Udacity Machine Learning Engineer Nanodegree Capstone Project

Starbucks App Data Challenge

John Hodge

jah70.udacity@gmail.com

May 31, 2021

Definition

Project Overview

The object of this capstone project is to answer two business challenges using the Starbucks dataset:

I. Can we use data to build a predictive model to determine whether a customer will view and complete a promotional offer based on the offer characteristics and customer demographic information?
II. Can we use offer characteristics and customer demographic data to predict how much revenue a completed offer will generate after subtracting reward costs?

I analyze five different machine learning (ML) predictive model solutions and compare their performance for three case studies. These predictive models allow the business to personalize the characteristics of each promotional offer based on each customer's demographic profile to maximize the probability that the customer views and completes a promotional offer. Additionally, the regression model predicts how much revenue a completed promotional offer generates after subtracting the reward cost to help the business maximize return on investment (ROI) per customer.

Domain Background

Starbucks is the world's largest coffeehouse chain and Fortune 500 company in the United States. The company's headquarters are in Seattle, Washington, where it was founded in 1971. As one of the world's largest companies, Starbucks operates over 30,000 stores and serves millions of customers worldwide.

Starbucks operates a customer rewards program through the Starbucks mobile app to retain customer loyalty and increase business success. The company's mobile app allows registered customers to place pick-up orders, pay inside stores, and earn reward points. In-app marketing through the mobile app is a critical component of Starbucks' direct marketing strategy. Starbucks sends customers promotional offers through the mobile app once every couple of days. These promotions include drink advertisements, discount offers, and buy one get one free (BOGO) offers.

To maximize the effectiveness of these promotional offers, not every customer receives the same promotional offer. Instead, Starbucks tailors promotions and advertisements to the unique characteristics of individual customers and their customer segments. In recent years, machine learning techniques have produced state-of-the-art systems for recommendation [1-2], customer segmentation [3], consumer demand forecasting [4-6], and forecasting consumer behavior

based on promotional marketing [7-8]. This project focuses on using machine learning and data science to predict customer responses to tailored marketing and promotional offers.

As a coffee enthusiast, Starbucks rewards member, and user of the Starbucks mobile app, this capstone project stands out as an exciting data science problem and machine learning investigation. Additionally, learning how a personalized marketing campaign works allows me to understand better how quantitative methods increase customer satisfaction and business success in future engineering projects.

Problem Statement

As stated in the Starbucks' Capstone Challenge overview, the task is to use the data to identify which groups of people are most responsive to each offer and how best to present each type of offer. Data analytics discovers the hidden traits that influence their purchasing decisions and responses to promotional offers for each customer segment in the simulated dataset. In this project, I develop a machine learning model to predict how much a customer will spend based on offer type, demographics, and responses to previous offers.

The goal is to determine what type of advertisement or promotional offer will achieve the highest return on investment (ROI) for a given customer over a set period. The ROI of an ad is the amount a customer spends minus the cost of the promotional discount. The business needs to understand the best type of marketing to serve each customer segment and accurately predict the ROI of each advertisement. Implicitly, this also predicts the people's responsiveness to each offer type as it will have either a positive, negative, or neutral expected value on how much they spend.

Metrics

Classification problem:

For the classification problem (determine whether a customer views and completes a promotion), the precision, accuracy, and recall of the model are used as evaluation metrics to determine which type of offer (discount, BOGO, or informational) is best for each customer. The f1 score, a weighted average of precision and recall, is the primary evaluation metric to determine the best model. Additionally, the area under the receiving operating characteristic (ROC) curve (AUC) is an evaluation metric.

Classification performance metrics:

- **f1 Score**: A weighted average of the precision and recall, where an F1 score reaches its best value at one and worst at 0. The f1 score equation is: f1 = 2 * (precision * recall) / (precision + recall)

- **Accuracy**: Proportion of correct predictions (both true positives and true negatives) among the total number of cases examined. Accuracy is the ratio (*tp* + *tn*)/(*tp* + *tn* + *fp* + *fn*) where *tp* is the number of true positives, *tn* is the number of true negatives, *fp* is the number of false positives, and *fn* is the number of false negatives. The best value is one, and the worst value is zero.
- **ROC AUC**: Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores. The ROC curve plots the true positive rate versus the false positive rate at various thresholds. A value of 1 is best.
- **Precision**: The precision is the ratio *tp* / (*tp* + *fp*). The best value is one, and the worst value is zero.
- **Recall**: The recall is the ratio *tp* / (*tp* + *fn*). The best value is one, and the worst value is zero.

Regression problem:

This project builds a predictive model of how much a customer will spend in response to an advertisement or promotional offer. Since this is a regression problem, the root mean squared error (RMSE) between the amount that the model predicted a customer would spend based on the offer type and how much they spend is the primary metric of model evaluation in this study. The explained variance score and R2 score are additional evaluation metrics under consideration.

Regression performance metrics:

- **RMSE**: The square root of the average squared difference between the estimated values and the actual value. We compute the mean squared error (MSE) as

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

- **Mean Cross-Validation**: Out-of-sample testing technique for assessing how the results of a model will generalize to an unseen data set.
- **R2 Score**: The coefficient of determination (R^2) is a statistical measure of how well the regression predictions approximate the real data points. An R2 score of 1 indicates that the regression predictions perfectly fit the data.
- **Max Error**: Calculates the maximum residual error, which is the most significant single point difference between the model prediction and the actual data point.
- **Explained Variance**: Explains the dispersion of errors of a given dataset. The best possible score is 1.0; lower values are worse.

Analysis

Data Exploration

The structure of the dataset provided Starbucks Capstone project notebook is structured as follows. Three files contain the data:

- portfolio.json - containing offer ids and metadata about each offer (duration, type, etc.) (See Fig. 1)
- profile.json - demographic data for each customer (See Fig. 2)
- transcript.json - records for transactions, offers received, offers viewed, and offers completed (See Fig. 3)

The three types of offers presented in the *_type* column of *portfolio.json* are:

- Buy-one-get-one (BOGO): a user needs to spend a certain amount to get a reward equal to that threshold amount.
- Discount: a user gains a reward equal to a fraction of the amount spent.
- Informational offer: there is no reward, but neither is there a required amount that the user is expected to spend.

Here is the schema and explanation of each variable in the files:

portfolio.json
Size: 10 offers by six fields

- *id* (string) - offer id
- *offer_type* (string) - type of offer, i.e., 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)

```
In [2]: portfolio.head(10)
```

Out[2]:

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

```
In [3]: print("Portfolio Data Dimensions: ", portfolio.shape)
Portfolio Data Dimensions:  (10, 6)
```

Figure 1: The first ten rows and the dataset dimensions of *portfolio.json*.

profile.json

Size: 17,000 users by five fields

- *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

```
In [4]: profile.head(10)
```

Out[4]:

	age	became_member_on	gender	id	income
0	118	20170212	None	68be06ca386d4c31939f3a4f0e3dd783	NaN
1	55	20170715	F	0610b486422d4921ae7d2bf64640c50b	112000.0
2	118	20180712	None	38fe809add3b4fcf9315a9694bb96ff5	NaN
3	75	20170509	F	78afa995795e4d85b5d9ceeca43f5fef	100000.0
4	118	20170804	None	a03223e636434f42ac4c3df47e8bac43	NaN
5	68	20180426	M	e2127556f4f64592b11af22de27a7932	70000.0
6	118	20170925	None	8ec6ce2a7e7949b1bf142def7d0e0586	NaN
7	118	20171002	None	68617ca6246f4fbc85e91a2a49552598	NaN
8	65	20180209	M	389bc3fa690240e798340f5a15918d5c	53000.0
9	118	20161122	None	8974fc5686fe429db53ddde067b88302	NaN

```
In [5]: print("Profile Data Dimensions: ", profile.shape)
Profile Data Dimensions:  (17000, 5)
```

Figure 2: The first ten rows and the dataset dimensions of *profile.json*.

transcript.json

Size: 306,534 offers by four fields

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

```
In [6]: transcript.head(10)
```

Out[6]:

	event	person	time	value
0	offer received	78afa995795e4d85b5d9ceeca43f5fef	0	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}
1	offer received	a03223e636434f42ac4c3df47e8bac43	0	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}
2	offer received	e2127556f4f64592b11af22de27a7932	0	{'offer id': '2906b810c7d4411798c6938adc9daaa5'}
3	offer received	8ec6ce2a7e7949b1bf142def7d0e0586	0	{'offer id': 'fafdcd668e3743c1bb461111dcafc2a4'}
4	offer received	68617ca6246f4fbc85e91a2a49552598	0	{'offer id': '4d5c57ea9a6940dd891ad53e9dbe8da0'}
5	offer received	389bc3fa690240e798340f5a15918d5c	0	{'offer id': 'f19421c1d4aa40978ebb69ca19b0e20d'}
6	offer received	c4863c7985cf408faee930f111475da3	0	{'offer id': '2298d6c36e964ae4a3e7e9706d1fb8c2'}
7	offer received	2eeac8d8feae4a8cad5a6af0499a211d	0	{'offer id': '3f207df678b143eea3cee63160fa8bed'}
8	offer received	aa4862eba776480b8bb9c68455b8c2e1	0	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}
9	offer received	31dda685af34476cad5bc968bdb01c53	0	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}

```
In [7]: print("Transcript Data Dimensions: ", transcript.shape)
Transcript Data Dimensions:  (306534, 4)
```

Figure 3: The first ten rows and the dataset dimensions of *transcript.json*.

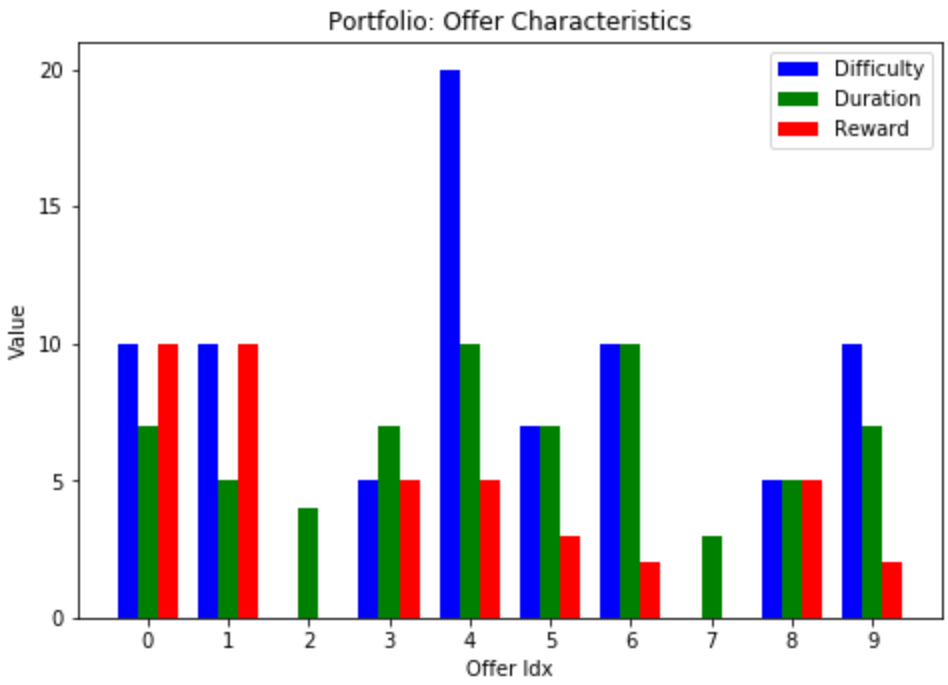
For the forecasting (predict spending) and classification (predict viewed and completed offers) problems, I join the three datasets into a unified dataset that combines the customer profile and offer characteristics with the transcript event and transaction data. Principle component analysis (PCA) creates customer segments based on their customer profiles. For the forecasting problem, the inputs are customer segment id and offer characteristics, and the output is the transaction amount within the offer duration.

Training, validation, and test datasets are created by randomly splitting up "offer received" events in the *transcript.json* dataset. The dataset is augmented by including a column of transaction amounts if a transaction occurs within the offer period. If not transaction occurs, the transaction amount is 0. The dataset will also have a cost column with the reward data from *portfolio.json*. I add a marketing-adjusted revenue column to the *transcript.json* dataset that subtracts the cost of the reward from the non-zero (positive) transaction amounts.

Exploratory Data Analysis and Visualization

Portfolio Dataset

Below is a summary and descriptive statistics of the difficulty, duration, and reward for each of the ten promotional offers.



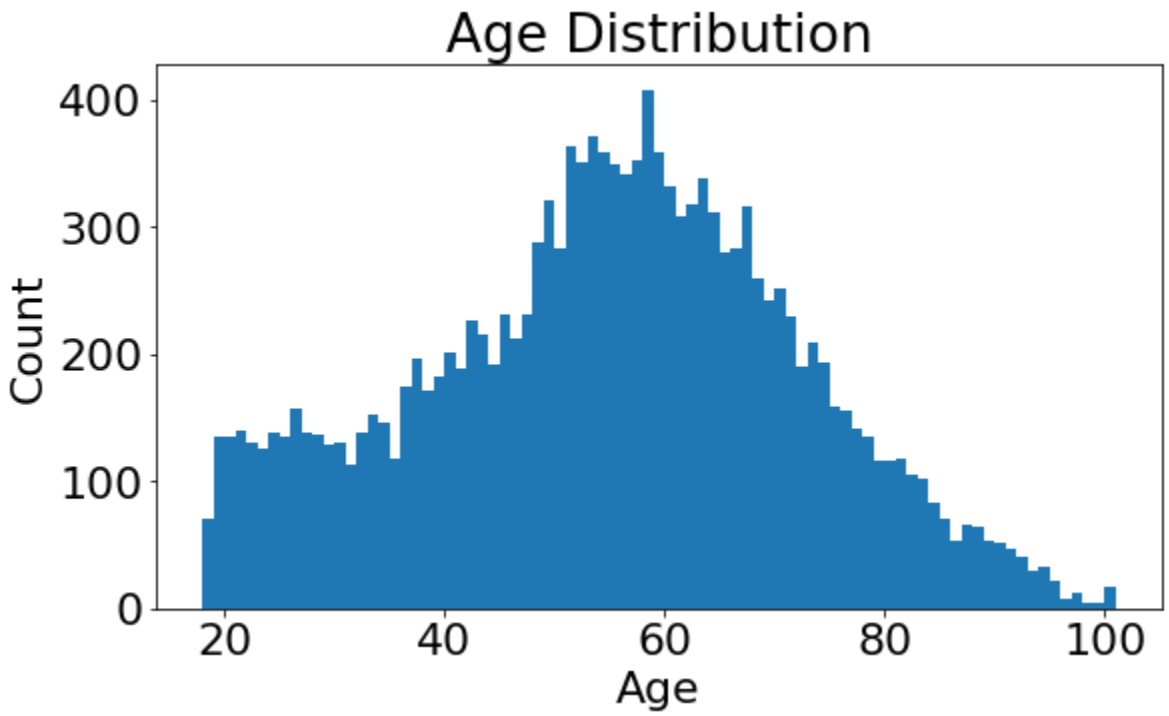
	difficulty	duration	reward
count	10.000000	10.000000	10.000000
mean	7.700000	6.500000	4.200000
std	5.831905	2.321398	3.583915
min	0.000000	3.000000	0.000000
25%	5.000000	5.000000	2.000000
50%	8.500000	7.000000	4.000000
75%	10.000000	7.000000	5.000000
max	20.000000	10.000000	10.000000

Profile Dataset

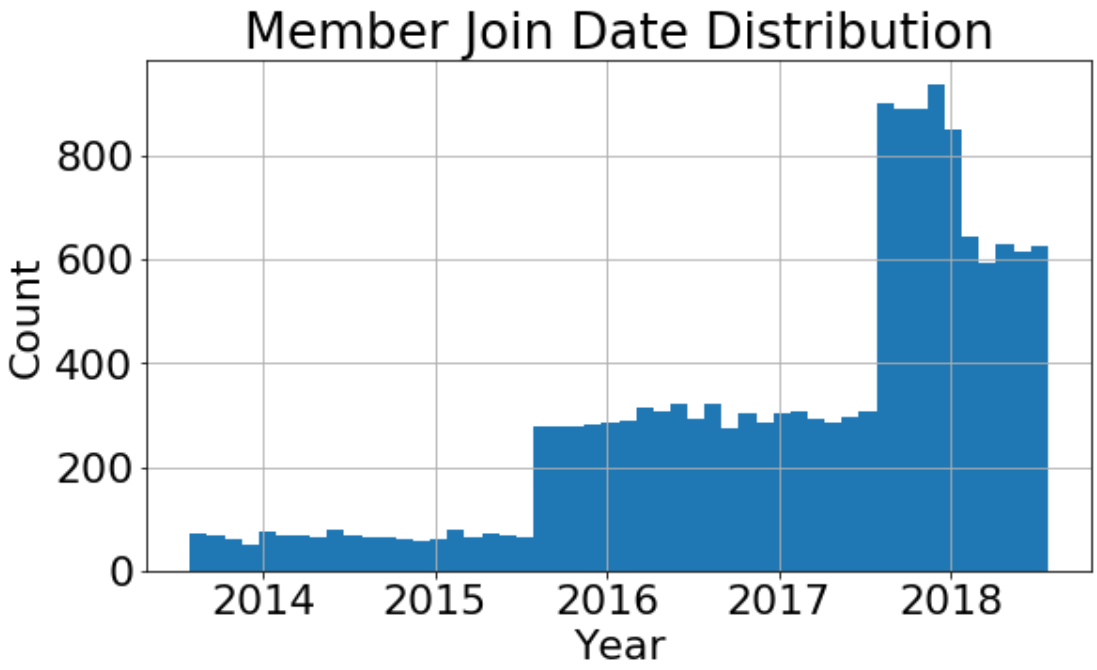
Some of the customer data columns in the profile dataset have missing data. As part of the data cleaning process, we remove customer profiles with missing gender, income, or age data. The number of customers missing data is 2175 out of 17000 customers. After data preparation, we have 14825 customers (rows) in the cleaned profile dataset with seven columns. Shown below are the descriptive statistics for the *profile* dataset.

	age	income	joinDate_month	joinDate_year
count	14825.000000	14825.000000	14825.000000	14825.000000
mean	54.393524	65404.991568	6.695582	2016.620169
std	17.383705	21598.299410	3.488853	1.198245
min	18.000000	30000.000000	1.000000	2013.000000
25%	42.000000	49000.000000	4.000000	2016.000000
50%	55.000000	64000.000000	7.000000	2017.000000
75%	66.000000	80000.000000	10.000000	2017.000000
max	101.000000	120000.000000	12.000000	2018.000000

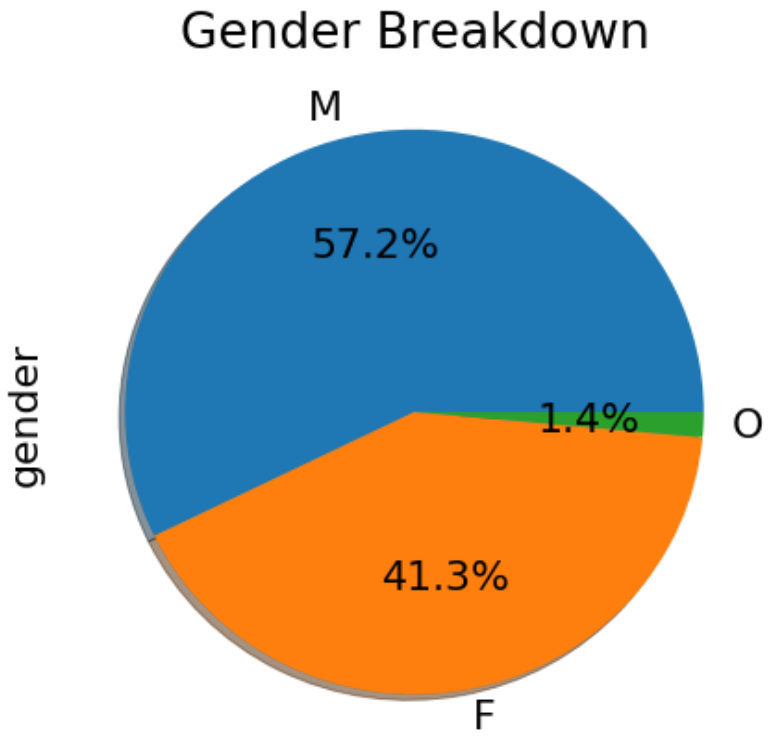
Below is a histogram that shows the distribution of customer ages in the *profile* dataset. The age of the median customer is approximately 55 years old.



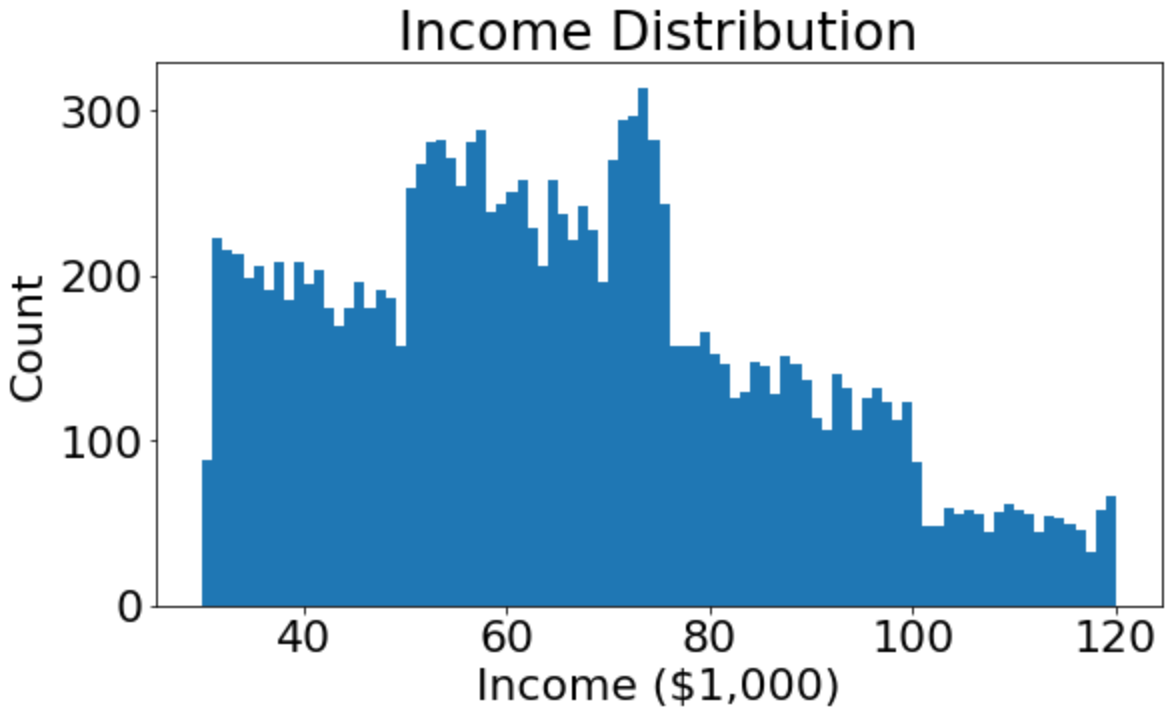
Below is a histogram that shows the distribution of member join dates in the *profile* dataset. The member join date heavily skews towards recency.



Below is a pie chart that shows the distribution of genders in the *profile* dataset. Males are more prevalent than other genders in this dataset.



Below is a histogram that shows the distribution of customer incomes in the *profile* dataset. The median income of customers is approximately $64,000.

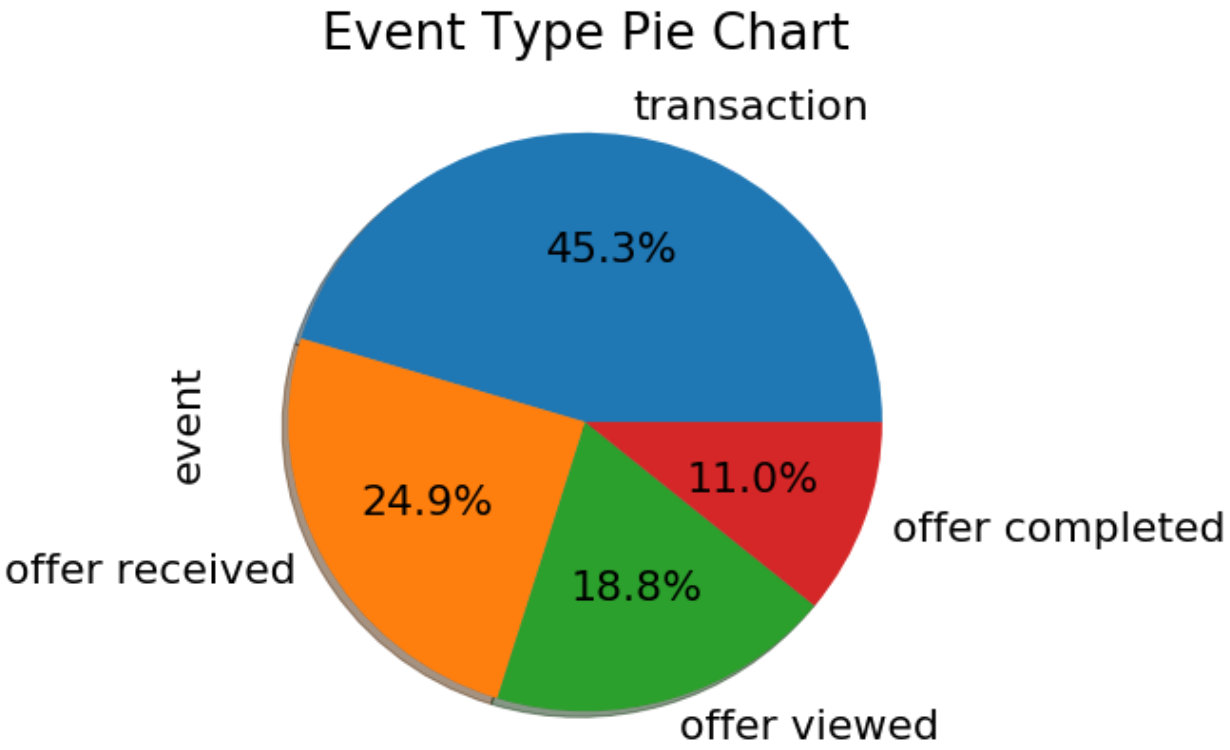


Transcript Dataset

Below are statistics on the *transcript* dataset. A pie chart shows the distribution of event types in the dataset. Transactions are the most common type of event in the dataset, with 45.3% of the entries.

transaction	138953
offer received	76277
offer viewed	57725
offer completed	33579

count	306534.000000
mean	366.382940
std	200.326314
min	0.000000
25%	186.000000
50%	408.000000
75%	528.000000
max	714.000000



event value	offer completed	offer received	offer viewed	transaction
['amount']	0	0	0	138953
['offer id']	0	76277	57725	0
['offer_id', 'reward']	33579	0	0	0

From the table above, we see that:

- *transaction* events result in an *amount* value
- *offer received* and *offer viewed* events result in only an *offer_id* value
- *offer completed* events result in *offer_id* and *reward* values

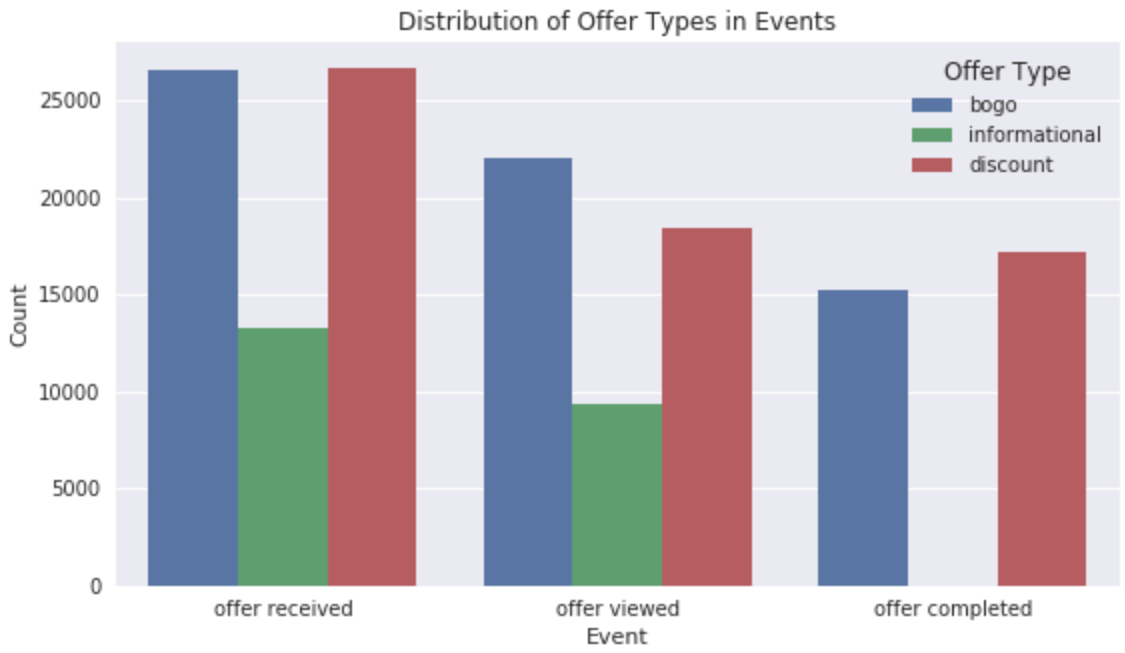


Table: Summary of offers received, viewed, and completed and their aggregate conversion rates in the cleaned dataset.

Offer Type	**Offers Received**	**Offers Viewed**	**Offers Completed**
BOGO	26,537	22,039 (83.1%)	15,258 (57.5%)
Informational	13,300	9,360 (70.4%)	N/A
Discount	26,664	18,461 (69.2%)	17,186 (64.5%)
Total	66,501	49,860 (75.0%)	32,444 (61.0%)*

* Note: Excludes informational offers in percentage.

View transactions for a specific customer

As an example, shown below are the events for a randomly chosen customer in the *transcript* dataset. These events include offers received, offers viewed, offers competed, and transactions.

```
# View examples of a randomly chosen customer
transcript[transcript.person == 'bb0f25e23a4c4de6a645527c275cd594']
```

	event	person	time	value
317	offer received	bb0f25e23a4c4de6a645527c275cd594	0	{'offer id': '2298d6c36e964ae4a3e7e9706d1fb8c2'}
12720	transaction	bb0f25e23a4c4de6a645527c275cd594	0	{'amount': 28.08}
12721	offer completed	bb0f25e23a4c4de6a645527c275cd594	0	{'offer_id': '2298d6c36e964ae4a3e7e9706d1fb8c2...
40812	offer viewed	bb0f25e23a4c4de6a645527c275cd594	96	{'offer id': '2298d6c36e964ae4a3e7e9706d1fb8c2'}
52288	transaction	bb0f25e23a4c4de6a645527c275cd594	162	{'amount': 21.62}
53492	offer received	bb0f25e23a4c4de6a645527c275cd594	168	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}
74855	offer viewed	bb0f25e23a4c4de6a645527c275cd594	186	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}
97732	transaction	bb0f25e23a4c4de6a645527c275cd594	264	{'amount': 35.95}
97733	offer completed	bb0f25e23a4c4de6a645527c275cd594	264	{'offer_id': '9b98b8c7a33c4b65b9aebfe6a799e6d9...
150932	offer received	bb0f25e23a4c4de6a645527c275cd594	408	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}
163478	offer viewed	bb0f25e23a4c4de6a645527c275cd594	408	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}
184965	transaction	bb0f25e23a4c4de6a645527c275cd594	450	{'amount': 25.18}
184966	offer completed	bb0f25e23a4c4de6a645527c275cd594	450	{'offer_id': '0b1e1539f2cc45b7b9fa7c272da2e1d7...

Algorithms and Techniques

We perform customer segmentation using the k-means clustering algorithm and use principal component analysis (PCA) for dimensionality reduction and visualization. For the classification problems, we evaluate the logistic regression (LogReg), support vector machine (SVM), k-nearest neighbor (kNN), XGBoost, and deep neural network (DNN) algorithms. We evaluate the linear stochastic gradient descent (SGD) model, support vector regression (SVR), kNN, XGBoost, and DNN models for regression problems. XGBoost is an open-source and efficient implementation of the gradient boosted tree algorithm. Gradient boosting is a supervised learning algorithm used in one of the course lessons to predict housing price data. I used DNN models extensively using PyTorch in the Deep Learning Nanodegree program.

Benchmark

The benchmark models for this study are logistic regression for the classification problems and linear SGD for the regression model. These are simple well-known learning algorithms with implementations available in scikit-learn that we use for comparison. We also consider the aggregate conversion rate for offers viewed and offers completed as a benchmark metric for the

classification problems. Further, we compare the performance of the SVM, kNN, XGBoost, and DNN models for each of the classification and regression problems.

Methodology

Data Preprocessing

The most challenging portion of the capstone project was the data preprocessing and preparation. The transcript data set includes 306,534 customer events, including 138,953 transactions, 76,277 offers received, 57,525 offers viewed, and 33,579 offers completed. It is easy to perform aggregate statistics on this dataset. However, we need to transform this dataset into more granular data for machine learning and tailored prediction. The raw transactions log does not directly tell us whether an individual offer was viewed or completed and whether completed offers are viewed before or after completion. Additionally, the transcript dataset does not directly tell us how much customers spent during each offering period or how much revenue was gained after adjusting for the cost of the reward amount. Below is an example of the transcript log for a randomly chosen customer with many events.

```
# View examples of a randomly chosen customer
transcript[transcript.person == 'bb0f25e23a4c4de6a645527c275cd594']
```

	person	event	value	time
317	bb0f25e23a4c4de6a645527c275cd594	offer received	{'offer id': '2298d6c36e964ae4a3e7e9706d1fb8c2'}	0
12720	bb0f25e23a4c4de6a645527c275cd594	transaction	{'amount': 28.08}	0
12721	bb0f25e23a4c4de6a645527c275cd594	offer completed	{'offer_id': '2298d6c36e964ae4a3e7e9706d1fb8c2...	0
40812	bb0f25e23a4c4de6a645527c275cd594	offer viewed	{'offer id': '2298d6c36e964ae4a3e7e9706d1fb8c2'}	96
52288	bb0f25e23a4c4de6a645527c275cd594	transaction	{'amount': 21.62}	162
53492	bb0f25e23a4c4de6a645527c275cd594	offer received	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}	168
74855	bb0f25e23a4c4de6a645527c275cd594	offer viewed	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}	186
97732	bb0f25e23a4c4de6a645527c275cd594	transaction	{'amount': 35.95}	264
97733	bb0f25e23a4c4de6a645527c275cd594	offer completed	{'offer_id': '9b98b8c7a33c4b65b9aebfe6a799e6d9...	264
150932	bb0f25e23a4c4de6a645527c275cd594	offer received	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}	408
163478	bb0f25e23a4c4de6a645527c275cd594	offer viewed	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}	408
184965	bb0f25e23a4c4de6a645527c275cd594	transaction	{'amount': 25.18}	450
184966	bb0f25e23a4c4de6a645527c275cd594	offer completed	{'offer_id': '0b1e1539f2cc45b7b9fa7c272da2e1d7...	450
193250	bb0f25e23a4c4de6a645527c275cd594	transaction	{'amount': 24.4}	474
201883	bb0f25e23a4c4de6a645527c275cd594	offer received	{'offer id': 'ae264e3637204a6fb9bb56bc8210ddfd'}	504
221968	bb0f25e23a4c4de6a645527c275cd594	offer viewed	{'offer id': 'ae264e3637204a6fb9bb56bc8210ddfd'}	516
232934	bb0f25e23a4c4de6a645527c275cd594	transaction	{'amount': 19.46}	540
232935	bb0f25e23a4c4de6a645527c275cd594	offer completed	{'offer_id': 'ae264e3637204a6fb9bb56bc8210ddfd...	540

As part of the data transformation and preprocessing, the value column is expanded into amount, offer_id, and reward columns. Additionally, the *time* (in hours) column is converted to the *time_days* (in days) since each offer is specified in days. The data frame below is an example for the same randomly chosen customer.

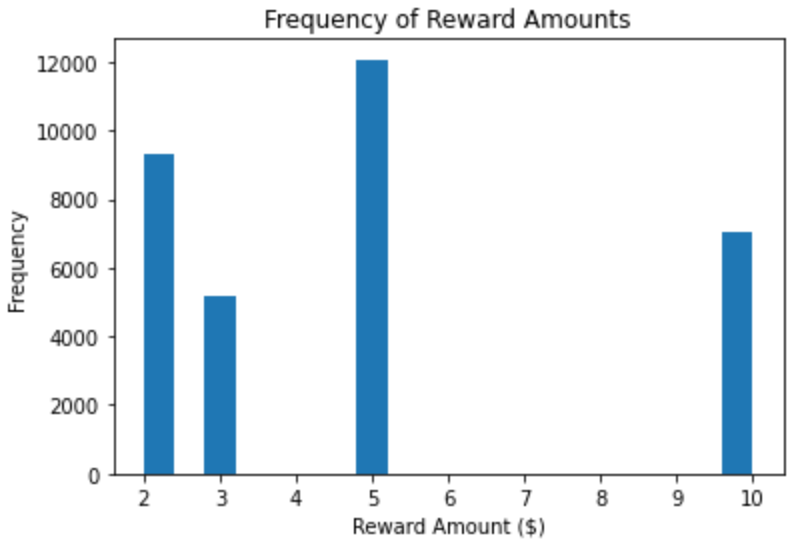
```
# View examples of a randomly chosen customer
transcript[transcript.customer_id  == 'bb0f25e23a4c4de6a645527c275cd594']
```

	customer_id	event	amount	offer_id	reward	time_days
317	bb0f25e23a4c4de6a645527c275cd594	offer received	NaN	2298d6c36e964ae4a3e7e9706d1fb8c2	NaN	0.00
12720	bb0f25e23a4c4de6a645527c275cd594	transaction	28.08	NaN	NaN	0.00
12721	bb0f25e23a4c4de6a645527c275cd594	offer completed	NaN	2298d6c36e964ae4a3e7e9706d1fb8c2	3.0	0.00
40812	bb0f25e23a4c4de6a645527c275cd594	offer viewed	NaN	2298d6c36e964ae4a3e7e9706d1fb8c2	NaN	4.00
52288	bb0f25e23a4c4de6a645527c275cd594	transaction	21.62	NaN	NaN	6.75
53492	bb0f25e23a4c4de6a645527c275cd594	offer received	NaN	9b98b8c7a33c4b65b9aebfe6a799e6d9	NaN	7.00
74855	bb0f25e23a4c4de6a645527c275cd594	offer viewed	NaN	9b98b8c7a33c4b65b9aebfe6a799e6d9	NaN	7.75
97732	bb0f25e23a4c4de6a645527c275cd594	transaction	35.95	NaN	NaN	11.00
97733	bb0f25e23a4c4de6a645527c275cd594	offer completed	NaN	9b98b8c7a33c4b65b9aebfe6a799e6d9	5.0	11.00
150932	bb0f25e23a4c4de6a645527c275cd594	offer received	NaN	0b1e1539f2cc45b7b9fa7c272da2e1d7	NaN	17.00
163478	bb0f25e23a4c4de6a645527c275cd594	offer viewed	NaN	0b1e1539f2cc45b7b9fa7c272da2e1d7	NaN	17.00
184965	bb0f25e23a4c4de6a645527c275cd594	transaction	25.18	NaN	NaN	18.75
184966	bb0f25e23a4c4de6a645527c275cd594	offer completed	NaN	0b1e1539f2cc45b7b9fa7c272da2e1d7	5.0	18.75
193250	bb0f25e23a4c4de6a645527c275cd594	transaction	24.40	NaN	NaN	19.75
201883	bb0f25e23a4c4de6a645527c275cd594	offer received	NaN	ae264e3637204a6fb9bb56bc8210ddfd	NaN	21.00
221968	bb0f25e23a4c4de6a645527c275cd594	offer viewed	NaN	ae264e3637204a6fb9bb56bc8210ddfd	NaN	21.50
232934	bb0f25e23a4c4de6a645527c275cd594	transaction	19.46	NaN	NaN	22.50
232935	bb0f25e23a4c4de6a645527c275cd594	offer completed	NaN	ae264e3637204a6fb9bb56bc8210ddfd	10.0	22.50

Below is a screen capture of the processed *transcript* dataset that includes one-hot encoding columns to information on which offers customers viewed, completed, completed before seen, and the associated transaction and reward amounts.

	customer_id	event	amount	offer_id	reward	time_days	offer_completed	offer_viewed	offer_compViewed	offer_compNotViewed	compTransAmt	rewardReceived	adjRev
12649	e4052622e5ba45a8b96b59aba68cf068	offer received	NaN	2298d6c36e964ae4a3e7e9706d1fb8c2	NaN	0.00	1	1	1	0	21.55	3.0	18.55
18066	e4052622e5ba45a8b96b59aba68cf068	offer viewed	NaN	2298d6c36e964ae4a3e7e9706d1fb8c2	NaN	0.25	0	0	0	0	0.00	0.0	0.00
32623	e4052622e5ba45a8b96b59aba68cf068	transaction	21.55	NaN	NaN	2.25	0	0	0	0	0.00	0.0	0.00
32624	e4052622e5ba45a8b96b59aba68cf068	offer completed	NaN	2298d6c36e964ae4a3e7e9706d1fb8c2	3.0	2.25	0	0	0	0	0.00	0.0	0.00
39543	e4052622e5ba45a8b96b59aba68cf068	transaction	25.19	NaN	NaN	3.50	0	0	0	0	0.00	0.0	0.00
42026	e4052622e5ba45a8b96b59aba68cf068	transaction	21.53	NaN	NaN	4.00	0	0	0	0	0.00	0.0	0.00
123538	e4052622e5ba45a8b96b59aba68cf068	offer received	NaN	3f207df678b143eea3cee63160fa8bed	NaN	14.00	0	0	0	0	0.00	0.0	0.00
163373	e4052622e5ba45a8b96b59aba68cf068	offer received	NaN	f19421c1d4aa40978ebb69ca19b0e20d	NaN	17.00	1	0	0	1	30.57	5.0	25.57
196833	e4052622e5ba45a8b96b59aba68cf068	transaction	30.57	NaN	NaN	20.00	0	0	0	0	0.00	0.0	0.00
196834	e4052622e5ba45a8b96b59aba68cf068	offer completed	NaN	f19421c1d4aa40978ebb69ca19b0e20d	5.0	20.00	0	0	0	0	0.00	0.0	0.00
198464	e4052622e5ba45a8b96b59aba68cf068	transaction	19.47	NaN	NaN	20.25	0	0	0	0	0.00	0.0	0.00
237363	e4052622e5ba45a8b96b59aba68cf068	offer viewed	NaN	f19421c1d4aa40978ebb69ca19b0e20d	NaN	22.75	0	0	0	0	0.00	0.0	0.00
257886	e4052622e5ba45a8b96b59aba68cf068	offer received	NaN	3f207df678b143eea3cee63160fa8bed	NaN	24.00	0	0	0	0	0.00	0.0	0.00
301914	e4052622e5ba45a8b96b59aba68cf068	transaction	24.71	NaN	NaN	28.75	0	0	0	0	0.00	0.0	0.00

We generate the histogram below to visualize the frequency of reward amounts for completed offers.



Our final merged and transformed dataset for model training analyzes 66,501 offers received and has 17 features for each offer received in the dataset. These feature columns cover the offer characteristics (difficulty, duration, reward), channel (social, email, web, mobile), offer type (BOGO, discount, informational), and customer characteristics (age, income, month joined, year joined, and gender). The feature columns used in our training dataset are 'difficulty', 'duration', 'reward', 'chan_social', 'chan_email', 'chan_web', 'chan_mobile', 'offer_type_bogo', 'offer_type_discount', 'offer_type_informational', 'age', 'income', 'joinDate_month', 'joinDate_year', 'gender_F', 'gender_M', 'gender_O'.

	difficulty	duration	reward	chan_social	chan_email	chan_web	chan_mobile	offer_type_bogo	offer_type_discount	offer_type_informational	age	income	joinDate_month	joinDate_year	gender_F	gender_M
0	0.5	0.571429	1.0	1	1	0	1	1	0	0	0.554217	0.777778	0.727273	0.8	0	1
1	0.5	0.571429	1.0	1	1	0	1	1	0	0	0.554217	0.777778	0.727273	0.8	0	1
2	0.0	0.142857	0.0	0	1	1	1	0	0	1	0.554217	0.777778	0.727273	0.8	0	1
3	1.0	1.000000	0.5	0	1	1	0	0	1	0	0.554217	0.777778	0.727273	0.8	0	1
4	1.0	1.000000	0.5	0	1	1	0	0	1	0	0.554217	0.777778	0.727273	0.8	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
66496	0.5	0.571429	0.2	0	1	1	1	0	1	0	0.626506	0.544444	0.909091	0.6	1	0
66497	0.5	0.571429	0.2	0	1	1	1	0	1	0	0.626506	0.544444	0.909091	0.6	1	0
66498	0.5	0.571429	0.2	0	1	1	1	0	1	0	0.759036	0.711111	0.272727	1.0	1	0
66499	0.5	0.571429	0.2	0	1	1	1	0	1	0	0.156627	0.166667	0.545455	0.8	0	1
66500	0.5	0.571429	0.2	0	1	1	1	0	1	0	0.493976	0.155556	0.636364	0.8	0	1

66501 rows × 17 columns

Implementation

I implement all ML models in this project, except for the XGBoost and DNN models, using the open-source scikit-learn library. I implement the XGBoost classifiers and regressors algorithms using the open-source XGBoost Python package and the DNN models using PyTorch.

Below is a screenshot of the XGBoost classifier instance used in this project. Through experimentation, we chose the learning rate as 0.1 and the number of estimators as 500.

```
# XGBoost Classifier
clf = XGBClassifier()

clf.learning_rate = 0.1
clf.n_estimators = 500
```

```
[ ] clf.fit(X_train, Y_train)

XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=500, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
```

Below is the DNN binary classifier used to predict whether an offer was viewed or completed. I used one DNN model for completed offers and another DNN model for viewed offers. Both DNN models used a binary cross-entropy with logits loss (BCEWithLogitsLoss) functions. The Adam optimizer is used with a learning rate (*lr*) of 0.001, which we determine through hyperparameter tuning. I originally experimented with the stochastic gradient descent with momentum optimizer, however, the Adam optimizer achieved significantly better training results.

```
criterion = nn.BCEWithLogitsLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

Below is the netlist of the 16-layer DNN model utilized in this project. There are 17 input features from our merged dataset and one output, which is the probability of whether a customer views or completes the offer, respectively. Layers 2 through 13 are linear layers with 32 inputs and 32 outputs. Layers 14 through 16 taper down from 32 inputs to 1 output node. Layers 1 through 15 use leaky rectified linear unit (Leaky ReLU) activation functions. Layers two through 15 include dropout with a value of 0.25 to improve the robustness of the DNN model.

```
Net(
  (fc1): Linear(in_features=17, out_features=32, bias=True)
  (fc2): Linear(in_features=32, out_features=32, bias=True)
  (fc3): Linear(in_features=32, out_features=32, bias=True)
  (fc4): Linear(in_features=32, out_features=32, bias=True)
  (fc5): Linear(in_features=32, out_features=32, bias=True)
  (fc6): Linear(in_features=32, out_features=32, bias=True)
  (fc7): Linear(in_features=32, out_features=32, bias=True)
  (fc8): Linear(in_features=32, out_features=32, bias=True)
  (fc9): Linear(in_features=32, out_features=32, bias=True)
  (fc10): Linear(in_features=32, out_features=32, bias=True)
  (fc11): Linear(in_features=32, out_features=32, bias=True)
  (fc12): Linear(in_features=32, out_features=32, bias=True)
  (fc13): Linear(in_features=32, out_features=32, bias=True)
  (fc14): Linear(in_features=32, out_features=16, bias=True)
  (fc15): Linear(in_features=16, out_features=8, bias=True)
  (fc16): Linear(in_features=8, out_features=1, bias=True)
  (dropout): Dropout(p=0.25, inplace=False)
```

The DNN model used for the regression problem uses the mean squared error (MSE) loss function. Similar to the classification problem, an Adam optimizer is used with a learning rate (*lr*) of 0.001. This hyperparameter was tuned through parametric study, as shown in the refinement section of this report.

```
criterion = nn.MSELoss()  # this is for regression mean squared loss
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

Refinement

For each model, I tuned the model hyperparameters through experimentation and parametric study to improve model performance. Below is an example of the parametric study that I performed to discover the DNN hyperparameters, resulting in the lowest validation test loss, RMSE, and highest R2 score. The highlighted row is the final DNN model that I chose. I also performed hyperparameter tuning for the k value of the kNN classifier and regressor, as shown in this blog post. Additionally, the learning rate, number of estimators, and objective function of the XGBoost classifier and regressor are tuned to achieve better performance in a reasonable training runtime.

Network Layers	# of Epochs	Batch Size	Learning Rate	Test Loss	R2 Score	Max Error	RMSE
12	100	16	0.005	34.165	0.45	31.59	5.84
12	100	16	0.01	35.03	0.44	30.15	5.92

12	100	32	0.01	33.82	0.46	27.84	5.82
12	250	32	0.01	33.81	0.45	31.98	5.81
16	250	32	0.005	33.47	0.45	26.01	5.79
16	500	64	0.001	32.79	0.46	24.38	5.73
16	**500**	**64**	**0.002**	**32.48**	**0.47**	**29.56**	**5.70**
16	300	64	0.0025	33.46	0.46	24.54	5.78
16	250	64	0.005	33.31	0.45	27.54	5.77
12	100	64	0.01	33.85	0.45	25.05	5.82
12	300	64	0.01	33.61	0.46	28.05	5.80
16	250	64	0.01	35.21	0.43	30.84	5.93
12	100	64	0.02	33.85	0.46	26.44	5.82
12	100	128	0.005	35.55	0.46	32.30	5.88
12	100	128	0.01	34.78	0.45	29.82	5.90
12	100	256	0.01	35.01	0.43	31.93	5.92

Results

Model Evaluation and Validation

Customer Segmentation

We utilize the *k*-means clustering algorithm for customer segmentation in the profile dataset. The feature data in the profile dataset is cleaned and normalized before *k*-means clustering. We implement the elbow method to determine the suitable number of customer segments (value of *k*). Noticeable inflection points in the distortion curve occur at $k = 2$ and $k = 4$. For this study, we choose $k = 4$ to get more detail into our distinct customer segments.

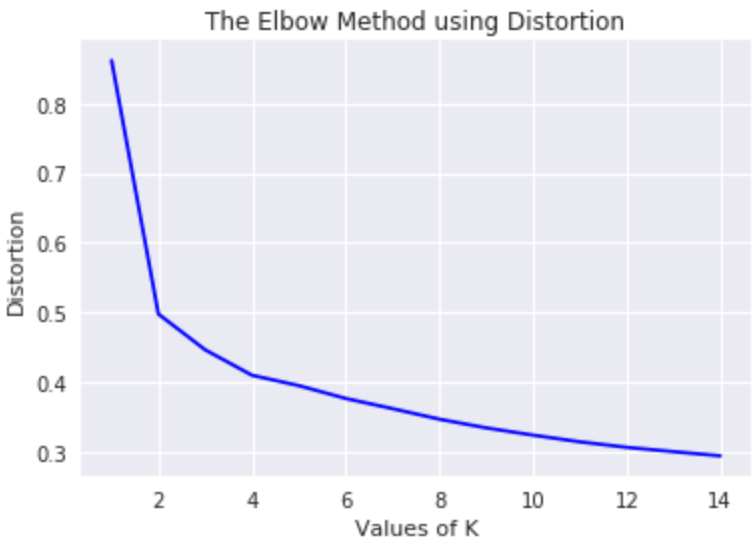


Figure: Distortion curve to evaluate k-means clustering performance with values of *k* = 1 to 14.

We visualize the customer segmentation clustering results by reducing the data to three principal dimensions using the principal component analysis (PCA) dimensionality reduction technique. Here, PCA is performed with three components to visualize the segmentation with three-dimensional cartesian coordinates (shown below). The sum of explained variance for the PCA transformation technique is 86.5%, meaning that most of the data's characteristics are captured in these three orthogonal dimensions. The figure below shows that the overwhelming majority of customer data points are grouped into four distinct customer segment clusters. Still, a small number of data points in the center of the chart are not clustered according to these three principal components.

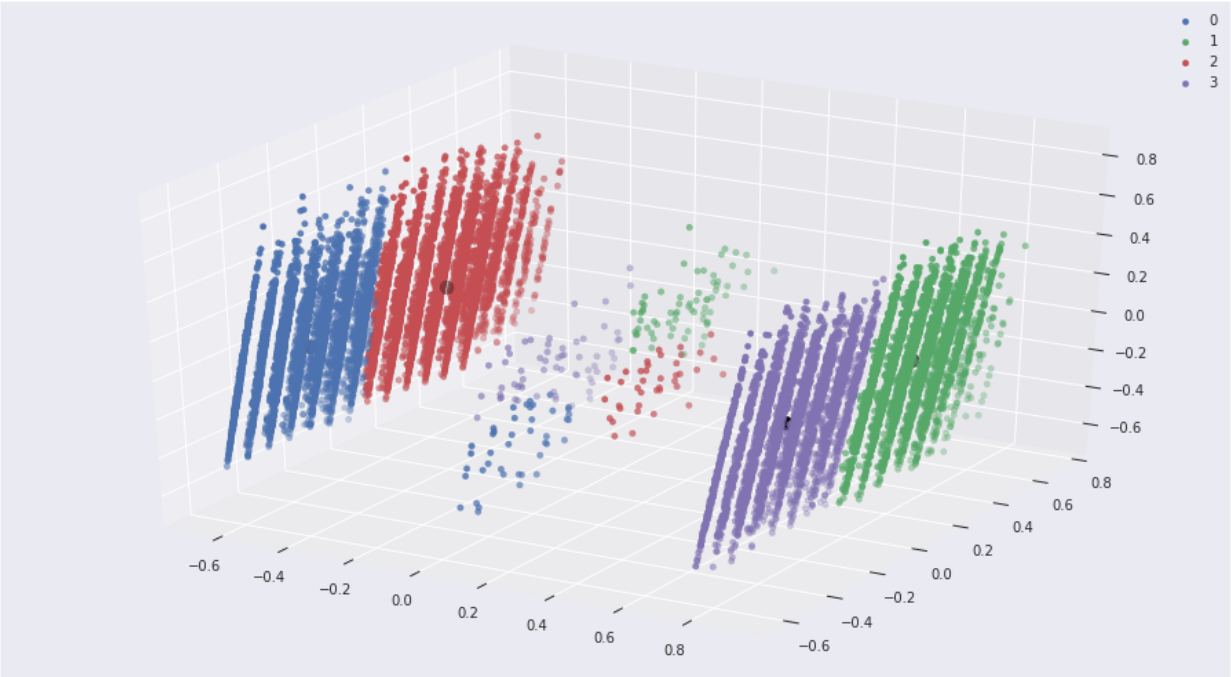


Figure: Customer segmentation results using *k*-mean clustering (*k*=4) and plotted using PCA (components = 3).

The business needs to understand the characteristics of its key customer segments. Understanding customer segments allows the company to better serve the varying needs of each customer segment and market to them accordingly. The average values are calculated for each column in the cleaned profile dataset to understand the composition of each of the four customer segments determined using k-means clustering. The table below shows that the customer segments are primarily grouped by gender and year joined.

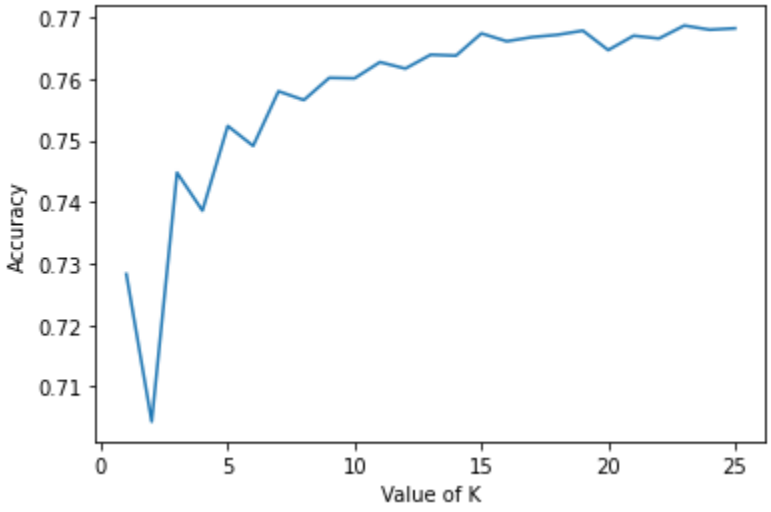
Table I: Average Descriptive Statistics of each customer segment

Customer Segment	Age	Income ($k)	Join Year	Join Month	Gender
All	54.4	65.4	2016.6	6.7	F: 41.3%, M: 57.2%, O: 1.4%
#0	57.8	71.4	2016.2	9.8	F: 97.8%,M: 0%, O: 2.2%
#1	52.0	61.6	2016.1	9.4	F: 0%, M: 99.1%, O: 0.9%
#2	57.5	71.3	2017.1	3.8	F: 98.1%, M: 0%, O: 1.9%
#3	52.1	60.5	2017.2	3.5	F: 0%, M: 98.9%, O: 1.1%

Predict whether a customer completes an offer

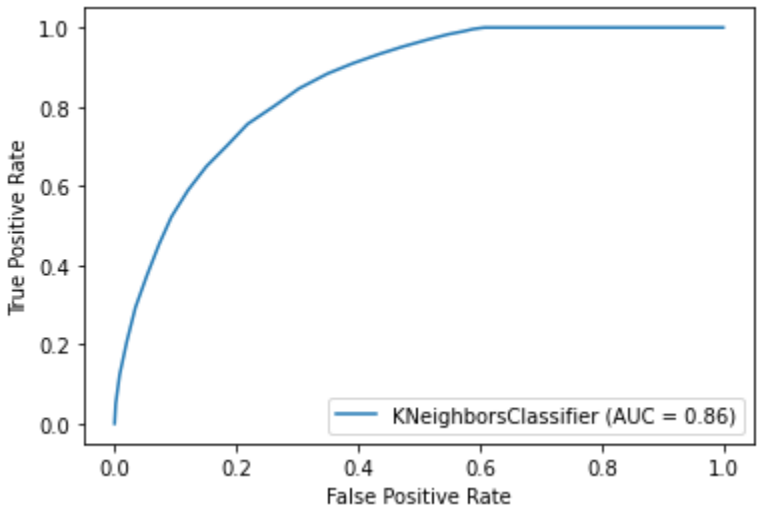
K-Nearest Neighbor (kNN) Classifier

Below is the training accuracy versus the value of *k* for the kNN classifier.



Below is a summary of the prediction performance and ROC curve for the kNN classifier model.

```
Accuracy score (training): 0.789
Accuracy score (test): 0.766
Accuracy score (validation): 0.772


Confusion Matrix:
[[4071 1331]
 [1011 4227]]


Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.75      0.78      5402
           1       0.76      0.81      0.78      5238

    accuracy                           0.78     10640
   macro avg       0.78      0.78      0.78     10640
weighted avg       0.78      0.78      0.78     10640
```

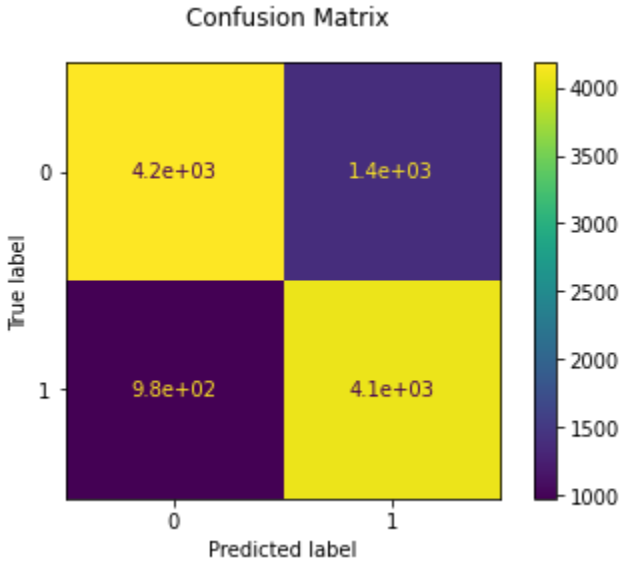


XGBoost Classifier

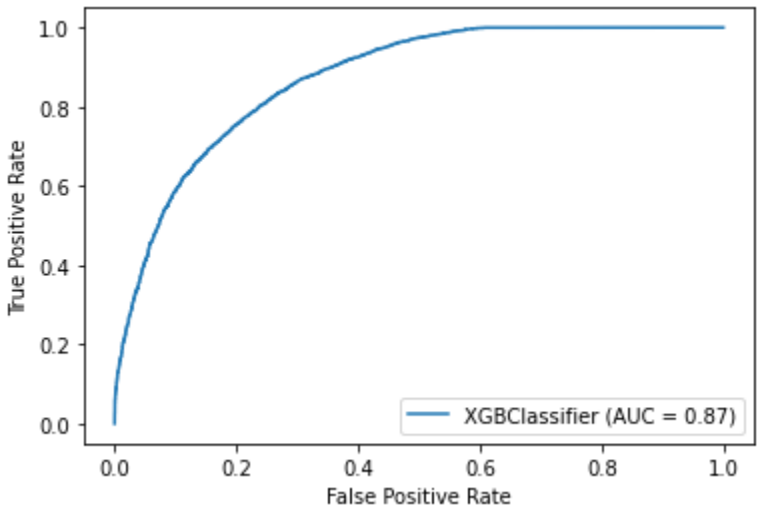
Below is a summary of the prediction performance for the XGBoost classifier model.

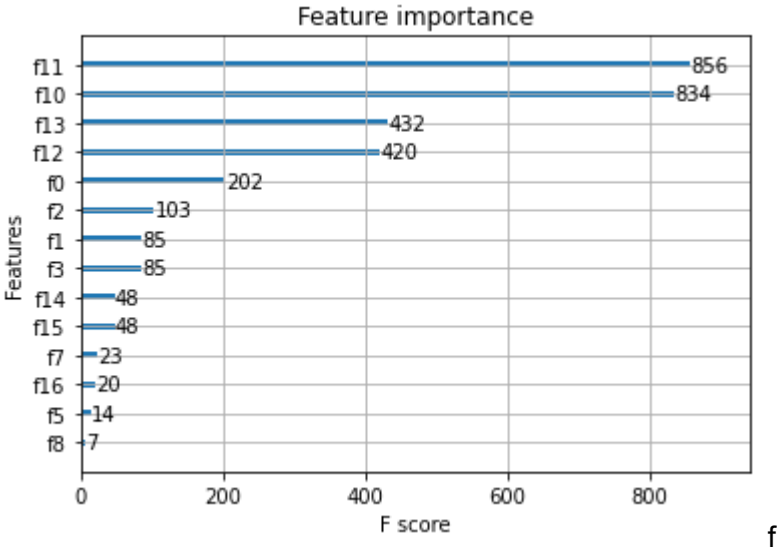
```
Model Evaluation:

XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=500, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)


Accuracy score (training): 0.791
Accuracy score (test): 0.778
Accuracy score (validation): 0.777


Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.75      0.78      5574
           1       0.75      0.81      0.78      5066

    accuracy                           0.78     10640
   macro avg       0.78      0.78      0.78     10640
weighted avg       0.78      0.78      0.78     10640

Confusion matrix:
[[4181 1393]
 [ 976 4090]]
```

Below is the confusion matrix for the XGBoost classifier model. The diagonal yellow squares indicate solid predictive performance.



In addition to the accuracy and f1-score, I compute the area under the receiver operating characteristic curve (ROC AUC) from prediction scores.



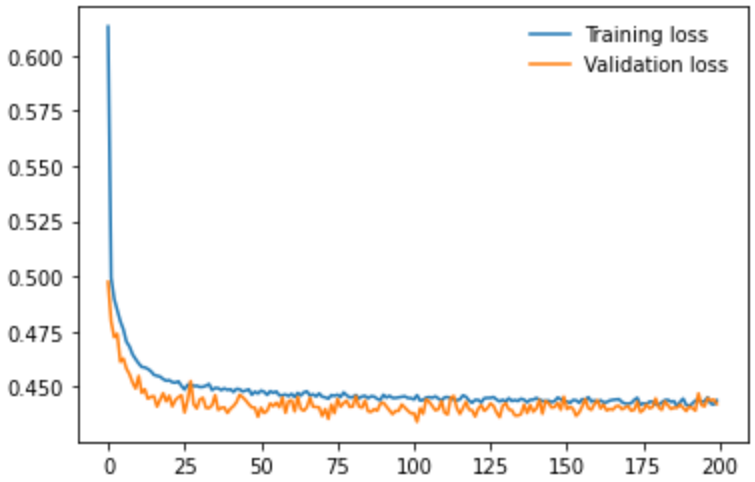


f

The most important features for the XGBoost model are income (*f11*), age (*f10*), year joined (*f13*), and month joined (*f12*).

Deep Neural Network (DNN) Classifier

Below is the training and validation loss for the DNN classifier model. The training process is stable, and the loss (y-axis) continues decreasing over the training epochs (x-axis).




```
Begin Test!
Test Loss: 0.429748


Test Accuracy: 77% (10332/13301)

Confusion Matrix:

[[4952 1821]
 [1148 5380]]

Classification Report:

              precision    recall  f1-score   support

         0.0       0.81      0.73      0.77      6773
         1.0       0.75      0.82      0.78      6528

    accuracy                           0.78     13301
   macro avg       0.78      0.78      0.78     13301
weighted avg       0.78      0.78      0.78     13301
```

Predict whether customers view an offer

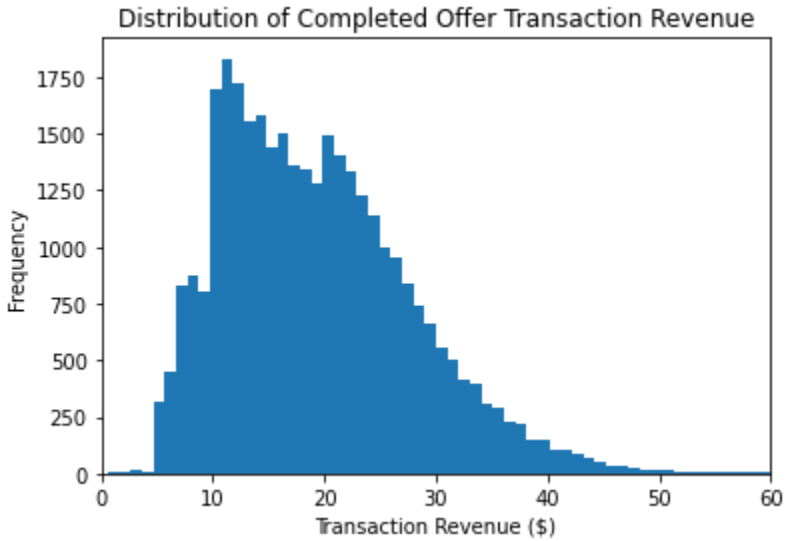
Next, our goal is to determine whether a customer views an offer or not. Below are the results from the XGBoost classifier that predicts whether a customer views an offer.

```
Accuracy score (training): 0.822
Accuracy score (test): 0.815
Accuracy score (validation): 0.821


Classification Report:
              precision    recall  f1-score   support

           0       0.69      0.59      0.63      2789
           1       0.86      0.90      0.88      7851

    accuracy                           0.82     10640
   macro avg       0.77      0.75      0.76     10640
weighted avg       0.81      0.82      0.82     10640

Confusion matrix:
[[1636 1153]
 [ 752 7099]]
```

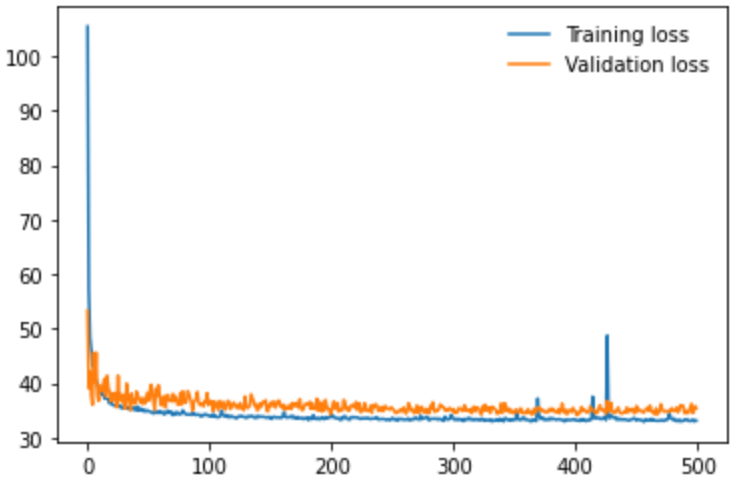
We show the results of all five ML models under investigation in the justification section below.

Predict amount spent

Below is a histogram of the transaction revenue for all of the completed offers. To improve model performance, we remove large positive outlier data for transaction totals above $60. I justify removing outlier data because the dataset includes several significant transaction amounts up to $1,000 that skew model predictions and reduces overall prediction performance.



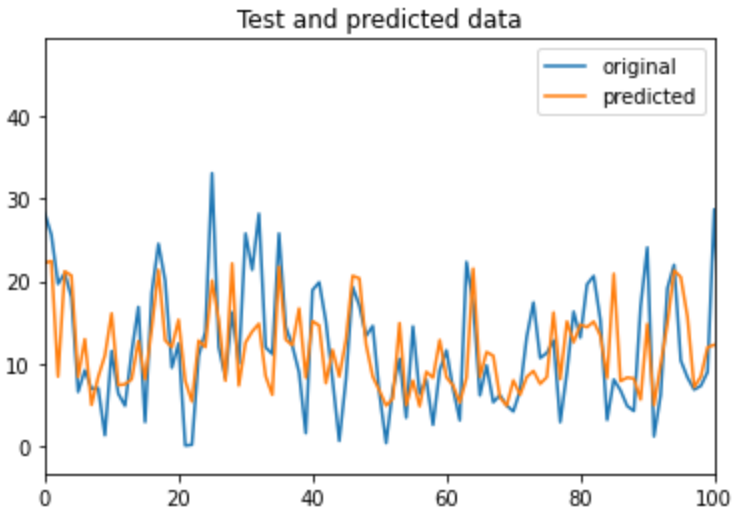
Below is the training and validation loss for the DNN regression model trained for 500 epochs. The loss is generally smooth and decreasing.

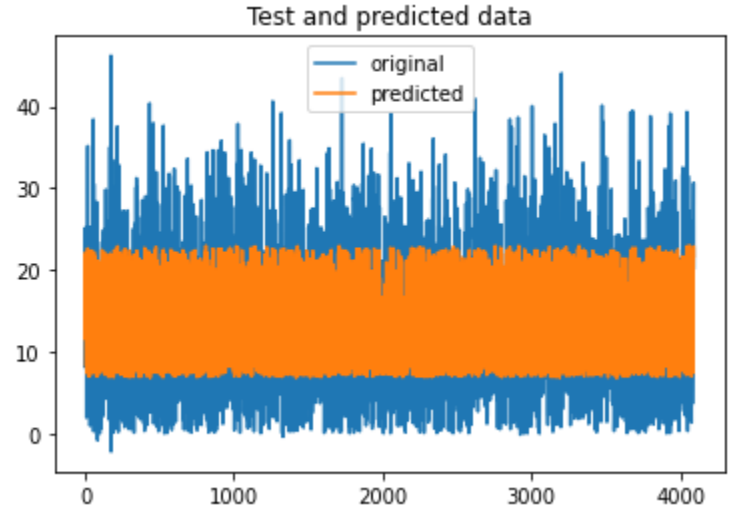


Below are the test results and RMSE score for the DNN regression model.

```
Begin Test!
Test Loss: 32.476784

R2 score: 0.47
Max error: 29.56
Explained Variance Score: 0.47


MSE: 32.47
RMSE: 5.70
```

The charts below compare test and predicted data points. When zooming in to the first 100 samples, we see that the model does a pretty good job of predicting the adjusted revenue value for each completed offer based on the customer and offer characteristics. However, when we zoom out of all ~4000 samples, we see that the model struggles to predict outlier data. Further investigation and training are required to capture those outliers better.





Justification

Classification (Offer Completed):

Offers are completed 61% of the time in this dataset. As a result, a validation accuracy above this value indicates that the model has a predictive value. Logistic regression (LogReg), our benchmark model, achieves an f1-score of 0.75 and a validation accuracy of 75.2%. The XGBoost model was the best performing ML model in this study for predicting completed promotions. The XGBoost model achieves a better f1-score, validation accuracy, and ROC AUC than the SVM, kNN, LogReg, and DNN models, as shown in the table below. I predicted that the DNN model would have the best performance. However, these results indicate that XGBoost achieves superior performance while only requiring a small fraction of the training resources.

Model	f1 Score	Validation Accuracy	ROC AUC	Precission	Recall
Logistic Regression (benchmark)	0.75	75.2%	0.84	0.76	0.76
SVM	0.75	74.9%	0.83	0.75	0.75
kNN	0.77	76.5%	0.85	0.77	0.77
XGBoost	**0.78**	**78.0%**	**0.87**	**0.78**	**0.78**
DNN	0.78	77.0%	Not available	0.78	0.78

Classification (Offer Viewed):

The relative performance between each model predicting whether a customer views an offer is very similar to the completed offers. Customers view 75% percent of the offers in the training dataset. All five ML models in this investigation achieve a validation accuracy higher than 75%, indicating that they have predictive value. Once again, the XGBoost model outperforms the benchmark LogReg model and the SVM, kNN, and DNN models.

Model	f1 Score	Validation Accuracy	ROC AUC	Precission	Recall
Logistic Regression (benchmark)	0.79	79.3%	0.84	0.79	0.79
SVM	0.78	77.8%	0.78	0.77	0.78
kNN	0.81	81.0%	0.85	0.8	0.81
XGBoost	**0.82**	**82.1%**	**0.87**	**0.81**	**0.82**
DNN	0.81	80.0%	Not available	0.81	0.81

Regression:

The DNN model shown in this study achieves a better RMSE, R2 score and explained variance compared to the benchmark linear SGD model and SVM, kNN, and XGBoost models. However, the DNN model required GPUs and more time to train. The XGBoost and kNN models achieve comparable performance to the DNN model while requiring only a fraction of the training resources.

Model	RMSE	Mean Cross-Val	R2 Score	Max Error	Explained Variance
Linear SGD (benchmark)	5.99	0.41	0.41	29.60	0.41

SVM	6.07	0.40	0.4	30.81	0.41
kNN	5.90	0.38	0.43	26.63	0.43
XGBoost	5.83	0.42	0.44	28.30	0.44
DNN	**5.70**	Not available	**0.47**	**29.56**	**0.47**

In each of these three studies, all five of the ML models I evaluated have relatively comparable performance. Our goal is to achieve near 100% accuracy in every model, but this is not realistic with this dataset. Since these five disparate models arrive at a comparable level of performance, we can be reasonably confident that we capture all or almost all of the probabilistic information in the dataset. Based on these results, I conclude that my ML model implementation adequately solves the problem.

Conclusion

Overall, this project was surprisingly challenging due to the structure of the *transcript* dataset and the preprocessing required to generate helpful ML model predictions. However, I learned a lot in this project and developed solutions to the challenges discussed in the project proposal:

- I developed predictive machine learning models to classify whether views and completes a promotional offer based on personalized customer and offer characteristics.
- Developed a predictive model to determine how much a customer spends by completing an offer after subtracting out the cost of the reward
- The ML predictive model outperforms the benchmark model in each of our three studies, and the benchmark models also perform pretty well.
- Determined the features that are most significant to whether or not a customer views and completes a promotional offer
- Defined key customer segment groups based on customer demographics

The models and insights developed in this project allow the company to understand the primary drivers of an effective and profitable promotional offer. The predictive models will enable the company to personalize the characteristics of each promotion to maximize its odds of success for each customer. This project is an excellent example of how machine learning models benefit marketing campaigns and create business value. Lastly, this project shows why a more efficient and straightforward machine learning model is often preferable to a larger and more complex deep learning model.

The code and analysis for this project are available on my GitHub page.

References

[1] S. Zhang, L. Yao, A. Sun, and Y. Tay, “Deep learning based recommender system: A survey and new perspectives,” ACM Computing Surveys (CSUR), vol. 52, no. 1, pp. 1–38, 2019. 6

[2] R. Mu, “A survey of recommender systems based on deep learning,” IEEE Access, vol. 6, pp. 69 009–69 022, 2018.

[3] K. K. Tsiptsis and A. Chorianopoulos, Data mining techniques in CRM: Inside customer segmentation. John Wiley & Sons, 2011.

[4] R. Law and N. Au, “A neural network model to forecast Japanese demand for travel to hong kong,” Tourism Management, vol. 20, no. 1, pp. 89–97, 1999.

[5] G. P. Zhang, Neural networks in business forecasting. IGI Global, 2004.

[6] A. L. Loureiro, V. L. Miguéis, and L. F. da Silva, “Exploring the use of deep neural networks for sales forecasting in fashion retail,” Decision Support Systems, vol. 114, pp. 81–93, 2018.

[7] A. Y. L. Chong, E. Ch’ng, M. J. Liu, and B. Li, “Predicting consumer product demands via big data: The roles of online promotional marketing and online reviews,” International Journal of Production Research, vol. 55, no. 17, pp. 5142–5156, 2017.

[8] Z. Zhao, J. Wang, H. Sun, Y. Liu, Z. Fan, and F. Xuan, “What factors influence online product sales? online reviews, review system curation, online promotional marketing and seller guarantees analysis,” IEEE Access, vol. 8, pp. 3920–3931, 2019.