Capstone Final Report

1 Optimizing App Offers With Starbucks

A Capstone Proposal for Udacity's Machine Learning Nanodegree

Prepared By:
Author: Aaron McUmber, Ph.D.
Submission Date: 12 June 2022

Prepared For:
Udacity
Machine Learning Engineer Nanodegree

1.1 Definition

1.1.1 Project Overview

Marketing new products and offers is a difficult and critical aspect to any business endeavor. Well established companies such as Starbucks regularly require new marketing strategies in order to maintain their customer loyalty and maintain an advantage ahead of their competition.

Targeting customers with new offers and promotions can have mixed results depending on the customer's preferences and interests. While some promotions may encourage increased attendance, other promotions to the same customer base may not elicit strong responses, or perhaps elicit negative responses. Building advertising models that are careful to consider customer pervious habits and demographic information can produce tailored recommendations to deliver the most effective promotion strategy in an effort to maximize customer loyalty.

The business case for building effective marketing models are clear, additional technological implications about time-phased recommendations presents an interesting challenge to build effective models where a recommendation may differ depending on timing in addition to a customer's profile.

1.1.2 Problem Statement

The analysis for this final report has developed predictive models that can predict if a BOGO offer will be effective provided BOGO offer details as well as customer profile data.

1.1.3 Metrics

The models generated will be evaluated using accuracy and against a DummyClassifier that will serve as a baseline accuracy. The DummyClassifier will select the most frequent class regardless of inputs. This will provide insight to any applied bias to the datasets.

1.2 Analysis

1.2.1 Data Exploration

Observing the raw dataset there are several sets that need to be cleaned before any data can be derived. First from the portfolio data, channels has multiple entries. These values will be expanded to one-hot elements.

`Portfolio`

	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

The transcript data has nested data within the value column dependent on event type. These values will be expanded as well.

`Transcript`

	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

The profile data has imputed values for NaNs for age (118) and the became_member_on data will need to be converted to datetime values.

`Profile`

	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

Upon cleaning the data the following data sets are shown below:

`Portfolio Cleaned`

	offer_reward	offer_difficulty	offer_type	id	web	email	mobile	social	offer_duration
0	10	10	bogo	ae264e3637204a6fb9bb56bc8210ddfd	1	1	1	1	168.0
1	10	10	bogo	4d5c57ea9a6940dd891ad53e9dbe8da0	1	1	1	1	120.0
2	0	0	informational	3f207df678b143eea3cee63160fa8bed	1	1	1	0	96.0
3	5	5	bogo	9b98b8c7a33c4b65b9aebfe6a799e6d9	1	1	1	0	168.0
4	5	20	discount	0b1e1539f2cc45b7b9fa7c272da2e1d7	1	1	0	0	240.0

`Profile Cleaned`

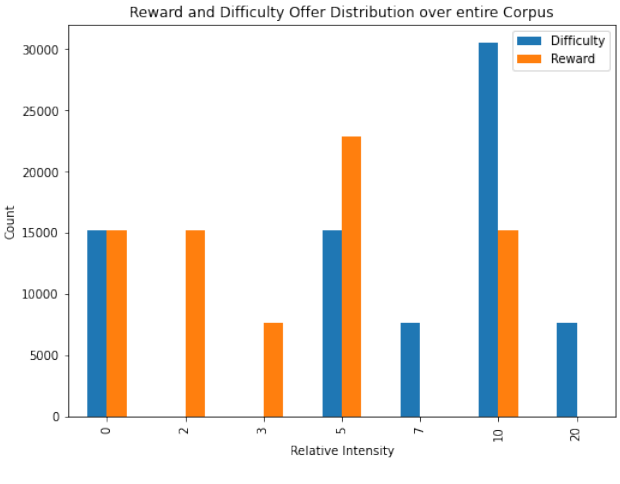
	gender	age	id	became_member_on	income
0	NaN	NaN	68be06ca386d4c31939f3a4f0e3dd783	2017-02-12	NaN
1	F	55.0	0610b486422d4921ae7d2bf64640c50b	2017-07-15	112000.0
2	NaN	NaN	38fe809add3b4fcf9315a9694bb96ff5	2018-07-12	NaN
3	F	75.0	78afa995795e4d85b5d9ceeca43f5fef	2017-05-09	100000.0
4	NaN	NaN	a03223e636434f42ac4c3df47e8bac43	2017-08-04	NaN

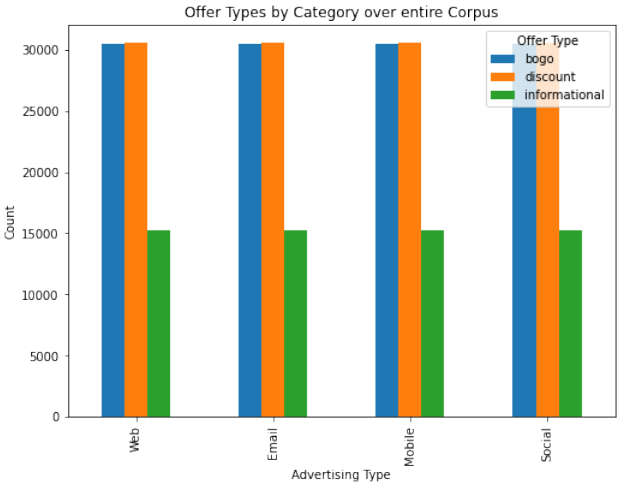
`Transcript Cleaned`

	person	event	time	offer_id	reward	amount
0	78afa995795e4d85b5d9ceeca43f5fef	offer received	0	9b98b8c7a33c4b65b9aebfe6a799e6d9	NaN	NaN
1	a03223e636434f42ac4c3df47e8bac43	offer received	0	0b1e1539f2cc45b7b9fa7c272da2e1d7	NaN	NaN
2	e2127556f4f64592b11af22de27a7932	offer received	0	2906b810c7d4411798c6938adc9daaa5	NaN	NaN
3	8ec6ce2a7e7949b1bf142def7d0e0586	offer received	0	fafdcd668e3743c1bb461111dcafc2a4	NaN	NaN
4	68617ca6246f4fbc85e91a2a49552598	offer received	0	4d5c57ea9a6940dd891ad53e9dbe8da0	NaN	NaN

1.2.2 Exploratory Visualization

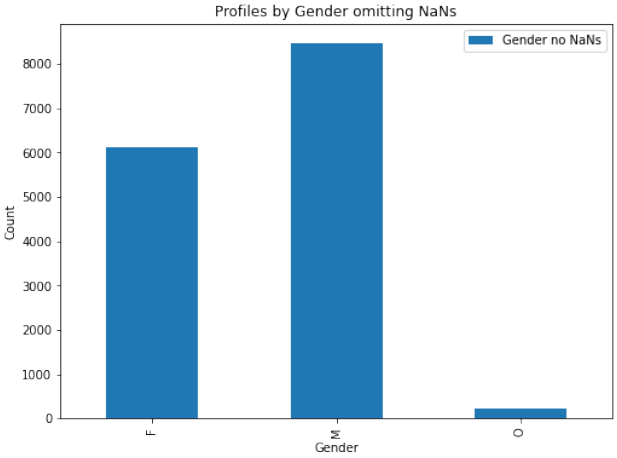
Once the data has been cleaned, visualizing the data will be key to help build feature data for the intended models. First, observing offer types across the corpus.

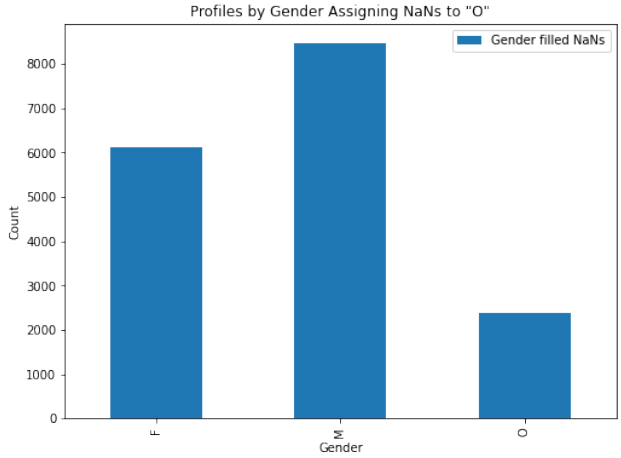




The offers extended appear to have been constructed such BOGO and discounts are extended twice as frequently as the informational offers. Additionally, difficulty and reward intensities appear to have a uneven distribution distribution.

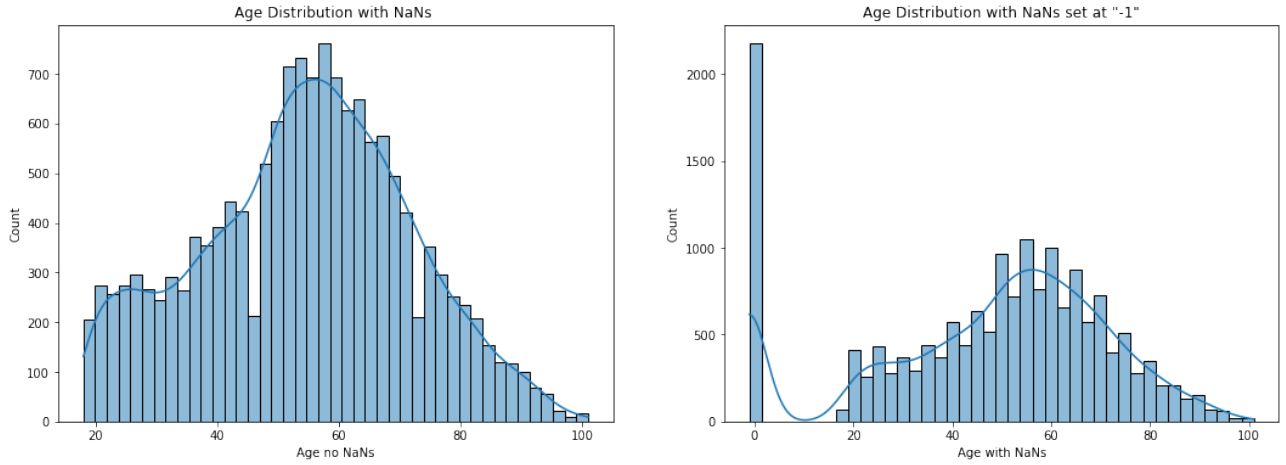
Reviewing NaNs within the profile data it appears that there is a significant population of NaNs across Gender, Age, and Income. Moreover, there is significant skew in gender where male is the most common followed by female and other is a significant minor population. When imputing nans into the other population it is apparent the population of profiles that do not have gender information is much larger than the other population.



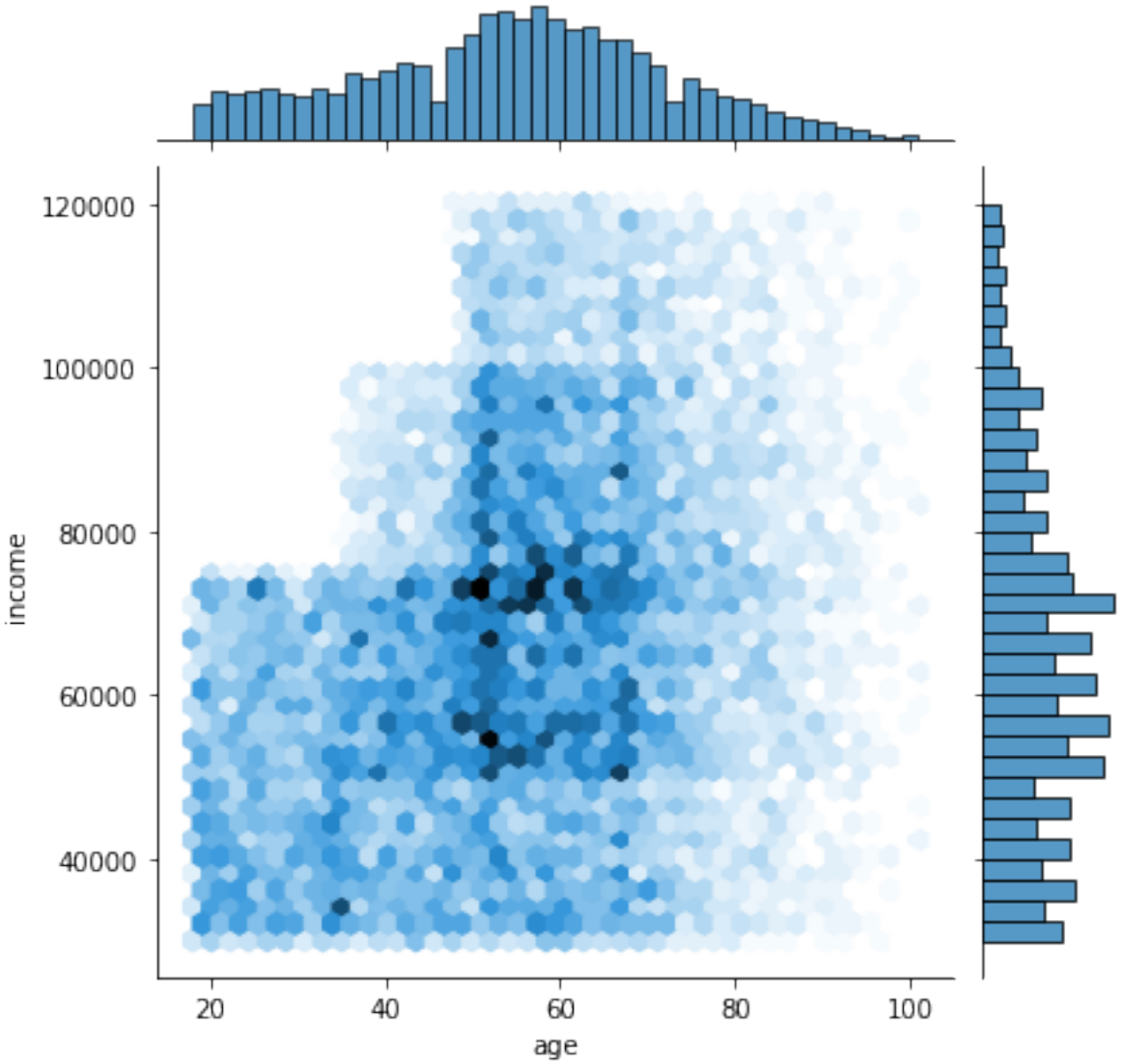


When reviewing age distribution, there appears to be a wide-skewed distribution with large populations in the

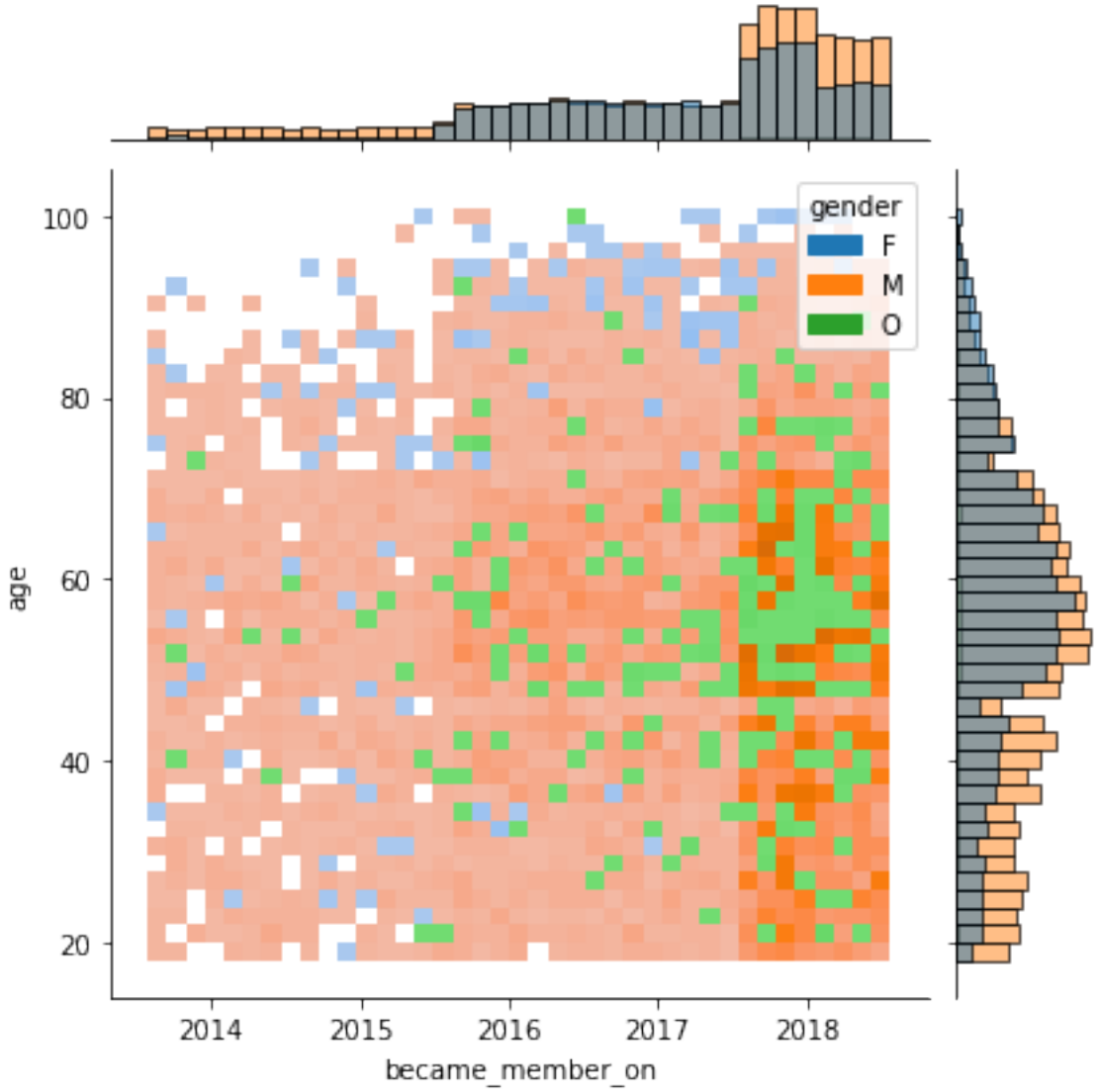
mid-20s to early-30s and in the 60s. When reviewing nans. Again a large population of profiles that do not provide an age.



When comparing age and income there seems to be stratified income levels depending on age.



When comparing membership and age distribution there appears to be an even distribution. However, when comparing to gender, other gender appears to be much more popular with the newer members, but distributed across age.



1.2.3 Algorithms and Techniques

The primary objective of this project is to build a best classifier using either a custom fully-connected neural network or a machine learning model from the Scikit-learn toolbox.

The cleaned data was then process to generate an offer success metric which was used as the predictive target for the classifiers predicting if an offer was or was not successful. Dependent on customer interaction with the offer.

Prior to feeding the classifier model principal component analysis (PCA) was performed to analyze the input vectors and compress the vector state to while maintaining maximum variance within the model. The input vectors were also scaled using a standard scaler and any nans are imputed as the median value for each vector.

Classifiers were selected across a variety of machine learning models and a custom neural network. The models selected were

The neural network has been implemented using pytorch. Three hidden layers of 256 nodes have been selected and batch normalization and dropout was inserted between the input portions of each hidden layer. The Adam optimizer was used optimize the neural network and Cross Entropy Loss was used to evaluate losses. Many of these choices were determined by best practices and familiarity to the algorithms, in order to provide the most effective model without overt optimization. This was an engineering decision in order to quickly evaluate against the scikit-learn model implemented as well.

The scikit-learn models were selected in order of complexity were Logistic Regression, Decision Tree, Gaussian Naive Bayes, Random Forrest, Gradient Boost, Ada Boost, and a Fully Connected Neural Network. Aside from the neural network all models were implemented using scikit-learn and the neural network was implemented using pytorch. Finally, the best scikit-learn model was optimized using scikit's `RandomizedSearchCV` model in investigate potential optimizations.

Evaluation is conducted using accuracy and comparison to the baseline model, mentioned below. F1 Score, precision, and recall were considered, especially given the observed skew in the dataset. However, though skew was observed, considering optimization above the benchmark model provided enough information to evaluate the models. Though, if the dataset had greater skew (e.g. 90% baseline accuracy) additional methods would have been implemented.

1.2.4 Benchmark

The baseline classifier used for model evaluation and comparison was scikit-learn's `DummyClassifier` using the most frequent strategy. This baseline classifier disregards inputs and selects only the most frequent label. This classifier was selected as a method to identify any slight bias into the dataset and methods to adjust for such bias. Note, this strategy does not work well for extreme bias and other methodologies should be explored. Though given the observed bias on the order of 60%, identifying the bias fo the data set was sufficient for the model performance analysis.

1.3 Methodology

1.3.1 Data Preprocessing

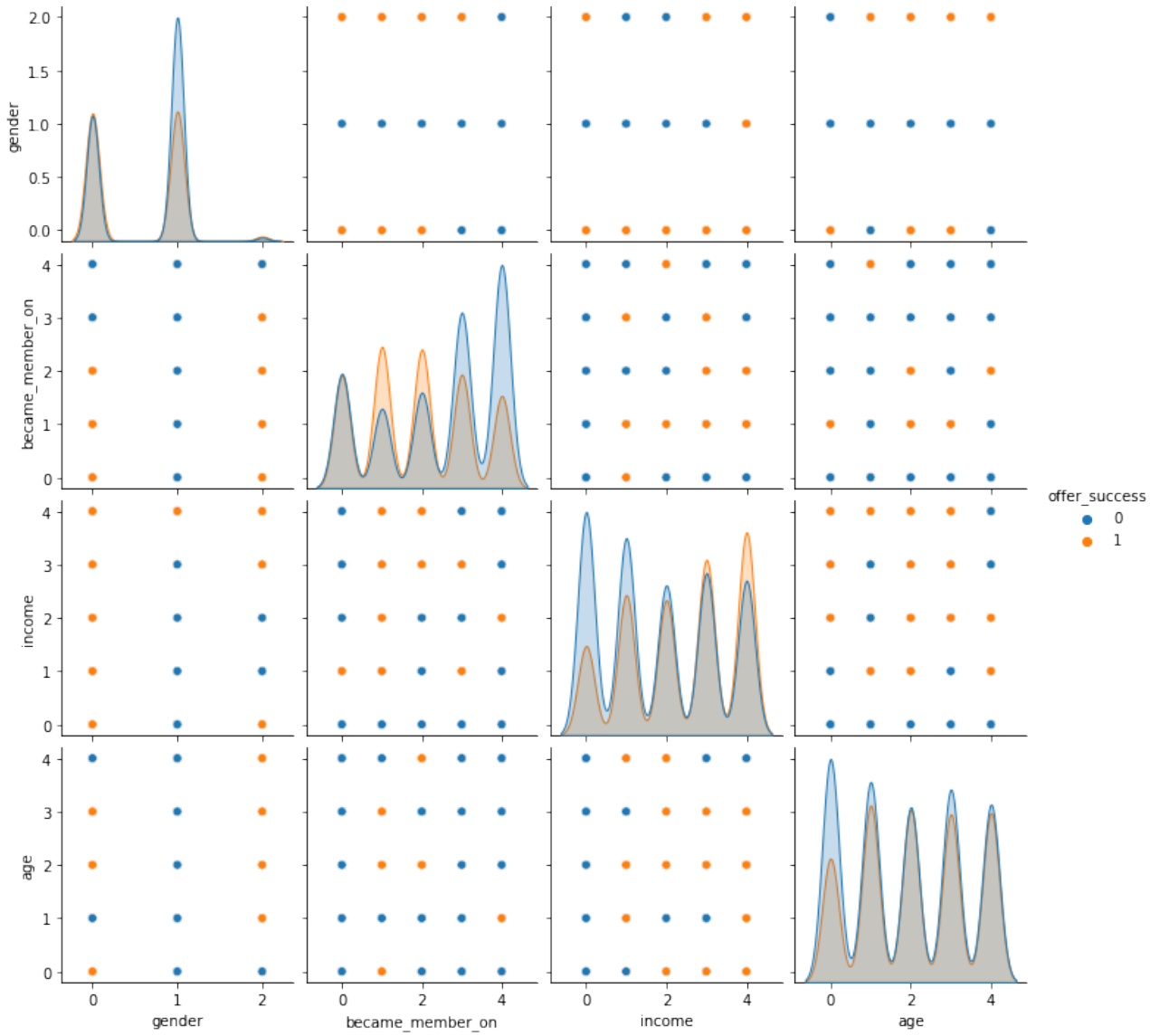
After cleaning the json data set into dataframes with primitive data types (int, float, string, category, etc.) additional features are required to generate before feeding the data into a model. First, the transcript dataset contains information about each individual atomistic event. These atomistic events need to be converted to each offer duration and collect metrics of each offer. These features are generated by the addition of some additional fields. Once these fields are generated, the dataset can be condensed into offer events utilizing key information using the following fields:

1. `event_id`: This is a unique id generated for each unique offer period started by an "Offer Received" event
2. `sales`: This is the amount of sales generated for each `event_id`
3. `cost`: This is the cost incurred by the `event_id` (i.e. if a discount was applied to a sale - the amount of the discount was captured here - a.k.a the reward of the offer)
4. `profit`: This is sales - cost for each `event_id`
5. `offer_viewed`: This is marked true for each item after the "Offer Viewed" event
6. `offer_valid`: This is marked true for each event less than the offer duration (defined by `elapsed_time` column)
7. `offer_redeemed`: This is marked true only at events labeled as "Offer Completed" - intended to be used to generate `offer_success` column
8. `offer_success`: This is marked true if and only if `offer_viewed`, `offer_valid`, and `offer_redeemed` are all true
9. `gender`: This column has been assigned a numerical category where -1 is nan, 0 is male, 1 is female, 2 is other
10. `became_member_on`: This column has been converted to a numerical category where each enumeration represents an equal distribution cut of 5 separate categories where 1 are the most loyal customers and 5 are the newest customers. 0 is NaN
11. `age`: This column has been converted to a numerical category where each enumeration represents an equal distribution cut of 5 separate categories where 1 are the youngest customers and 5 are the oldest customers. 0 is NaN
12. `income`: This column has been converted to a numerical category where each enumeration represents an equal distribution cut of 5 separate categories where 1 are the lowest income customers and 5 are the most affluent customers. 0 is NaN
13. `offer_start`: this column is filled with the value of when the offer was started for all offers within the `event_id`
14. `elapsed_time`: this column is the time elapsed after the start of the event_id defined by `offer_start`

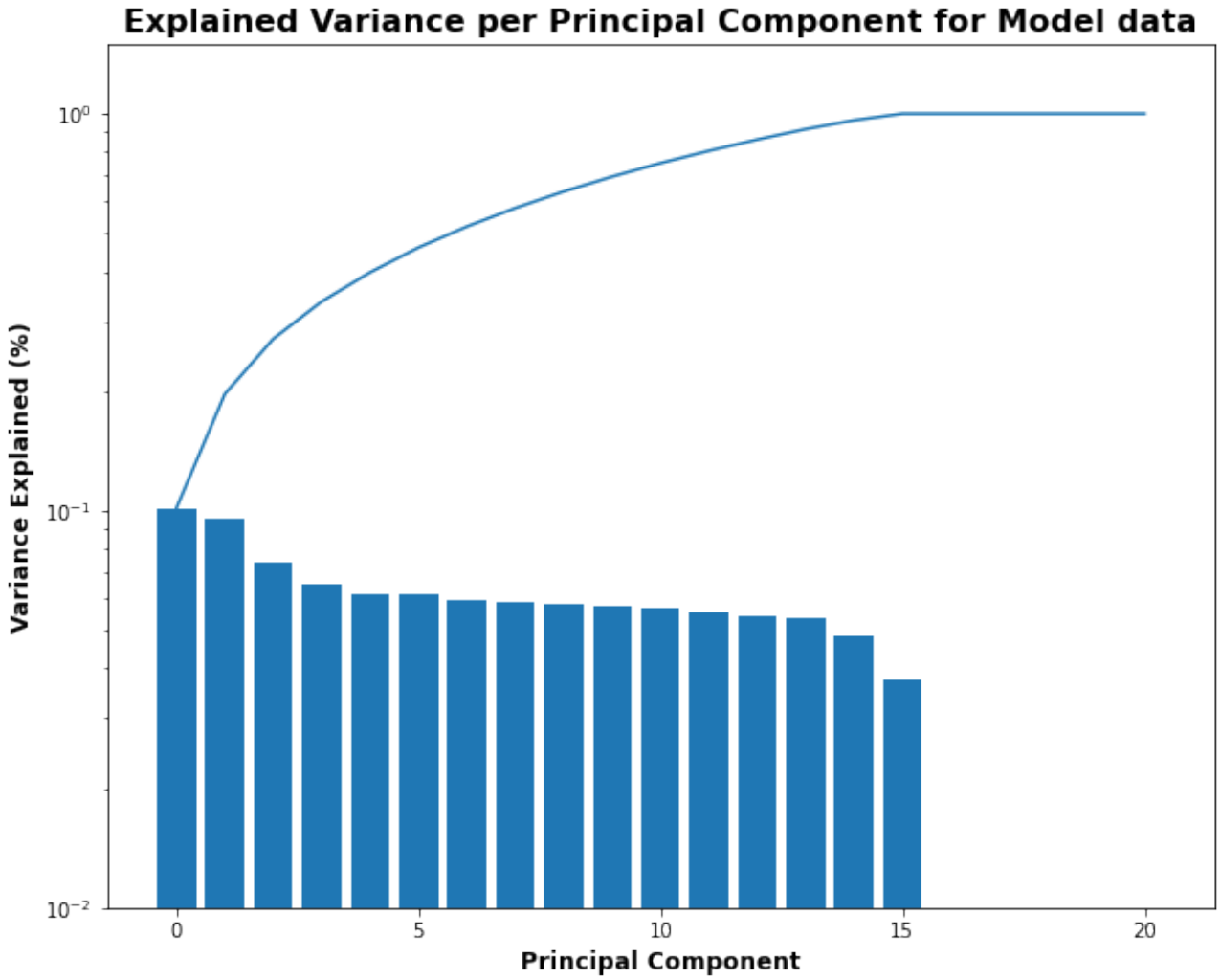
`profit` was originally used as my target value to generate a `best_offer` category, but was ultimately scrapped. `offer_success` is ultimately used to evaluate given portfolio and profile information if an offer will be successful to the individual.

1.3.2 Refinement

Upon filtering the data the offer type appears to have different response metrics, unfortunately. Applying predictive measurements using the entire corpus yields inexact models. Upon reviewing additional data correlations the BOGO offer type appears to have strong correlations with the input vectors.



Filtering the dataset to BOGO offers provided improved results. Though, PCA analysis suggests that some vectors may be condensed into a smaller set of composite vectors while maintaining similar explained variance.



Given explained variance, a reduction the number of components by 40% retains similar variance and reduces the number of input vectors. Therefore, the final PCA model used will contain 15 components to reduce the number of input vectors into the classifiers.

1.3.3 Implementation

Several implementation strategies have been utilized to achieve model analysis. First, the algorithms implemented to clean the dataset and generate the features for model ingestion were developed using a factory pattern. The classifier models were built alongside an associated trainer class that was developed using a facade archetype such that, if necessary, a common API between the scikit-learn and pytorch models could be interfaced with via the trainer class. Moreover, the `Pipeline` class was heavily leveraged to feed data into models such that the data is scaled, imputed, and PCA applied prior to model ingestion.

1.4 Results

1.5 Justification

Analyzing the Starbucks customer data is quite challenging. The provided data can be analyzed and manipulated several ways to produce varying degrees of success.

Though the performance of the classifiers were less than anticipated and certainly a much more simple set of models than the design originally proposed. Several performance indicators and engineering decisions were necessary to drive a model capable of performing close to previously reported evaluations. Given additional optimization, greater accuracy could be attained. However, the review of several models may provide additional benchmarks for the readers and additional analysis of the dataset.

The top performing models, the neural network and the gradient boost classifiers demonstrate measurable increased classification performance over the dummy classifier and mark medium performance. This information can still provide useful information and analysis to perform business decisions moving forward. Through analysis decisions can be made to either tailor the next offer performance champaign or provide information towards phasing out under performing offers given the existing analysis.

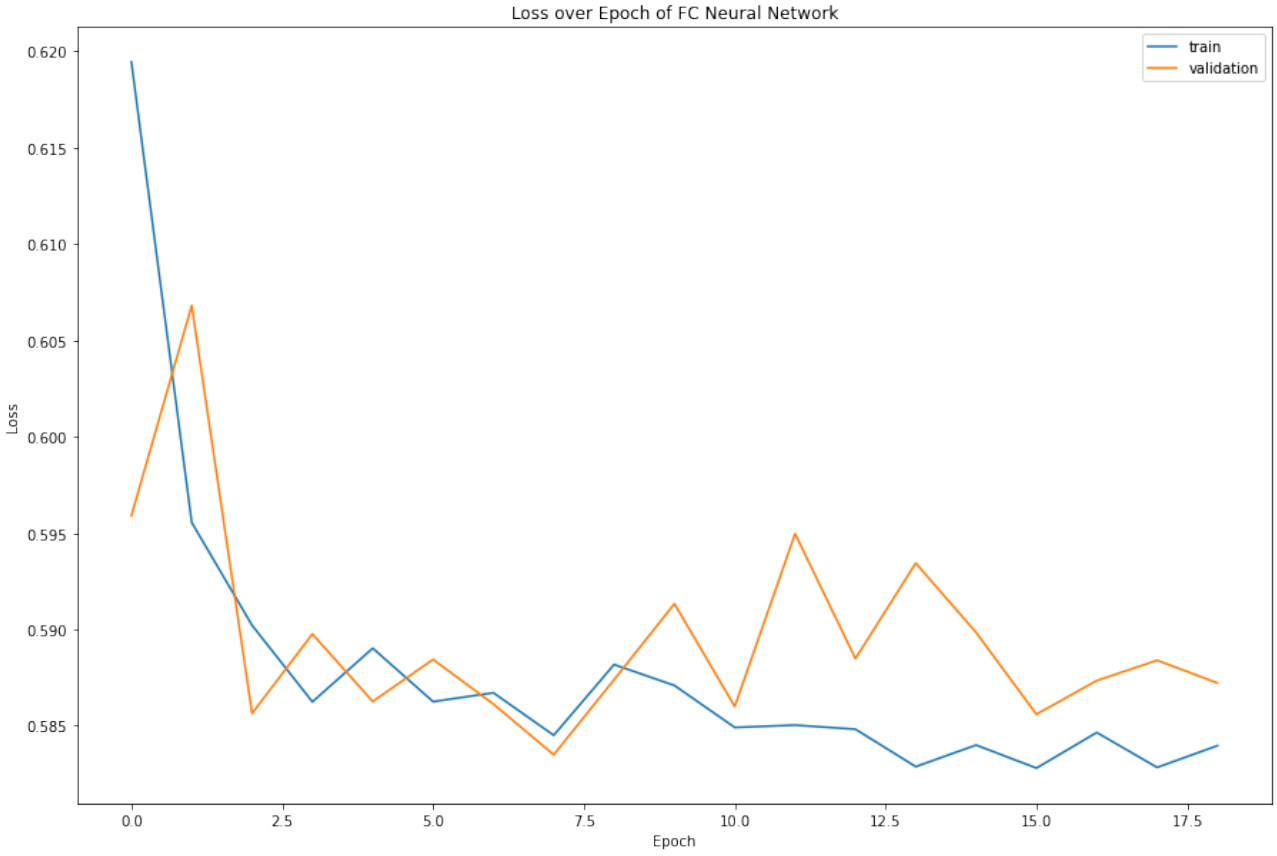
1.6 Model Evaluation and Validation

The datasets were split 60-20-20 into training, validation, testing sets. Due to the nature of the scikit learn model structure, these models were trained on the training set and tested against the test set. The pytorch models were trained on the training set and validated between each epoch against the validation set. These metrics provided evaluation metrics to continue model training or triggering early stopping set at a patience of 10 epochs. Ultimately, the pytorch model was halted at 18 epochs due to an early stopping callback event.

Along with the my original proposal, I first attempted to build models that were capable of predicting the best offer type provided similar data inputs. The best offer was determined by the sum of the sales from the transactions between offer events minus the reward generated by the offer. Unfortunately, this method did not produce any models that performed better than a dummy classifier and the effort was discarded.

A new analysis strategy was adopted after this first failed attempt to analyze offer success. Offer success was defined by an offer that was completed, within the offer validity time, defined by the duration of the offer, and that was viewed by the customer prior to completing the offer. Similar models were tested as before, but upon the entire corpus of data. However, the results of this analysis and model production generated an optimal model able to predict if an offer will be successful with a ~6% accuracy better than a dummy classifier.

Finally, the data was filtered to only BOGO offers (shown in this notebook). Where the two best models, Gradient Boosting and a custom fully connected neural network with batch normalization and dropout, were able to produce similar performance of an 8.2% and 8.4% accuracy better than dummy classifier (test accuracy of 68.57% and 68.76%), respectively. The following figure demonstrates the loss over epoch of the neural network.



As a comparison, I reviewed other posts covering similar data analysis of the Starbucks customer dataset and found that others were able to achieve a similar analysis with model performance closer to 71% using XGBoost. This is may beh worth considering - though I suspect, I may need to clean my data in by some additional means to achieve such performance.