Machine Learning Engineer (MLE)

Capstone Project Report

Udacity Machine Learning Engineer Nanodegree

Ravi Subramanian

December 16, 2020

Table of Contents

Introduction

This project is focused on presenting personalized offers based on spending habits. I chose this project as it not only applies to Starbucks, but also to many other real world problems on presenting the right offers to the right customers at the right time.

Domain Background

As a customer of many products myself, I often get frustrated with irrelevant ads, offers and rewards that doesn't apply to me. I often feel companies are bombarding with spam mails, offers and rewards that lead me to ignore reading the offer even sometimes if it totally applies to me. (missed offers)

Personalized timely offers is one of the best way to engage customers using the products that not only benefits the customer needs but also adds business value as the customer is engaged, motivated and use more product offerings creating a win-win situation.

Problem Statement

The main problem here is to present the offers that are timely and relevant and eventually engage customers, meaning we would like to determine the right offer that is sent and used by customer based on their past purchases and interaction with previous sent offers. We also want to avoid sending inappropriate offers. One possible key metric is to increase the efficiency of the offers *(Increase the percentage of USED OFFERS/SENT OFFERS per customer)*

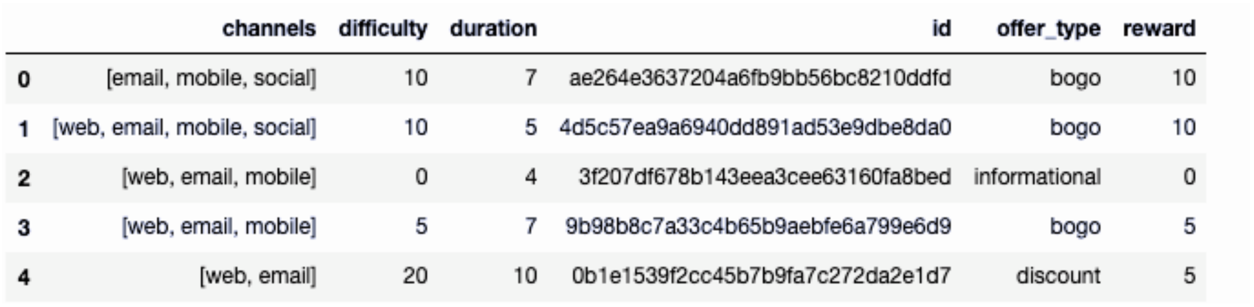
Dataset

The data provided contains 3 files. It data was captured over a 30-day period.

I. Portfolio.json

This json file has information about the different kinds of offers (Total 10 offers)

- **Reward:** The USD given to the customer for completing an offer
- **Channels:** The medium in which the offer is sent (Email, Mobile, Social, Web)
- **Difficulty:** The minimum USD that the customer must spend in order to complete that offer
- **Duration:** The number of days that the offer will last
- **Offer Type:** Buy One Get One (BOGO), Discount, Informational
- **ID:** A unique id for a particular offer

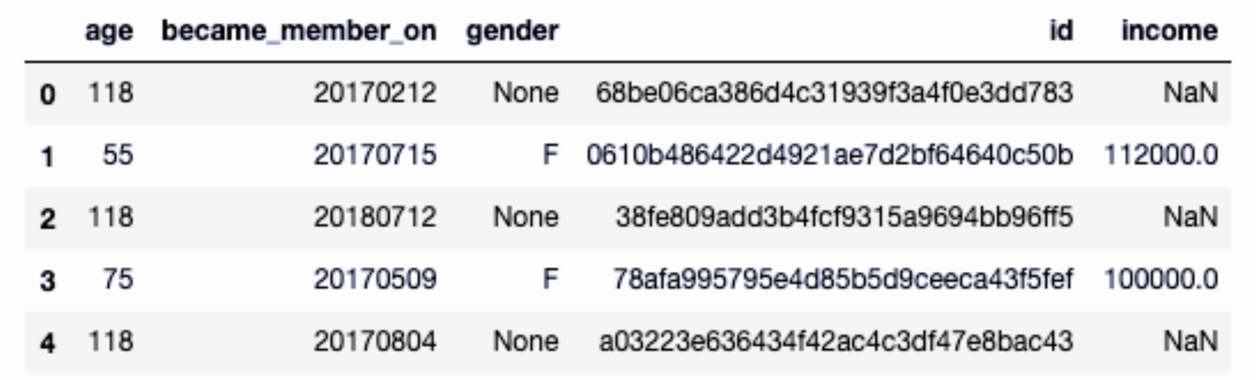


	channels	difficulty	duration	id	offer_type	reward
0	[email, mobile, social]	10	7	ae264e3637204a6fb9bb56bc8210ddfd	bogo	10
1	[web, email, mobile, social]	10	5	4d5c57ea9a6940dd891ad53e9dbe8da0	bogo	10
2	[web, email, mobile]	0	4	3f207df678b143eea3cee63160fa8bed	informational	0
3	[web, email, mobile]	5	7	9b98b8c7a33c4b65b9aebfe6a799e6d9	bogo	5
4	[web, email]	20	10	0b1e1539f2cc45b7b9fa7c272da2e1d7	discount	5

II. Profile.json

This json file has information about the customer demographics

- **Gender:** M (Male), F (Female), O (Other) & None if not available
- **Age:** Birth year defaults to 1900 if no age is available
- **ID:** of a customer (unique)
- **Became a member on:** Date in YYYYMMDD
- **Income:** USD annual, NaN if not available

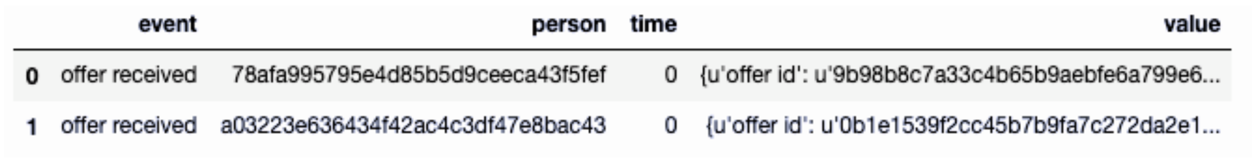


	age	became_member_on	gender	id	income
0	118	20170212	None	68be06ca386d4c31939f3a4f0e3dd783	NaN
1	55	20170715	F	0610b486422d4921ae7d2bf64640c50b	112000.0
2	118	20180712	None	38fe809add3b4fcf9315a9694bb96ff5	NaN
3	75	20170509	F	78afa995795e4d85b5d9ceeca43f5fef	100000.0
4	118	20170804	None	a03223e636434f42ac4c3df47e8bac43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

III. Transcript./ json

This json file has information about the customer purchases and offer interactions.

- **Customer ID:** Unique customer id for this transaction/interaction
- **Event:** Transaction, Offer Received, Offer Viewed, Offer Completed
- **Value:** For the type of event it contains the amount spent if transaction, offer id if offer received or views, offer id or reward id if compare completed.
- **Time:** The time in hours since the start of this test (approximately 30 days)



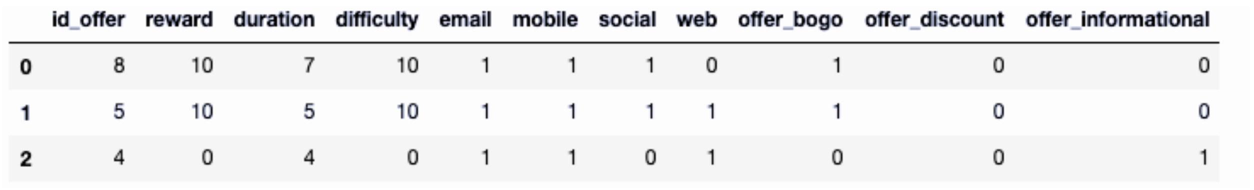
	event	person	time	value
0	offer received	78afa995795e4d85b5d9ceeca43f5fef	0	{u'offer id': u'9b98b8c7a33c4b65b9aebfe6a799e6...
1	offer received	a03223e636434f42ac4c3df47e8bac43	0	{u'offer id': u'0b1e1539f2cc45b7b9fa7c272da2e1...

Data Exploration, Cleaning and Feature Engineering

Portfolio Dataset

- Checked the unique offers:

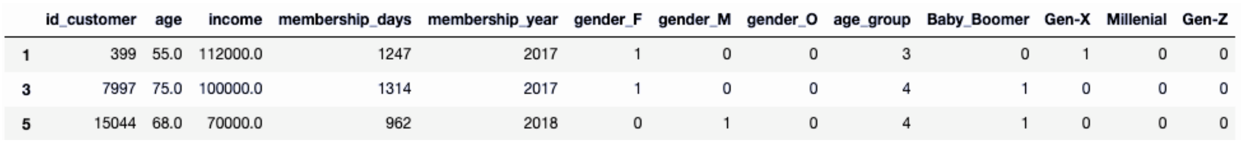
  array([u'bogo', u'informational', u'discount'], dtype=object)
- Applied OHE for offer type and channels
- Renamed and Simplified *offer_id*



	id_offer	reward	duration	difficulty	email	mobile	social	web	offer_bogo	offer_discount	offer_informational
0	8	10	7	10	1	1	1	0	1	0	0
1	5	10	5	10	1	1	1	1	1	0	0
2	4	0	4	0	1	1	0	1	0	0	1

Profile Dataset

- Renamed id to id_customer
- Simplified the id numbers
- Removed records with age=118 /Income = NaN / Gender = None
- Changed the membership date format to YYYY-MM-DD
- Included membership year feature (YYYY)
- Applied OHE for gender type
- Split age into different age groups (Baby Boomer, Gen-X, Millennials, Gen-Z

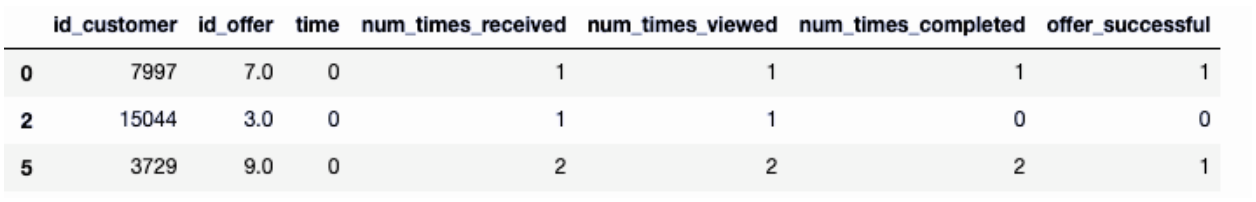


	id_customer	age	income	membership_days	membership_year	gender_F	gender_M	gender_O	age_group	Baby_Boomer	Gen-X	Millenial	Gen-Z
1	399	55.0	112000.0	1247	2017	1	0	0	3	0	1	0	0
3	7997	75.0	100000.0	1314	2017	1	0	0	4	1	0	0	0
5	15044	68.0	70000.0	962	2018	0	1	0	4	1	0	0	0

Transcript Dataset

- Renamed person to id_customer to match the profile dataset
- Applied the same simplified id_customer matching profile dataset
- Applied OHE for event (offer_received, offer_viewed and offer_completed)

- Dropped same records (age=118 /Income = NaN / Gender = None)
- Cleaned the offer column and split into meaningful features with offer_id
- Matched the same simplified offer_id from portfolio dataset
- Feature engineered to create offer_success_rate (class Y label)



	id_customer	id_offer	time	num_times_received	num_times_viewed	num_times_completed	offer_successful
0	7997	7.0	0	1	1	1	1
2	15044	3.0	0	1	1	0	0
5	3729	9.0	0	2	2	2	1

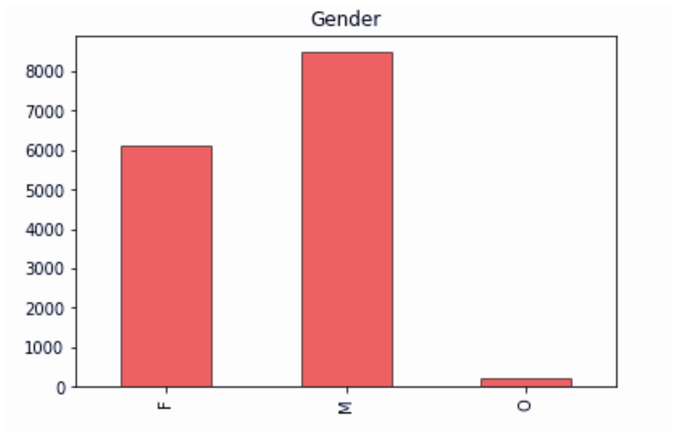
Data Analysis & Insights

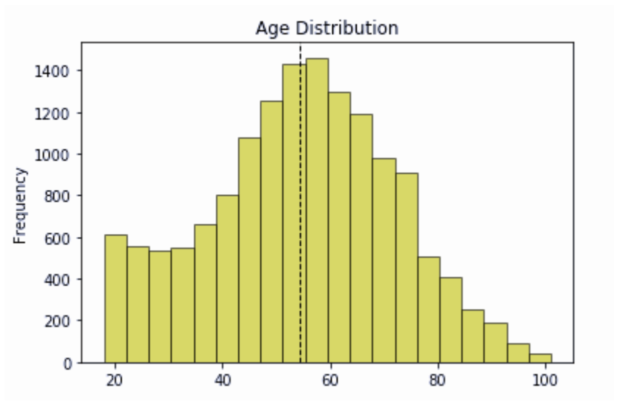
Before diving into modeling, used the clean data set and analyzed the data insights:

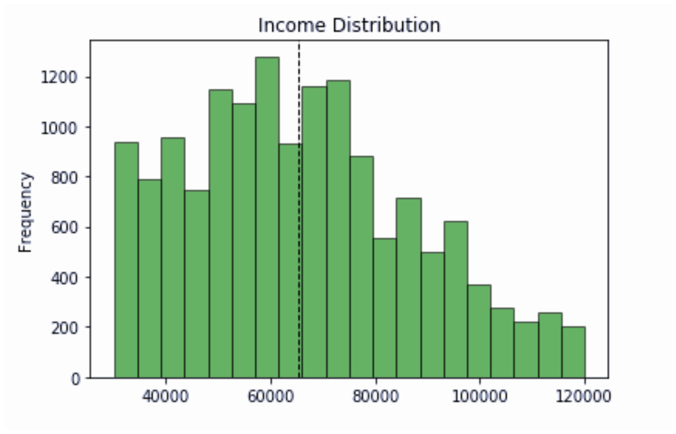
Summary of Findings:

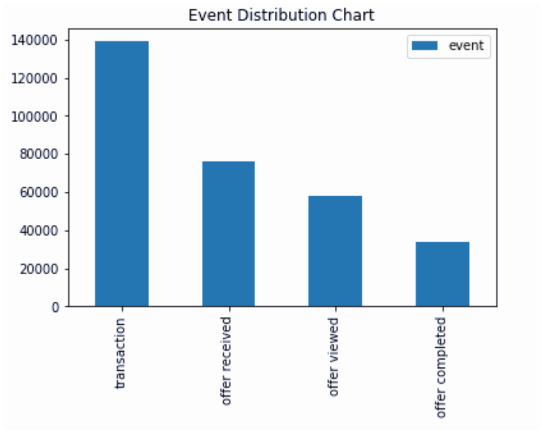
- There are 57% Males and 41% Females in the given dataset
- Mean age of the customers in this dataset is 54
- Income varies from $30000 to $120,000 with a mean of $64,404
- Total Offer success rate (Offer completed/Offer viewed) is ~50%
- Females respond to offers better than Males (Female: 58% and Male:45%)
- Baby Boomers are the top responders to offers (53%).
- Older age group respond to offers more than younger age group

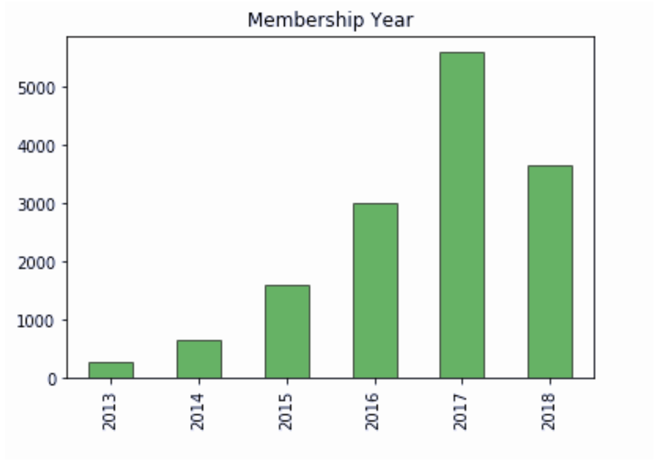
Note: Details are in the notebook







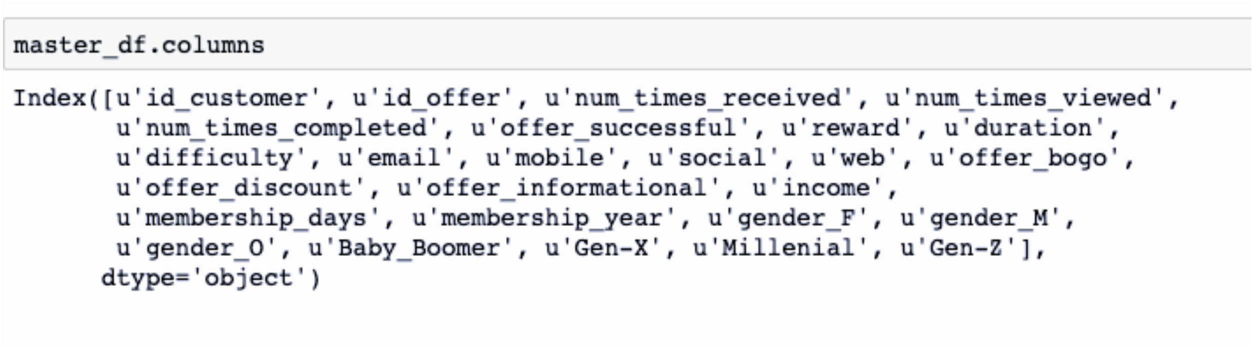




Prepare master dataset for modeling

- Combined all the 3 datasets (Portfolio, Profile and Transcript) into a single combined dataset for modeling

- Used SKlearn MinMaxScaler for feature scaling



```
master_df.columns
```

```
Index([u'id_customer', u'id_offer', u'num_times_received', u'num_times_viewed',
       u'num_times_completed', u'offer_successful', u'reward', u'duration',
       u'difficulty', u'email', u'mobile', u'social', u'web', u'offer_bogo',
       u'offer_discount', u'offer_informational', u'income',
       u'membership_days', u'membership_year', u'gender_F', u'gender_M',
       u'gender_O', u'Baby_Boomer', u'Gen-X', u'Millenial', u'Gen-Z'],
      dtype='object')
```

- Currently, there is a disparity of classes in the variables, before modeling, we would like to make sure the dataset is balanced across train, test and validation datasets, because algorithms tends to categorize into the class with more instances, the majority class, while at same time giving the false sense of highly accurate model. I used SKlearn stratified shuffle split to balance the dataset and split it into train, test and validation before we start the modeling.

- Converted the train, test and validation datasets into numpy float32 array for acceptable format for modeling

Evaluation Metrics

- Created functions for evaluating and printing the confusion matrix (True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN))
- Created functions for calculating Accuracy, Precision, Recall, F1 and F2 scores.

Modeling

```
#Let's check and final benchmark the OFR (Offer Success Rate)
TSR=float(master_df.offer_successful.sum()*1.00 / master_df.offer_successful.count()*1.00)
print("Total Offer Success Rate = ", TSR)
```

```
('Total Offer Success Rate = ', 0.6323098252233005)
```

At present with the current dataset, the Starbucks offer success rate is the following: Current metric: 0.63 or 63%

A. Logistic Regression *(Benchmark)*

Model A. Logistic Regression (LR)

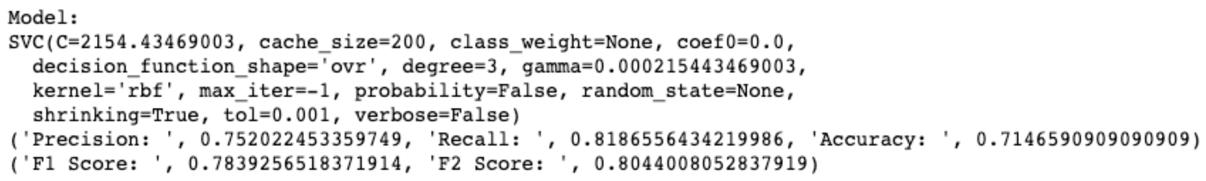
I chose Logistic regression as a benchmark model because it's

```
Model:
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=5000, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=0, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
('Precision: ', 0.7185173932857359, 'Recall: ', 0.8501078360891445, 'Accuracy: ', 0.694659090909091)
('F1 Score: ', 0.7787931176422163, 'F2 Score: ', 0.820070044037588)
```

typically great with binary classification problem. It is an industry wide model that is used as a benchmark model. It is easier to train and implement as compared to other methods. It not only gives a measure of how relevant an independent variables is, but also tell us about the direction of the relationship (positive or negative). For this project, Scikit Learn's has been used with some custom hyper parameters to benchmark the model

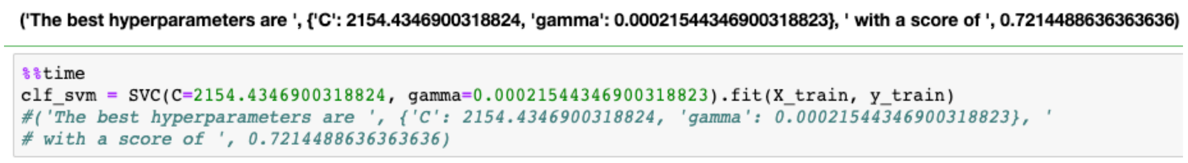
The model results are the following: LR accuracy score is: **0.69**

Model B. Support Vector Machine (SVM)



```
Model:
SVC(C=2154.43469003, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.000215443469003,
  kernel='rbf', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False)
('Precision: ', 0.752022453359749, 'Recall: ', 0.8186556434219986, 'Accuracy: ', 0.7146590909090909)
('F1 Score: ', 0.7839256518371914, 'F2 Score: ', 0.8044008052837919)
```

SVM has a regularization parameter "C" (1/Lambda), which can reduce overfitting. It also uses kernel trick, so we can build expert knowledge about the problem via engineering the kernel. It also has the tendency to avoid local minima.   Used with Sklearn SVC implementation, Optimal Hyper parameters chosen with GridSearchCV (C values & Gamma values)
The optimal hyper parameters:



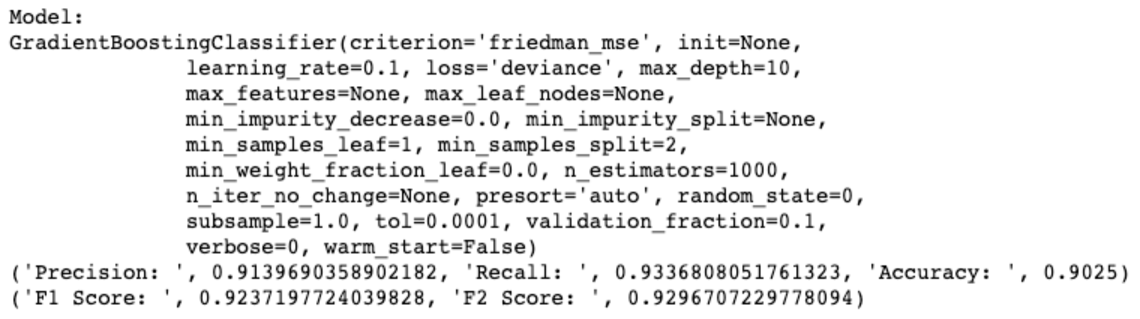
```
('The best hyperparameters are ', {'C': 2154.4346900318824, 'gamma': 0.00021544346900318823}, ' with a score of ', 0.7214488636363636)
```

```
%%time
clf_svm = SVC(C=2154.4346900318824, gamma=0.00021544346900318823).fit(X_train, y_train)
#('The best hyperparameters are ', {'C': 2154.4346900318824, 'gamma': 0.00021544346900318823}, '
# with a score of ', 0.7214488636363636)
```

SVM accuracy score is: **0.75**

Model C. Gradient Boost

Gradient boost is one the best algorithm that provides a great predictive accuracy. As we are interested in improving our offer



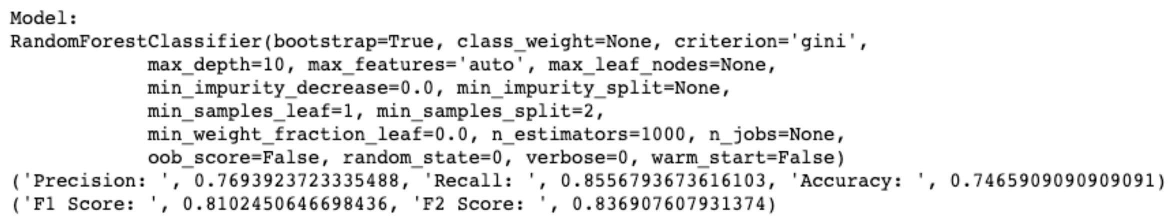
```
Model:
GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=10,
              max_features=None, max_leaf_nodes=None,
              min_impurity_decrease=0.0, min_impurity_split=None,
              min_samples_leaf=1, min_samples_split=2,
              min_weight_fraction_leaf=0.0, n_estimators=1000,
              n_iter_no_change=None, presort='auto', random_state=0,
              subsample=1.0, tol=0.0001, validation_fraction=0.1,
              verbose=0, warm_start=False)
('Precision: ', 0.9139690358902182, 'Recall: ', 0.9336808051761323, 'Accuracy: ', 0.9025)
('F1 Score: ', 0.9237197724039828, 'F2 Score: ', 0.9296707229778094)
```

success rate, accuracy, the key metric is really important and Gradient boost suits our need perfectly. It also has a great deal of flexibility like optimizing on different loss functions and provides many options when it comes to hyper parameter tuning further that makes the function fit very flexibly.

Used SKlearn gradient boosting algorithm with a learning rate of 0.1, max_depth =10 and 1000 estimators.
The model results are the following:
Gradient Boost accuracy score is: **0.90**

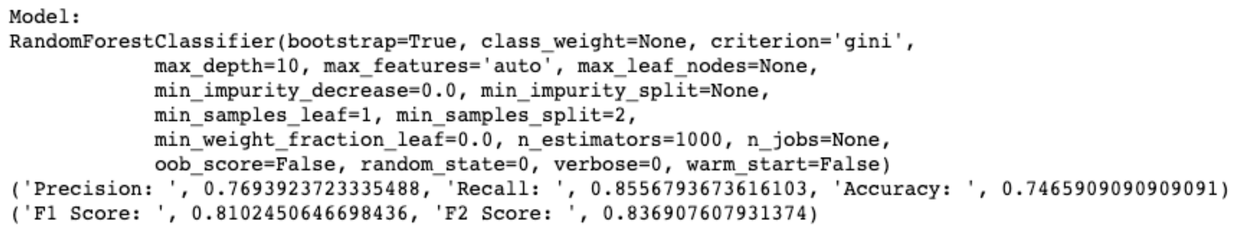
Model D. RandomForrest Classifier

I chose Random forrest algorithm to account into the favor of



```
Model:
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=10, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=1000, n_jobs=None,
            oob_score=False, random_state=0, verbose=0, warm_start=False)
('Precision: ', 0.7693923723335488, 'Recall: ', 0.8556793673616103, 'Accuracy: ', 0.7465909090909091)
('F1 Score: ', 0.8102450646698436, 'F2 Score: ', 0.836907607931374)
```

overfitting, as random forrest algorithm has a tendency to alleviate



```
Model:
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=10, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=1000, n_jobs=None,
            oob_score=False, random_state=0, verbose=0, warm_start=False)
('Precision: ', 0.7693923723335488, 'Recall: ', 0.8556793673616103, 'Accuracy: ', 0.7465909090909091)
('F1 Score: ', 0.8102450646698436, 'F2 Score: ', 0.836907607931374)
```

overfitting, while maintaining the accuracy our key metric. It works very well with categorical values and automates any missing value present i the data. Used SKlearn random forrest classifier algorithm with max_depth =10 and 1000 estimators. The model results are the following:

Random Forrest accuracy score is: **0.75**

Model Comparison

Gradient Boost is the best performing model with **90% accuracy score**.
(Precision:0.91, Recall: 0.93, Accuracy: 0.90, F1: 0.92, F2: 0.92)

	Logistic Regression	SVM	Gradient Boost	Random Forrest
Accuracy	0.69	0.71	**0.90**	0.75
Precision	0.71	0.75	**0.91**	0.77
Recall	0.85	0.81	**0.93**	0.85
F1	0.78	0.78	**0.92**	0.81
F2	0.82	0.80	**0.94**	0.84

Conclusion

The gradient boost is the best model and achieves an accuracy score of 0.90. This is definitely a great improvement with the optimal hyper parameters.

The current metric we have is 63% success rate. With our Gradient Boost algorithm the probability of the success rate is 90% and that would meet our business objective in terms of offer completion rate and eventually engage our customers creating both customer and business value.

Future Improvements

Curious to use neural network and other deep learning models in the future and find out the metrics and compare against the standard logistic regression algorithms.