Capstone Project Toni Magdy
Machine Learning Engineer Nanodegree 29th August 2020

Definition

Project Overview

- Once every few days, Starbucks sends out an offer to users of the mobile app. An offer can be merely an advertisement for a drink or an actual offer such as a discount or BOGO (buy one get one free).
- Starbucks needs a way to send to each costumer the right offer.

Problem Statement

- Our goal is to analyze historical app data to find most appropriate offer for costumers.
- The appropriate offer when the costumer sees the offer and buy the products under the offer influence.
- we aim to know if the user will complete the offer or not, we will build a model that predicts whether or not the user will complete the offer.

Metrics

It's a binary classification problem, so that below evaluation metrics should be able to determine the model performance

- Accuracy = $\frac{True\ Positives + True\ Negatives}{Total}$

Accuracy is how many points did we classify correctly.

- Precision = $\frac{True\ Positives}{True\ Positives + False\ Positives}$

Precision is proportion of positive cases that we classify correctly.

- Recall = $\frac{True\ Positives}{True\ Positives + False\ Negatives}$

Recall is proportion of actual positive cases the we classify correctly.

Analysis

Data Exploration

- The dataset is provided by Udacity and Starbucks.
- The program used to create the data simulates how people make purchasing decisions and how those decisions are influenced by promotional offers.
- The data is contained in three files.

Profile Data: **Rewards program users (17000 users x 5 fields)**

Demographic data about users

Profile data contains demographic data about 17000 users of Starbucks app

```
profile rows: 17000, profile columns: 5
```

Out[13]:

	age	became_member_on	gender	id	income
10412	85	20180404	F	503053089f114898b546bc6740d8e978	84000.0
14449	45	20180607	F	f2e49f5002c540eb92ca320fea990319	73000.0
16016	68	20171009	M	ab68c87257344ba7963064dd8b4b9350	33000.0
9449	118	20161031	None	c99a06c81f8540b49cb6a66719ea62dc	NaN
3302	60	20170302	M	7366bef4c288476dab78b09a33d0e692	52000.0
12484	118	20160727	None	b04385001db14fdf87829c6163ae9ddd	NaN
14313	98	20150403	M	75225655a1c44546a18f100f7c864f98	37000.0
15713	26	20180117	F	28416b56bdc94890a4996dd2dcc598b4	45000.0
5257	56	20180711	M	2d3c956111ad434786e39ed79354dd5a	66000.0
15232	118	20180525	None	270e7fd65f7e45c58b79d0d8ad2c72ab	NaN

Fig. 1 sample of profile data

Portfolio Data: **Offers sent during 30-day test period (10 offers x 6 fields)**

Contains data about offers

```
portfolio rows: 10, portfolio columns: 6
```

Out[12]:

	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

Fig. 2 sample of portfolio data

Transcript Data: **Event log (306648 events x 4 fields)**

records for transactions, offers received, offers viewed, and offers completed

```
transcript rows: 306534, transcript columns: 4
```

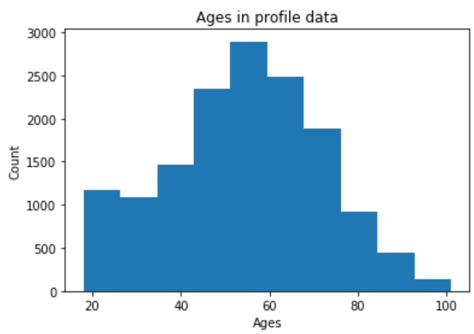
Out[11]:

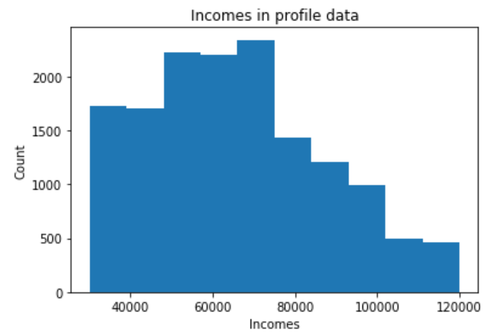
	event	person	time	value
39728	transaction	9b9bd320b3b34859abfee2109a0b4831	90	{'amount': 3.6}
50487	transaction	06b1031271174d8596c1996478f07ede	150	{'amount': 0.31}
246979	offer received	489f08a011894421991b8cc0e6e0a946	576	{'offer id': '2298d6c36e964ae4a3e7e9706d1fb8c2'}
146286	transaction	991386e4c20041428093919ed3c8f2ba	390	{'amount': 0.43}
15223	offer viewed	48225ea573e545e0b704ce3fcca8bb9e	0	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}
54706	offer received	055640cd12d04eb4b8a51ec67d451fc7	168	{'offer id': '2906b810c7d4411798c6938adc9daaa5'}
51845	transaction	ae0e47bc419940d68686ae364e73212b	156	{'amount': 2.24}
232039	transaction	796cc7c1e8534e78bdff45f9e11494d6	534	{'amount': 1.97}
22257	offer completed	e110e63527c24ad1b482f76acde24a42	18	{'offer_id': 'f19421c1d4aa40978ebb69ca19b0e20d...
7390	offer received	8cc0db430879405898d8390ca74ad13a	0	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}

Fig 3. sample of transcript data

Exploratory Visualization

Fig 4. Plots showing **Age** and **Income** distribution in profiles of users

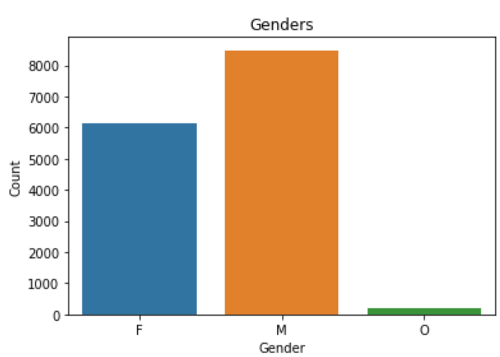


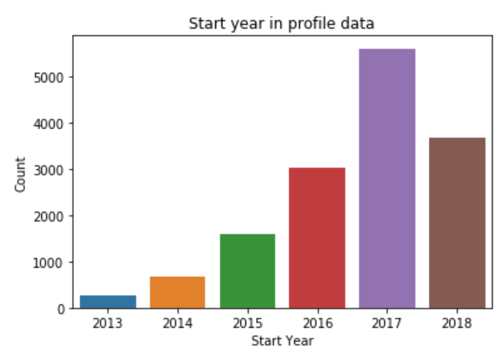


As showing in age plot ages is normally distributed and most ages are around (40-70).

In Income plot we see that most incomes are between 40K to 70K.

Fig 5. Plots showing **Gender** distribution and **start year** in profiles of users

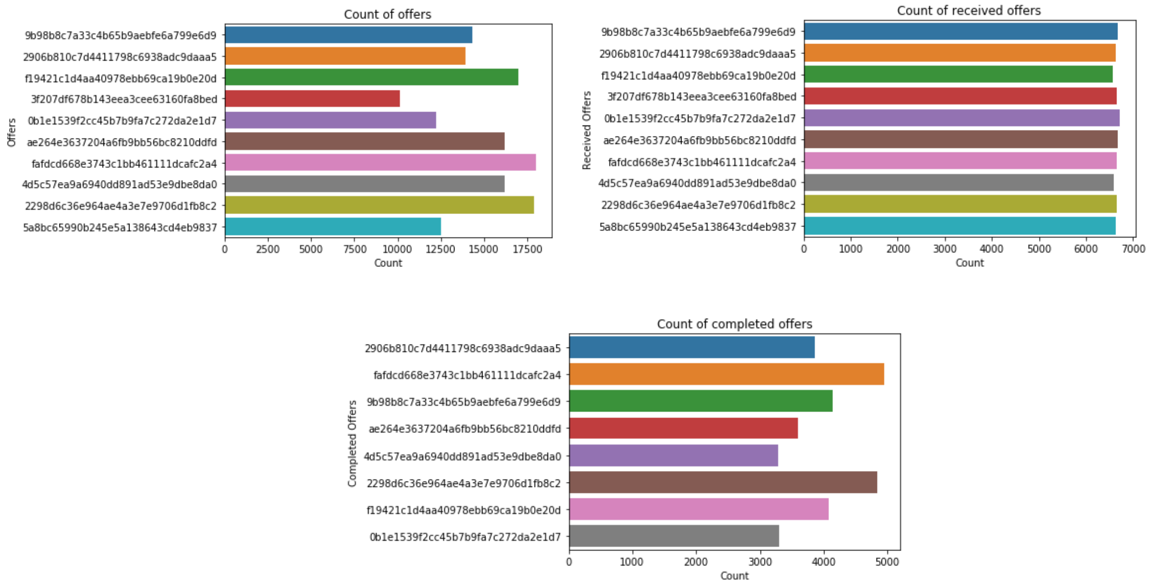




It can be seen that there are more **males** then **females**, and a little amount of **other** gender.

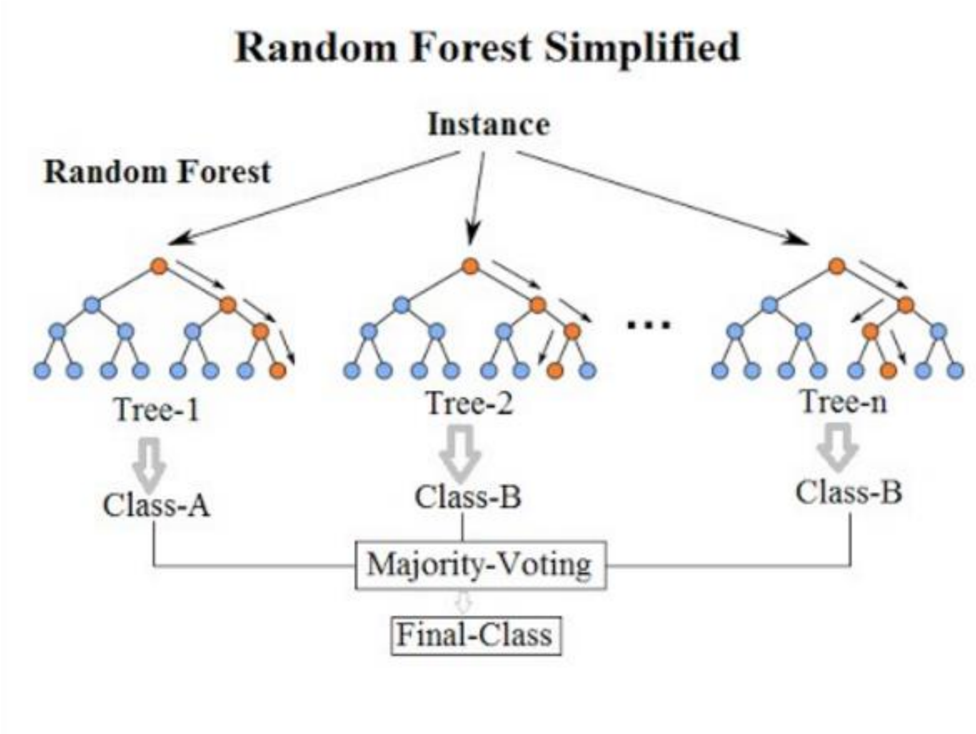
In start year plot the number of users is increasing from the start until the biggest number in **2017.**

Fig 6. Plots showing number of offers in transcript data



Algorithms and Techniques

Random Forest**:** Random forests are an ensemble learning method for classification that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the of the individual trees. Random forests correct for decision trees' habit of overfitting to their training set. (Wikipedia)



XGBoost: The XGBoost (eXtreme Gradient Boosting) is a popular and efficient open-source implementation of the gradient boosted trees algorithm. Gradient boosting is a supervised learning algorithm that attempts to accurately predict a target variable by combining an ensemble of estimates from a set of simpler and weaker models. The XGBoost algorithm performs well in machine learning competitions because of its robust handling of a variety of data types, relationships, distributions, and the variety of hyperparameters that you can fine-tune. (AWS docs)

LinearLearner: The Amazon SageMaker linear learner algorithm provides a solution for both classification and regression problems. With the Amazon SageMaker algorithm, you can simultaneously explore different training objectives and choose the best solution from a validation set. You can also explore a large number of models and choose the best. The best model optimizes either of the following:

- Continuous objectives, such as mean square error, cross entropy loss, absolute error.
- Discrete objectives suited for classification, such as F1 measure, precision, recall, or accuracy.

Compared with methods that provide a solution for only continuous objectives, the Amazon SageMaker linear learner algorithm provides a significant increase in speed over naive hyperparameter optimization techniques. It is also more convenient. (AWS docs)

Benchmark

We'll use Logistic Regression as benchmark model, it's simple and easy to implement.

Logistic Regression Results:

```
#evaluate the model
print("Logistic Regression Model: ")
evaluate_model(lr, X_test, y_test)
```

```
Logistic Regression Model:
Accuracy:  0.6970904443274941
Precision:  0.6657324089098687
Recall:  0.7190885914595284
```

Methodology

Data Preprocessing

In this section we'll clarify the steps of cleaning the data and prepare our data for modeling.

1 – **Profile Data**

There are 2175 incomplete rows in profile data with null values in gender and income and with age 118, we'll drop them.

We'll rename column id to customer_id

Change the became_member_on column to date format to be more readable and make columns for start year and start month.

Make column for each category of gender column.

The output should be like:

	gender	age	customer_id	became_member_on	income	start_year	start_month	male	female	other
11585	M	54	379bdc8a080b4b14b70d0468e6ce9a75	2014-02-22	79000.0	2014	2	1	0	0
8553	F	47	5dfdad4241764dfe959f51b7460e42b1	2014-06-20	97000.0	2014	6	0	1	0
8223	F	82	0ecc8a63f8ce437aaec064b14cd5813f	2016-06-04	81000.0	2016	6	0	1	0
8669	F	19	f052f7c3f89044f9bb7097a72e62101c	2018-04-10	55000.0	2018	4	0	1	0
13632	M	51	afbd0cb9440b45d696ae1fd2874de803	2017-08-09	114000.0	2017	8	1	0	0
10501	M	21	7a30763c05734cb3bd14123e1b6b9d63	2014-06-11	73000.0	2014	6	1	0	0
18	M	57	6445de3b47274c759400cd68131d91b4	2017-12-31	42000.0	2017	12	1	0	0
1279	M	30	ecd3afe895454a42867d70f0f4be7015	2013-10-13	41000.0	2013	10	1	0	0
9472	M	49	31aaa0fa063549759eed0d800c5a26bb	2018-01-31	51000.0	2018	1	1	0	0
13618	F	68	ed89ba0d99a14e71ae5ad56768df5662	2018-05-08	57000.0	2018	5	0	1	0

2 – **Portfolio Data**

We'll rename id column to offer_id.

We'll make column for each category of channels, and offer_type.

The output should be like:

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

3 – **Transcript Data**

First, we'll split the value column to two columns, offer_id, and amount.

We'll rename person column to customer_id.

Since we dropped some profiles, those profiles may appear here, we need to make sure to drop any customer_id that not in profile data.

Now our data have two types of events, transaction, and offers (viewed, received and completed), we need to split our data into two parts.

We have transaction data and offers data, we'll make column for each category in offer_type

The output should be like:

	customer_id	event	time	offer_id	amount	offer_received	offer_viewed	offer_completed
11407	be01dde700574797b5dc59b0ad45242f	offer completed	0.0	2298d6c36e964ae4a3e7e9706d1fb8c2	0.0	0	0	1
258374	13853a0e60ad42bea59e490fb39b0044	offer completed	27.0	9b98b8c7a33c4b65b9aebfe6a799e6d9	0.0	0	0	1
203680	bec9d368f0a54822998cebebb6fb35ec	offer completed	22.0	ae264e3637204a6fb9bb56bc8210ddfd	0.0	0	0	1
3234	b94e988ed63a498aab070a6a64458812	offer received	0.0	5a8bc65990b245e5a138643cd4eb9837	0.0	1	0	0
136251	ba4fff69b9224b87a1916f9fd9d0c2c8	offer received	17.0	9b98b8c7a33c4b65b9aebfe6a799e6d9	0.0	1	0	0
254755	83f7a1c222b240efa169e3ce318460af	offer viewed	26.5	3f207df678b143eea3cee63160fa8bed	0.0	0	1	0
3513	34b216d046e74a53adf47e99b819c882	offer received	0.0	0b1e1539f2cc45b7b9fa7c272da2e1d7	0.0	1	0	0
148125	737789ee6eb143639c8dd7c58ab40253	offer completed	17.0	f19421c1d4aa40978ebb69ca19b0e20d	0.0	0	0	1
23403	cb96e2ccccca4921bca1de8f5f5e51e4	offer completed	1.5	2906b810c7d4411798c6938adc9daaa5	0.0	0	0	1
215046	76f2d0fd5c084f6e9ef847c096225fcf	offer completed	23.5	2298d6c36e964ae4a3e7e9706d1fb8c2	0.0	0	0	1

	customer_id	time	amount
98545	cbbde640385f454fb4a81f301fe08c5a	13.75	26.31
157035	f223c55edc8849698b4e901ffc14b23d	17.75	4.84
161164	c325c5e0684044d69ed99953fcd703ca	18.25	2.90
42017	8b187db07b274a9583550f15a397c0d0	5.25	30.89
21624	72c2e3e12e4e42ccabcff144c2a30a84	1.25	1.01
69830	4f3aab9a035e4e73b8b6842031ccddb4	8.00	8.96
84372	29655cba61a04d079efde18203b4f232	10.25	27.31
77221	ced3380d8a334054ab0f389572485292	9.00	7.39
215542	59117e97e0424ab89455dc9607b9b7e7	23.50	10.70
46823	d88b4dbc2ff54dbcb9b99db4802657ac	6.75	1.69

Combine Data

Now we've cleaned our data, we need to combine all of this in order to make one dataset to train our models with.

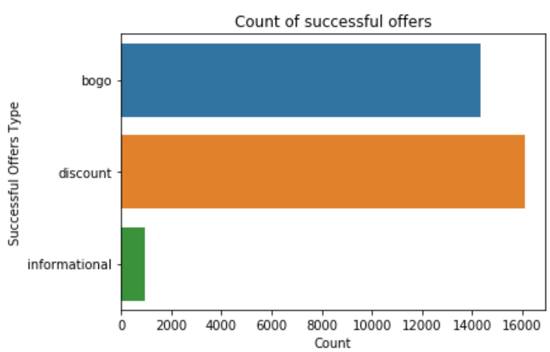
In the combined data each row should have offer information and customer demographic data and whether or not the offer was successful.

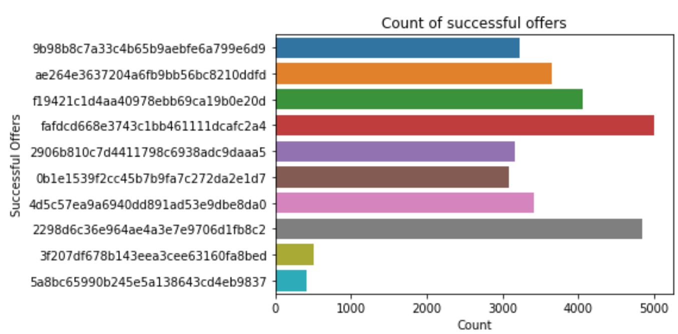
The combined data contains 66501 record.

```
[49]: combined_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 66501 entries, 0 to 66500
Data columns (total 25 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   success          66501 non-null  int64
 1   offer_id         66501 non-null  object
 2   customer_id      66501 non-null  object
 3   time             66501 non-null  float64
 4   spent            66501 non-null  float64
 5   reward           66501 non-null  int64
 6   difficulty       66501 non-null  int64
 7   duration         66501 non-null  int64
 8   offer_type       66501 non-null  object
 9   web              66501 non-null  int64
 10  email            66501 non-null  int64
 11  mobile           66501 non-null  int64
 12  social           66501 non-null  int64
 13  bogo             66501 non-null  int64
 14  discount         66501 non-null  int64
 15  informational    66501 non-null  int64
 16  gender           66501 non-null  object
 17  age              66501 non-null  int64
 18  became_member_on 66501 non-null  datetime64[ns]
 19  income           66501 non-null  float64
 20  start_year       66501 non-null  int64
 21  start_month      66501 non-null  int64
 22  male             66501 non-null  int64
 23  female           66501 non-null  int64
 24  other            66501 non-null  int64
```

Here's the count of successful offers in our data.





Then we'll drop columns customer_id, became_member_on, offer_id, gender, and offer_type.

The last step of preprocessing the data, there are some numeric features the need to be scaled, we'll apply MinMaxScaler to columns **time, spent, reward, difficulty, duration, age, income.**

And finally, we'll shuffle our data and split it to 80% train and 20% test to start building models.

Implementation

The implementation of the 3 algorithms we mentioned in Algorithms and techniques section.

1 – Random Forest

Random forest model is an ensemble learning model made up of many decision trees, it should give great results for our problem.

Random Forest model was trained

We trained a random forest model with **max_depth = 10**

2 – SageMaker XGBoost Model

we split the training data into training and validation sets to validate the model.

After splitting the data, we made a csv files and uploaded it to s3.

```
In [235]: from sagemaker.amazon.amazon_estimator import get_image_uri
          container = get_image_uri(sagemaker_session.boto_region_name, 'xgboost','0.90-1')

          #create xgb estimator
          xgb = sagemaker.estimator.Estimator(container,
                                              role,
                                              train_instance_count = 1,
                                              train_instance_type = 'ml.m4.xlarge',
                                              output_path = 's3://{}/{}/output'.format(sagemaker_session.default_bucket(), prefix),
                                              sagemaker_session = sagemaker_session)

          #set the hyperparameters
          xgb.set_hyperparameters(max_depth = 2,
                                  eta = 0.02,
                                  gamma = 3.60,
                                  min_child_weight = 2,
                                  subsample = 0.70,
                                  objective = 'binary:logistic',
                                  early_stopping_rounds = 100,
                                  num_round = 400)
```

Then we'll make batch transform job to test our model performance.

3 – SageMaker LinearLearner

We created a record set to train the model with, then we created our model and trained it with our formatted data.

```
In [220]: #make LinearLearner model
          from sagemaker import LinearLearner
          prefix = 'linear'
          linear = LinearLearner(role = role,
                                 train_instance_count = 1,
                                 train_instance_type = 'ml.m4.xlarge',
                                 predictor_type = 'binary_classifier',
                                 output_path = 's3://{}/{}/output'.format(sagemaker_session.default_bucket(), prefix),
                                 sagemaker_session = sagemaker_session,
                                 epochs = 50)
```

```
In [222]: #change the values of the data to numpy array, then create a record set to train the model
          train_x_np = X_train.values.astype('float32')
          train_y_np = y_train.values.astype('float32')
          formatted_train_data = linear.record_set(train_x_np, labels = train_y_np)
```

```
In [223]: #train the model
          linear.fit(formatted_train_data)
```

Refinement

We'll improve our models with hyperparameter tuning, we'll tune the XGBoost model with sagemaker tuner.

```
In [236]: from sagemaker.tuner import IntegerParameter, ContinuousParameter, HyperparameterTuner


          #make hyperparameter tuner to tune the model with hyperparamters ranges,
          #we'll train 6 different models and our goal is to maximize accuracy.
          xgb_tuner = HyperparameterTuner(estimator = xgb,
                                          objective_metric_name = 'validation:auc',
                                          objective_type = 'Maximize',
                                          max_jobs = 6,
                                          max_parallel_jobs = 3,
                                          hyperparameter_ranges = {
                                              'max_depth': IntegerParameter(2, 4),
                                              'eta': ContinuousParameter(0.02, 0.04),
                                              'gamma': ContinuousParameter(3.60, 3.65),
                                              'min_child_weight': IntegerParameter(1, 2),
                                              'num_round': IntegerParameter(400, 700),
                                              'subsample': ContinuousParameter(0.70, 0.73),
                                          })
```

Best training job with parameters:

eta: 0.037, gamma: 3.63, max_depth: 4, min_child_weight: 2, num_round: 699 subsample: 0.72

Results

Model	Accuracy	Precision	Recall
Random Forest	0.91.5	0.88	093
XGBoost	0.91.8	0.90	0.92
LinearLearner	0.88	0.87	0.89
Logistic Regression	0.69	0.65	0.73

Justification

At the end we can say that our three models is better than the benchmark model, the best model was the XGBoost model after tuning with slightly better results than random forest model.