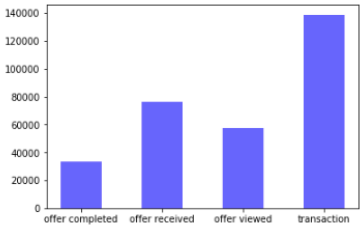
```
In [319]: offer_types = portfolio['offer_type'].unique()
          offer_types

Out[319]: array(['bogo', 'informational', 'discount'], dtype=object)
```



From the graph above, we can easily find out actually the 3 offer types can be easily grouped in their own clusters, which provides us an important insight that these 3 types of offers are cleanly distinguishable rather than mixing with each other. Furthermore, we believe that this can help us explain our findings based on offer types more easily.



The chart above indicates the distribution of customer's ages. From there, we can find out that the customers' ages gather around the mean age of around 56 to 58 years old. Below is another chart showing the distribution of offer types. As a result, we saw that the transaction type accounts the majority. This helps us better understand how much data is available for us to develop a methodology to analyze how campaign info can impact consumers' behavior.



c. Algorithms and Techniques

The algorithm we will be using is K-means algorithm to help us discover each user group. The strategy we will be using to discover those groups starts with the workflow as below. First, we will need to make sure we have gathered all the important and highly impactful features needed to produce a meaningful prediction result. Once all features'

data are ready, we will build a PCA model to analyze the variance explanation by different number of components. Once the number of components is decided, we will then reverse engineer the components to discover their composition. Subsequently, we should be able to find out the centroids of each cluster based on these components. With the information of centroids in hand, we should then be able to extract the groups' centroids info based on the original features. Ultimately, we then get a much better picture of what features can determine the groupings of a specific user.

d. Benchmark

In order to ensure the model can produce meaningful and useful predictions, we need to make sure that users can be evenly classified into various groups. As a result, we need to make sure the components we use to explain the variance between data is high enough, and we set this benchmark to be 90%.

Methodology

a. Data Preprocessing

Before we can implement our algorithm to classify each user for his/her own group, we need to make sure all data points are in the valid format for model training. The respective data preprocessing steps are listed as follows.

- Replace users of age 118 years old with NaN values

```
In [323]: # Convert age 118 to np.nan
          profile['age'] = profile['age'].apply(lambda a: np.nan if a == 118 else a)
```

- Convert the feature "became_member_on" to feature "membership"

```
now = datetime.datetime.today().date()
profile['membership'] = pd.to_datetime(profile['became_member_on'], format='%Y%m%d').apply(lambda t: (now-t.date()).da
profile.drop(['became_member_on'], axis=1, inplace=True)
```

- Remove all data rows with missing values

```
In [350]: trimmed_df = new_model_df.dropna()
```

- Obtain the counts of users who complete offers after viewing and receiving offers within valid offer duration

```
In [339]: # Here we are trying to calcuate the number of offer completed after both offer received and offer viewed happened wit
def user_offer_completion_counts(group):
    offer_counts = 0
    prev_offer_completed_index = 0
    latest_offer_received_index = None

    for i in range(len(group)):
        if group.iloc[i]['event'] == 'offer received':
            latest_offer_received_index = i

        if group.iloc[i]['event'] == 'offer completed':
            for j in range(i):
                in_time = latest_offer_received_index != None and int(group.iloc[i]['time']) - int(group.iloc[latest_of

                if group.iloc[j]['event'] == 'offer viewed' and j >= prev_offer_completed_index and j > latest_offer_re
    #                 print(int(tt.iloc[j]['time']))
    #                 print(int(tt.iloc[j]['duration']))
    #                 print(latest_offer_received_index)
    #                 print(i)
                    offer_counts += 1
            prev_offer_completed_index = i

    return offer_counts
```

- Get dummy data for gender features

```
In [353]: new_gender_columns = pd.get_dummies(trimmed_df['gender'])

In [354]: new_gender_columns.head()
```

Out[354]:

Customer	F	M	O
0610b486422d4921ae7d2bf64640c50b	1	0	0
78afa995795e4d85b5d9ceeca43f5fef	1	0	0
e2127556f4f64592b11af22de27a7932	0	1	0
389bc3fa690240e798340f5a15918d5c	0	1	0
2eeac8d8feae4a8cad5a6af0499a211d	0	1	0

b. Implementation

To cluster each user point with the effective features, we will use not only their profile attributes such as age, income, and gender, but also the statistics of their offer completion, which means that we also want to know the number of times each user really gets influenced by the offer info and then make a purchase. To be able to find this information, some tedious data processing needs to be done. In this case, we developed a workflow to recognize the fact that an user is really affected by the offer info rather than just making a random or regular purchase.

The way we define this workflow is to come up with the process going from offer received, offer viewed, and offer completed. In addition to this, we also want to make sure this entire process is done within the valid duration of each offer, which means that the time between offer received and offer completed should be smaller than the offer duration.

As a result, we define a custom function named "user_offer_completion_count()" to find out how many valid completed offers made by each consumer for a specific offer. Several examples are provided as the screenshots below.

Out[427]:

	duration	offer_type	event	person	time	offer_id
143202	5	bogo	offer received	004c5799adbf42868b9cff0396190900	408	f19421c1d4aa40978ebb69ca19b0e20d
143288	5	bogo	offer viewed	004c5799adbf42868b9cff0396190900	408	f19421c1d4aa40978ebb69ca19b0e20d
144407	5	bogo	offer completed	004c5799adbf42868b9cff0396190900	432	f19421c1d4aa40978ebb69ca19b0e20d
146550	5	bogo	offer received	004c5799adbf42868b9cff0396190900	504	f19421c1d4aa40978ebb69ca19b0e20d
147320	5	bogo	offer viewed	004c5799adbf42868b9cff0396190900	516	f19421c1d4aa40978ebb69ca19b0e20d
148247	5	bogo	offer completed	004c5799adbf42868b9cff0396190900	558	f19421c1d4aa40978ebb69ca19b0e20d

In [428]:
```
user_offer_completion_counts(tt)
```

Out[428]: 2

```
t = tmp_summary_df.groupby(['person', 'offer_id'])
# tt = t.get_group(('004c5799adbf42868b9cff0396190900', 'f19421c1d4aa40978ebb69ca19b0e20d'))
tt = t.get_group(('005500a7188546ff8a767329a2f7c76a', 'ae264e3637204a6fb9bb56bc8210dddfd'))
# tt = t.get_group(('00715b6e55c3431cb56ff7307eb19675', '0b1e1539f2cc45b7b9fa7c272da2e1d7'))
tt
```

	duration	offer_type	event	person	time	offer_id
1013	7	bogo	offer received	005500a7188546ff8a767329a2f7c76a	0	ae264e3637204a6fb9bb56bc8210dddfd
2575	7	bogo	offer viewed	005500a7188546ff8a767329a2f7c76a	60	ae264e3637204a6fb9bb56bc8210dddfd
4049	7	bogo	offer received	005500a7188546ff8a767329a2f7c76a	168	ae264e3637204a6fb9bb56bc8210dddfd
5111	7	bogo	offer viewed	005500a7188546ff8a767329a2f7c76a	186	ae264e3637204a6fb9bb56bc8210dddfd
15771	7	bogo	offer received	005500a7188546ff8a767329a2f7c76a	576	ae264e3637204a6fb9bb56bc8210dddfd
16283	7	bogo	offer viewed	005500a7188546ff8a767329a2f7c76a	576	ae264e3637204a6fb9bb56bc8210dddfd

```
user_offer_completion_counts(tt)
```

0

```
t = tmp_summary_df.groupby(['person', 'offer_id'])
# tt = t.get_group(('004c5799adbf42868b9cff0396190900', 'f19421c1d4aa40978ebb69ca19b0e20d'))
# tt = t.get_group(('005500a7188546ff8a767329a2f7c76a', 'ae264e3637204a6fb9bb56bc8210dddfd'))
tt = t.get_group(('00715b6e55c3431cb56ff7307eb19675', '0b1e1539f2cc45b7b9fa7c272da2e1d7'))
tt
```

	duration	offer_type	event	person	time	offer_id
67055	10	discount	offer received	00715b6e55c3431cb56ff7307eb19675	168	0b1e1539f2cc45b7b9fa7c272da2e1d7
68086	10	discount	offer completed	00715b6e55c3431cb56ff7307eb19675	210	0b1e1539f2cc45b7b9fa7c272da2e1d7
68256	10	discount	offer viewed	00715b6e55c3431cb56ff7307eb19675	228	0b1e1539f2cc45b7b9fa7c272da2e1d7
75980	10	discount	offer received	00715b6e55c3431cb56ff7307eb19675	576	0b1e1539f2cc45b7b9fa7c272da2e1d7
77735	10	discount	offer completed	00715b6e55c3431cb56ff7307eb19675	666	0b1e1539f2cc45b7b9fa7c272da2e1d7

```
user_offer_completion_counts(tt)
```

0

Finally, after we computed the valid offer completion counts for each user and merge with each user's profile attribute, we obtained our final data frame for model training. The final data frame can be viewed as below.

```
trimmed_df.head()
```

Customer	ae264e3637204a6fb9bb56bc8210dddfd	4d5c57ea9a6940dd891ad53e9dbe8da0	3f207df678b143eea3cee63160fa8bed	9b98b8c7a33
0610b486422d4921ae7d2bf64640c50b	0	0	0	
78afa995795e4d85b5d9ceeca43f5fef	1	0	0	
e2127556f4f64592b11af22de27a7932	0	0	0	
389bc3fa690240e798340f5a15918d5c	0	0	0	
2eeac8d8feae4a8cad5a6af0499a211d	0	0	0	

c. Refinement

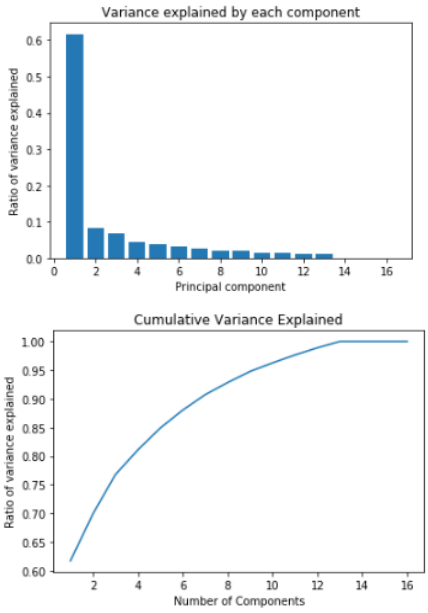
A final refinement to make sure all the data in the final data frame are valid and not skewed. We applied the MinMaxScaler to all the points so that when we run a distance-based algorithm such as K-means, data points with large range will not distort the prediction result.

```
In [357]: scaler = MinMaxScaler()
          new_model_df = scaler.fit_transform(trimmed_df)

In [358]:
          new_model_df = pd.DataFrame(new_model_df)
          new_model_df.columns = trimmed_df.columns.values

In [359]: new_model_df.shape
Out[359]: (14825, 16)
```

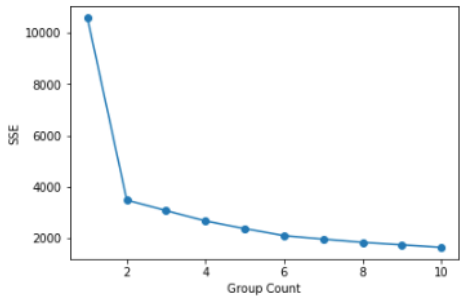
Results

a. Model Evaluation and Validation

Based on our PCA model prediction result, we found out that the 1st component actually can explain the majority of variance. As result, it can estimated that the total number of the component we will need to the model prediction shouldn't be very large. As predicted, the second chart below indicates that we only need 7 components in order to meet our benchmark of 90% of explained variance.

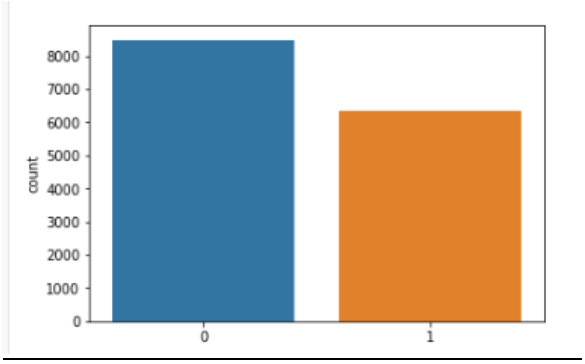


Thus, we chose the top 7 components for our K-means model. With that, we found out when we have 2 components, the SSE (Sum of Squared Error) reduced sharply and forms the ankle in the elbow line in the chart. As a result, we chose the value of 2 to be number of clusters.



b. Justification

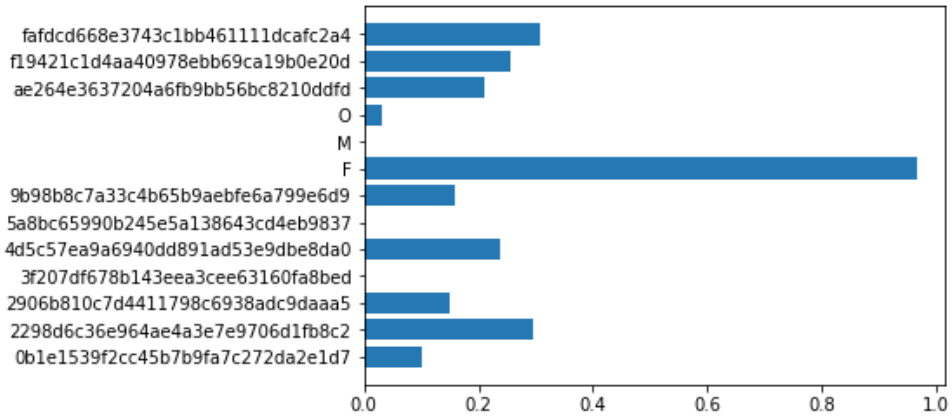
Now we have obtained our groupings for all users. It is time for us to evaluate the credibility of the model prediction. From the bar chart below. We can easily find out that our data points are quite evenly distributed among these two groups. This is quite promising as we certainly don't want to see the majority of points only cluster in one of the groups.

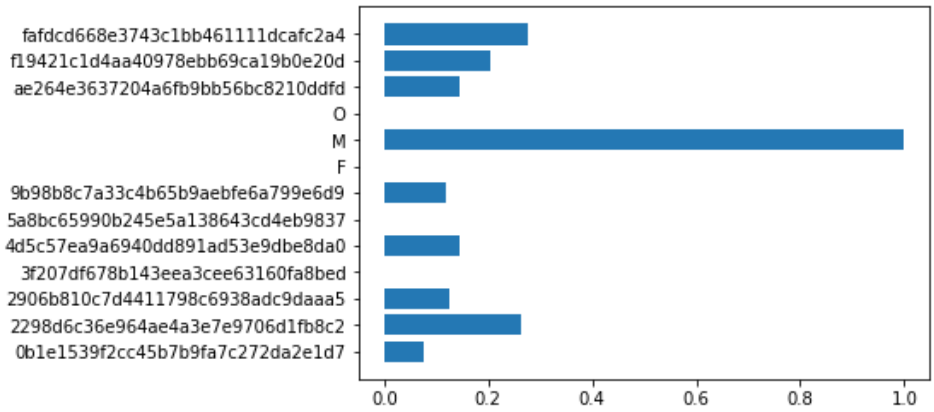


Further taking a look into the composition of each cluster, we found out the sole feature that distinguishes the two clusters is gender. Other than the gender feature, other features basically have the similar impacts on the data points distribution. In addition to that, we also found out that the two clusters actually share similar patterns in terms of the preferred offers they received.
The top 3 offers both groups share are: **fafdcd668e3743c1bb461111dcafc2a4, f19421c1d4aa40978ebb69ca19b0e20d, 2298d6c36e964ae4a3e7e9706d1fb8c2**.

1. **fafdcd668e3743c1bb461111dcafc2a4 (discount)**
2. **f19421c1d4aa40978ebb69ca19b0e20d (bogo)**
3. **2298d6c36e964ae4a3e7e9706d1fb8c2 (discount)**

From the these offers, we can find out that the preferred channels the customers use to access the offer info are: **web, email, mobile, social**





As a result, from the prediction result, we thus are able to answer all the questions we previously defined.

- How offer types vary from each other based on different pairs of features?
- Check out the distribution of users who have consumed offers
- To see if users can be classified by offer ID
- To see if users can be classified by offer Type