Machine Learning Engineer Nanodegree
Starbucks Capstone Project Report

Cole Lineberry

July 29 2022

Find a working classification model that will help Starbucks figure out promotional offer effectiveness in their mobile application

Project Overview

Starbucks is a world wide coffee maker and one of the Fortune 500 companies. They have thousands of customers who buy their coffee daily. These customers have the option to use a mobile application to purchase coffee. The company has gathered lots of customer data and demographics from these mobile apps. The marketing team at Starbucks has been offering some buy one get one free promotions (BOGO). These are hard to predict if they will be successful. Not all users receive an offer every week and not all users receive the same offer. Starbucks would like to be able to predict if these offers will be completed. A completed offer means that the terms of the offer have been met.

The dataset provided by Starbucks contains simulated data that mimics customer behavior on the Starbucks rewards mobile app. Once every few days, Starbucks sends out an offer to users of the mobile app. An offer can be an ad for a drink or an actual offer for a discount or BOGO. Some users might not receive any offer during certain weeks.
The data is contained in three files:

- 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

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

Problem Statement

Starbucks would like to know which offers to send to which demographic groups based on previous responses to offers. They would also like to be able to predict which offers will most likely be completed/converted by these groups. Starbucks has compiled 30 days worth of data.
If we assume past performance is a measure of future conversion rates we can apply machine learning algorithms to predict these future outcomes.
- Machine Learning task: The problem is defined as a classification task, which demographics are most likely to convert. What is the likelihood that something will convert.
- Input: Previous customer responses and associated demographic data.
- Output: Find a machine learning algorithm that can predict which individuals are most likely to convert in the marketing campaign.

This project will consist of data cleaning and preparation for running the machine learning algorithms. These algorithms will be evaluated based on F1 score and compared to a benchmark. This will allow us to pick the best machine learning algorithm to be used to predict an outcome of sending a mobile offer to users.

Metrics

The models used to predict the event outcomes will be judged based on F1 scores. Classification problems respond well to this metric for figuring out if a model is performing well. These scores will be calculated by the sklearn_metrics packages method called fbeta_score and the accuracy_score.
F1 scores are defined by Towards Data Science as

“F1 Score is used to measure a test’s accuracy. F1 Score is the Harmonic Mean between precision and recall. The range for F1 Score is [0, 1]. It tells you how precise your classifier is (how many instances it classifies correctly), as well as how robust it is (it does not miss a significant number of instances).
High precision but lower recall, gives you an extremely accurate, but it then misses a large number of instances that are difficult to classify. The greater the F1 Score, the better is the performance of our model. “

https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234

Data Exploration

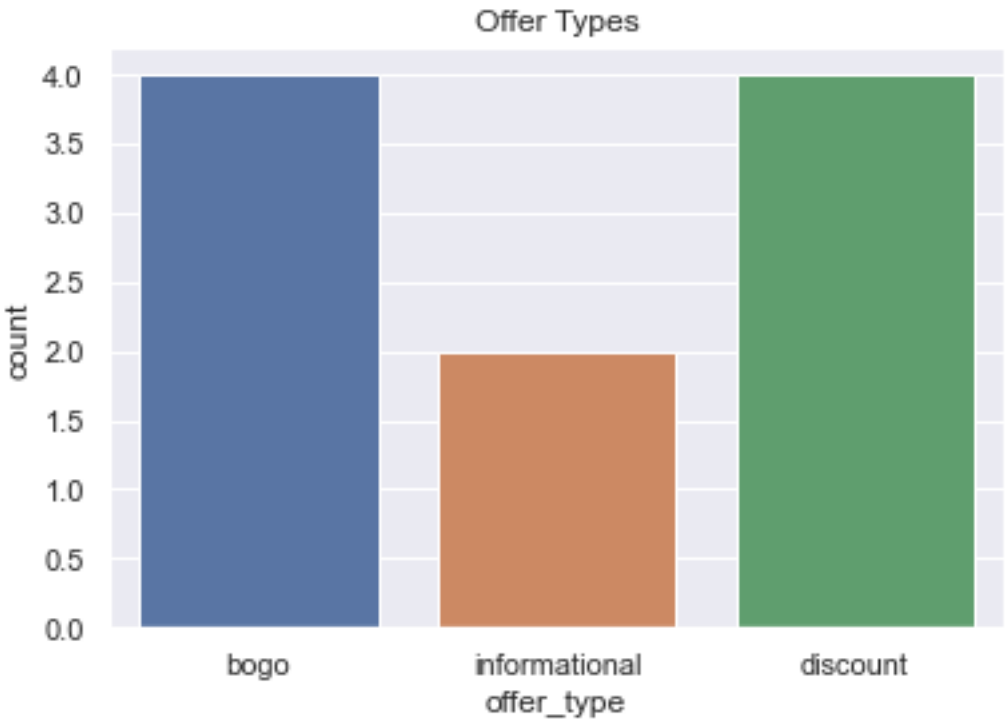
For this problem we can analyze several attributes of the data to gain an understanding of how this data is related to itself and the process of getting an order completed.

Portfolio
The portfolio file has lots of user data as it pertains to transactions
The portfolio file looks like this:

	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

We will need to link the id types to the ids in both the transaction and profile files. A very critical piece of information here is the offer type which will need to be converted to a number using the map function.

This is the breakdown of the counts of offer types:

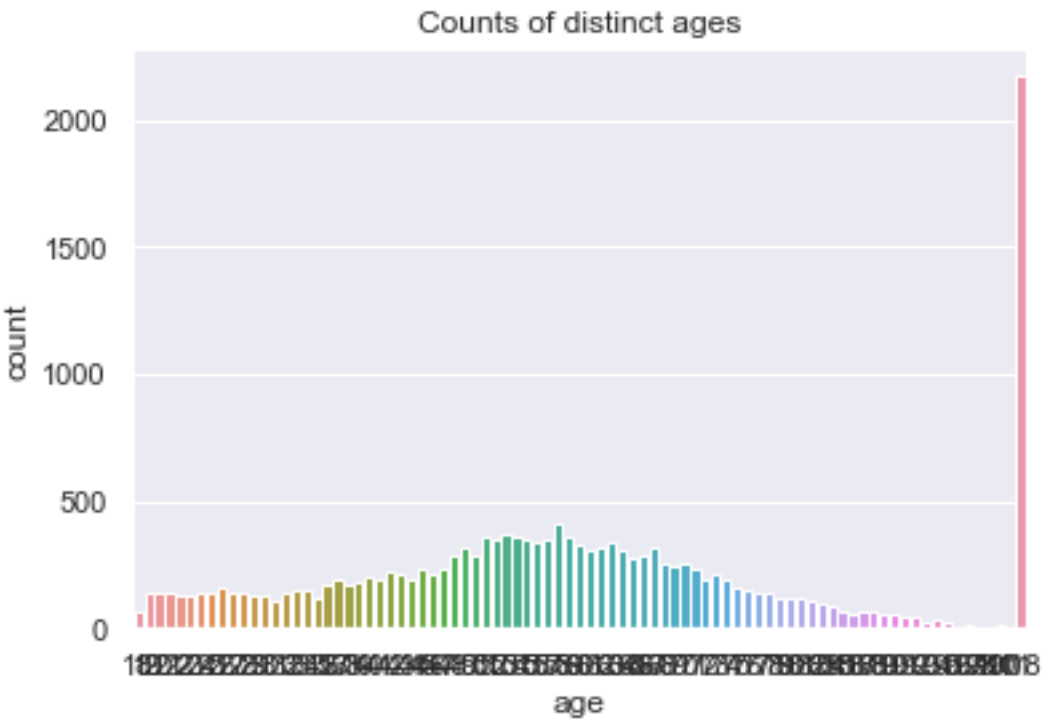


Profile
The profile file has demographic data about a particular user.
The profile file looks like this:

	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

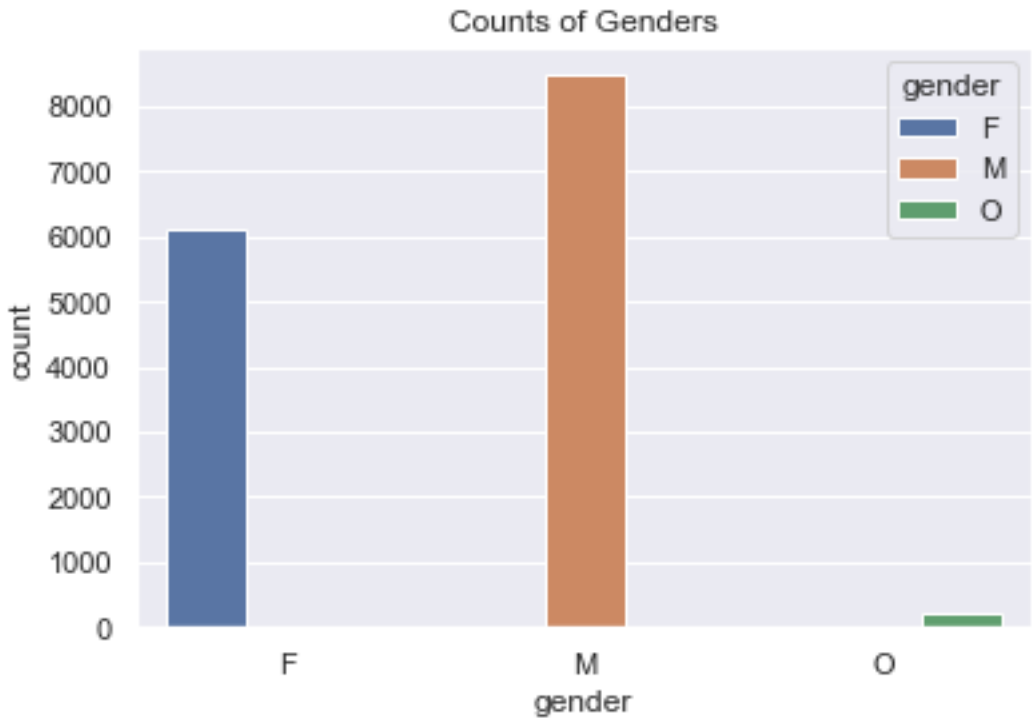
This file will be linked on the id to the portfolio and transaction files. Some data manipulation will need to be done for the income and gender as those are important to understanding the demographics of the mobil app users.

This is a breakdown of the ages of the users -



As is evident from the graph there are lots of distinct ages with all of the ages that were not entered being set to 108 which skews the graph seen above.

This is a breakdown of the gender numbers -



As is evident from the graph there are several other selections for gender that will need to be conformed to the mean value in order for our algorithms to work against this data.

Here is the distribution of incomes -

As is evident from this graph the incomes will need to be divided up and categorized to be useful in a machine learning algorithm context.

Transactions
The transactions file describes a specific event that happened to a users. This will be linked to the other tables using the person id.

This is the transactions file:

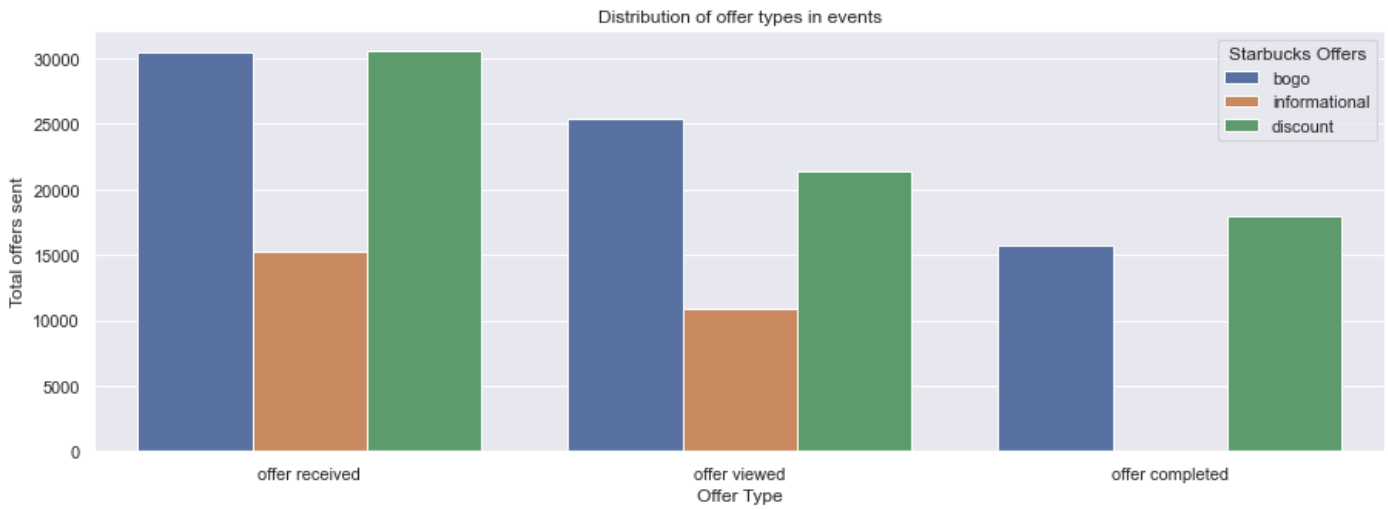
	person	event	value	time
0	78afa995795e4d85b5d9ceeca43f5fef	offer received	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}	0
1	a03223e636434f42ac4c3df47e8bac43	offer received	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}	0
2	e2127556f4f64592b11af22de27a7932	offer received	{'offer id': '2906b810c7d4411798c6938adc9daaa5'}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

The most important attribute in the transaction table is the event column which we will use to determine if a user has completed an offer or not.
Here is the counts of event types:

Exploratory Visualization

In this section, you will need to provide some form of visualization that summarizes or extracts a relevant characteristic or feature about the data. The visualization should adequately support the data being used. Discuss why this visualization was chosen and how it is relevant.
Questions to ask yourself when writing this section:
- _Have you visualized a relevant characteristic or feature about the dataset or input data?_
- _Is the visualization thoroughly analyzed and discussed?_
- _If a plot is provided, are the axes, title, and datum clearly defined?_

A very interesting data visualization is the distribution of offer types vs events which is shown below:



This view was created by combining the data sets together and then producing a count to show what happened to each offer type, answering the question of if it was received, viewed and ultimately if it was completed. According to this chart, buy one get one offers were the most viewed, but discount offers were the most completed. This leads us to the conclusion that users are more likely to complete a discount offer. Also these same discount offers are the least viewed proportionally to what is sent out.

Algorithms and Techniques

The algorithms I will be using to create a predictive model are the following:
- KNeighborsClassifier
- RandomForestClassifier
- DecisionTreeClassifier

This solution will use the K Nearest Neighbors classifier algorithm as the benchmark algorithm for testing the other algorithms against. A setting of Five nearest neighbors will be used to start.

According to Towards data science the K Nearest neighbor algorithm is great for machine learning classification problems.

“The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems.” https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

It works by associating like values to each other and then predicting similar values based on that.

“The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.”
https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

This makes it a great benchmark which will be discussed later

Decision Tree Classifier is used to classify data in machine learning algorithms. It will split the data along lines based on features in the data. This uses a series of decisions to determine like features and then classify them. A setting of 10 random state will be used to Start.

Random Forest Classifier also works great for classification machine learning predictions. Towards Data Science defines Random Forest Classification as
“Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model’s prediction”
https://towardsdatascience.com/understanding-random-forest-58381e0602d2

Random Forest classification takes lots of decision trees and evaluates the output of them all giving a better overall result than a single decision tree could produce. A setting of 10 random state will be used to Start.

Benchmark

The benchmark that will be used for testing the algorithms will be the F1 score that is produced from the K Nearest Neighbors algorithm with a five setting on the nearest neighbors option. This should produce an F1 score that we can compare the other algorithms against and should be easy to see the best, defined by the highest, F1 score available.

Data Preprocessing

All of the classification algorithms require the data be standardized and work best with numerical data so several transformations where required for these data sets to be able to get an F1 score from each of the algorithms and to keep the data consistent between algorithms.

- These are some steps taken:
- replace missing age values with mean age
- replace the missing income values with mean income
- replace missing gender value using gender mode
- Set Gender O rows to the mean gender value
- Convert income to 4 different levels

income < 10000 = 'Low'
income >= 10000)& income <= 50000 = 'Lower_Middle'
income >= 50000 & income <= 100000 = 'Upper_Middle'
income >= 100000 = 'High'

- Convert age to 4 different levels
  age < 18 = 'Child'
  age >= 18 & age <= 30 = 'Young_Adult'
  age >= 30 & age <= 65 = 'Old_Adult'
  age >= 65 = 'Senior'

Offer id from transaction was encapsulated in an array in the table, that needed to be broken out
Merge all the data from the three data frames into one mega data frame so we can make inferences about the data.
change all the age_groups to numbers
change all the income levels to numbers
change all the offer types to numbers
change all the gender types to numbers
drop unused columns

Implementation

In the initial implementation this solution will use the event that was converted to a 1,2, or 3 to create the benchmark and initial algorithms. Then a feature and label set was defined:
features = preTrain.drop('event', axis=1)
label = preTrain['event']

Then the data was split into training and test as shown below:

```
X_train, X_test, y_train, y_test = train_test_split(features, label, test_size = 0.4, random_state = 0)

print("Training set: {} rows".format(X_train.shape[0]))
print("Testing set: {} rows".format(X_test.shape[0]))
```

```
Training set: 100548 rows
Testing set: 67033 rows
```

Then the classifier algorithm was defined and an f1 score generated like this -

```
neighbor = KNeighborsClassifier(n_neighbors = 5)
```

```
training_prediction =  (neighbor.fit(X_train, y_train)).predict(X_train)
```

```
test_prediction = (neighbor.fit(X_train, y_train)).predict(X_test)
```

```
training_f1 =  accuracy_score(y_train, training_prediction) * 100
```

```
test_f1 = fbeta_score(y_test, test_prediction, beta = 0.5, average='micro' ) * 100
```

```
print(f"Benchmark Training F1 Score:{training_f1}, Testing F1 Score:{test_f1}")
```

```
Benchmark Training F1 Score:75.31626685762023, Testing F1 Score:67.28477018781794
```

This technique was reused for each of the different algorithms so eventually it was put into a function like this -

```
#lets make a repeatable way to build training and predictions
def runPrediction (classifier, X_train, y_train ):
    my_training_prediction = (classifier.fit(X_train, y_train)).predict(X_train)
    my_test_prediction = (classifier.fit(X_train, y_train)).predict(X_test)
    my_training_f1 = accuracy_score(y_train, my_training_prediction) * 100
    my_test_f1 = fbeta_score(y_test, my_test_prediction, beta = 0.5, average='micro' ) * 100
    print("Training F1 Score:{}, Testing F1 Score:{}", my_training_f1, my_test_f1)
    return my_training_f1, my_test_f1
```

Refinement

The main refinement change that was made to this solution was the addition of the Balanced Random Forests algorithm. Balanced Random Forest is the same as random forest but attempts to account for imbalanced data where it will attempt to balance each bootstrapping iteration by under sampling. This will be used to just add another data point to compare against the benchmark and hopefully account for any imbalance in the normal Random Forest algorithm.

In addition several of the attributes were converted to numbers between -1 and 1 in case there were any algorithms that were being thrown off by the features not being fit properly. Here is how those values were reshaped -

```
#convert income to a number between -1 and 1
preTrain['income'] = std.fit_transform(preTrain['gender'].values.reshape(-1, 1))
```

```
preTrain['offer_type'] = std.fit_transform(preTrain['offer_type'].values.reshape(-1, 1))
```

```
preTrain['reward'] = std.fit_transform(preTrain['reward'].values.reshape(-1, 1))
```

```
preTrain['difficulty'] = std.fit_transform(preTrain['difficulty'].values.reshape(-1, 1))
```

```
preTrain['income_level'] = std.fit_transform(preTrain['income_level'].values.reshape(-1, 1))
```

```
preTrain['age_group'] = std.fit_transform(preTrain['age_group'].values.reshape(-1, 1))
```

Model Evaluation and Validation

From the initial comparison table above we can see that the K Nearest Neighbor benchmark is beaten by the Random Forest and Balanced Random Forest Classifer and Decision Tree Classifier algorithms. These test and training F1 scores are similar and are within range of each other. In that case Starbucks should use the Decision Tree Classifer to make decisions about which type of offers to send since this algorithm had the best scores. However, once we removed the event column and worked solely off the completed column we got very different scores. Removing the guess work on if an event was completed brought our scores up significantly. Now the balanced random forest score is the worst and the Decision Tree score

was the best followed closely by the random forest score. This means Starbucks should use this data set and the decision tree algorithm to predict which transactions will be completed.

An analysis of the data graphs in the previous section suggest that high income women are more likely to complete an offer so focusing offers sent to them will generate the best completion rate.

The final model chosen was the decision tree algorithm. This provided the best F1 score. Initially this was not the case as our models were incorrectly using the event column which had multiple values when we were only really concerned with the completed outcomes. During evaluation it was discovered that the event column was not producing very confident models as all of the F1 scores were low. Once the change was made to use completion column the scores for all of the algorithms improved. It was surprising that the balanced forest algorithm performed the worst in this case but proves that this data did not need balancing.

The F1 scores are very high in the final configuration and that gives a good indication that we can trust this model.

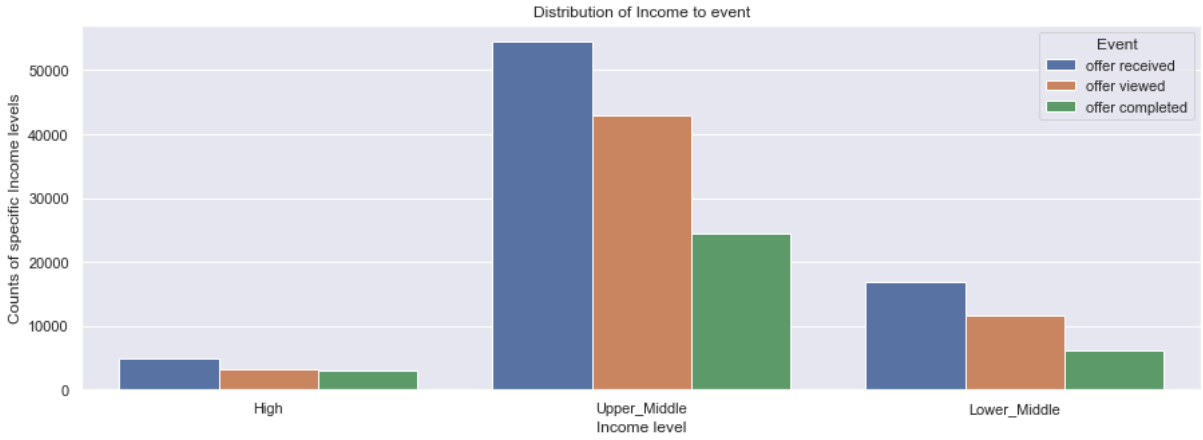
Justification

	Algorithm Type	Training F1 score	Testing F1 score
0	K Neighbors Classifier	76.760353	75.667209
1	Random Forest Classifier	80.109997	79.507108
2	Balanced Random Forest Classifier	55.513784	53.152925
3	Decision Tree Classifier	80.109997	79.633912

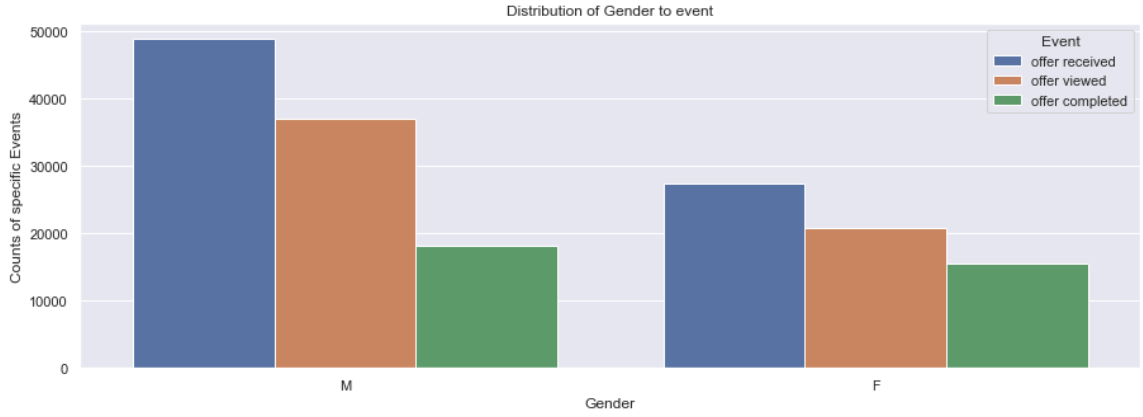
The Benchmark established was the F1 score from the Nearest Neighbor algorithm. In the final configuration there was no significant difference between the F1 scores for Decision Tree Classifier and Random Forest Classifier. Both scores were better than the benchmark algorithm and both would perform well in predicting the transaction completion. These higher F1 scores give a good justification for using these algorithms.

Free-Form Visualization

There are two interesting details that came out of the data analysis -
1) High income levels are more likely to complete an offer.



2) Women are more likely to complete an offer



It is interesting that high earning women are the most likely to complete an offer. Though the data set contains more men overall (which may be a statistical anomaly as the unknowns genders were set to men). Women are completing offers about as many times overall as the men. Also people who have more income complete offers at a higher rate, even though they may not need the money that an offer inherently provides.

Reflection

This was a very interesting project to work on with a robust data set. I found it challenging to wrangle the data down to something useful for the algorithms to use. I also found it

challenging to get the data set to actually produce good F1 scores. My first round through with the event feature as the predictor was a great surprise to me. I am glad that I was able to get the F1 Scores to improve by switching to the other feature completed. This rework took considerable effort but was worth it to get some algorithm numbers that were more reliable. As mentioned above the gender vs completed rate was fascinating as was the income to gender information contained in this data set.

Improvement

In the future we could add a web endpoint and wrap that with some html code to take in a few input items like Gender, Age, Income and predict the likelihood of an offer event being completed. This would use the Decision Tree Classifier and would be able to accurately predict a users behavior given those entered customer data points. It would also be intersting to study the pay vs gender data that is available here and see if there are other demographic predictors we could use for other features in this dataset.

I think there are more classification algorithms we could try and see if we can beat the benchmark set in this study.

References:

F1 definitions - https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234