Machine Learning Engineer Nanodegree

Capstone Project

Mengjia Lyu
September 10, 2020

I. Definition

Project Overview

Starbucks is the largest coffeehouse chain in the world. With its large app user base, targeted advertising can effectively increase the frequency of transactions per customer. Business insights are invaluable when it comes to targeted advertising.

In order to maximize the efficacy of app advertising, we need to predict with reasonable accuracy what kinds of offers are most likely to lead to purchases for different customers. Machine learning algorithms can be employed to learn from data on transactions, customer demographics and offers sent.

Problem Statement

The problem can be framed as a binary classification problem, whose task is to predict if a given customer would respond to a certain offer.

The inputs/predictors are the demographic, transactional and profiling data associated with a given customer and an offer, whose detail is shown in latter sections. The goal/target is to predict the probability that the customer actually applies the offer, and adjust advertising strategies based on such information to maximize efficiency. For a given customer, we can selectively send him or her the offers that output the highest probability by the algorithm.

By building a predictive algorithm that can best predict the likelihood of a customer's responding to an offer based on key predictors, Starbucks will learn the rules for targeted advertising and be better equipped to gain maximum return on its online promotions.

Datasets and Inputs

The datasets are provided by Starbucks and can be accessed from the Jupyter notebook console.

More specifically, the inputs/predictors are:

- difficulty
- age
- income
- duration
- reward
- membership_time
- Dummy variables
- offer_bogo
- offer_discount
- offer_informational

- channel_email
- channel_mobile
- channel_social
- channel_web
- gender_female
- gender_male
- gender_other

Our output/target variable is:

- outcome

Metrics

For any binary classification problem, accuracy is an important metric as it measures the proportion of correct predictions.

We will use also Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores to measure the performance of a model or result in my project. ROC AUC can measure how well predictions are ranked by the probabilities^[See Reference [9]].

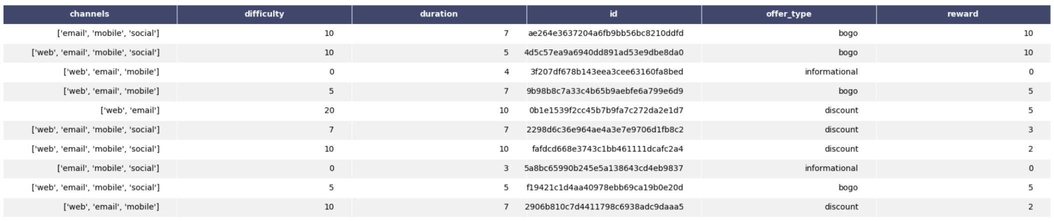
II. Analysis

Data Exploration

As with all maching learning problems, everything starts with data. Three datasets are provided by Starbucks for this problem:

1.The ***portfolio*** dataset

- `portfolio` contains information on offers and includes the following fields:
    - `id:` offer id represented by a string
    - `offer_type:` the specific offer type that encompasses
        - BOGO (Buy one get one free)
        - informational (for example, seasonal products)
        - discount
    - `difficulty:` the minimum amount of money required to spend for the offer to take effect
    - `reward:` the dollar value rewarded for completing an offer
    - `duration:` the shelf time of the offer
    - `channels:` the channels via which the offers are sent, encompassing mail, mobile, social and web
- first 10 rows sample of the dataset

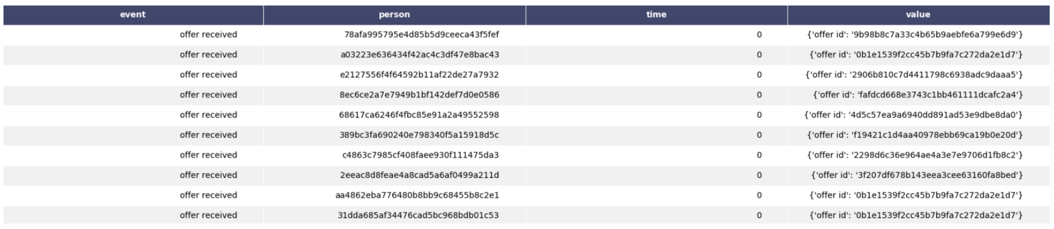


channels	difficulty	duration	id	offer_type	reward
['email', 'mobile', 'social']	10	7	ae264e3637204a6fb9bb56bc8210ddfd	bogo	10
['web', 'email', 'mobile', 'social']	10	5	4d5c57ea9a6940dd891ad53e9dbe8da0	bogo	10
['web', 'email', 'mobile']	0	4	3f207df678b143eea3cee63160fa8bed	informational	0
['web', 'email',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

- After checking for missing values, we find out that the `portfilio` dataset does not contain any missing values.
- As `channel` is a categorical variables, it is one-hot encoded to dummy variables `channel_email` , `channel_mobile` , `channel_social` and `channel_web` .
- As `offer_type` also is a categorical variables, it is one-hot encoded to dummy variables `offer_bogo` , `offer_dicount` and `offer_informational`

2.The *transcript* dataset

- `transcript` contains information on customers' transaction history in one month and includes the following fields:
    - `value:` dictionaries whose key is either `offer id` or `amount` and whose value represents the corresponding offer id or amount of money incurred in the transaction
    - `event:` the event of encompassing four types
        - transaction
        - offer received
        - offer viewed
        - offer completed
    - `person:` customer id represented by a string
    - `time:` the time in hours since start of the test
- first 10 rows sample of the dataset



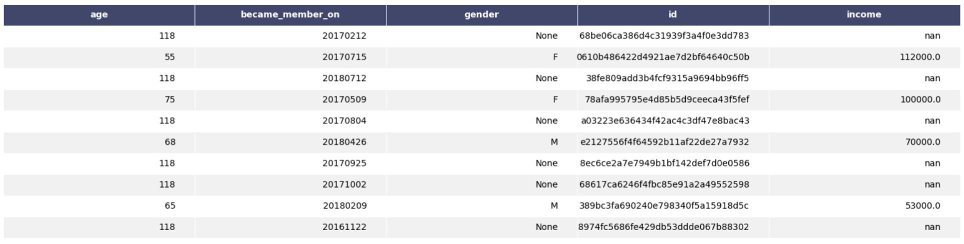
event	person	time	value
offer received	78afa995795e4d85b5d9ceeca43f5fef	0	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}
offer received	a03223e636434f42ac4c3df47e8bac43	0	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}
offer received	e2127556f4f64592b11af22de27a7932	0	{'offer id': '2906b810c7d4411798c6938adc9daaa5'}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

- After checking for missing values, we find out that the `portfolio` dataset does not contain any missing values.
- As `event` is a categorical variable, it is one-hot encoded to dummy variables.
- As `value` is a list of dictionaries, it is unpacked according to the specific key.
- - if the key is an offer, the value is moved to a new column
- - if the key is a transaction, the value is in a new column

3.The *profile* dataset

- `profile` contains customers' demographic information and includes the following fields:
    - customer id represented by a string
    - the age of the customer
    - the date when customer created an app membership account
    - the gender of the customer
    - customer's income

    - first 10 rows sample of the dataset



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

- As can be seen from the sample, the dataset contains missing values.
- After checking for missing values, we find out that the `portfilio` dataset contains 2175 missing values for both `income` and `gender` variables, making up 12.79% of the tatal data.
    - To deal with missing data, normally list-wise deletion or imputation is adopted. Upon a closer look, I discover that all customers whose `income` and `age` are missing have an age of 118 years, which is cleary a made-up number. Using t-test, it can be found out that the missing values are randomly distributed across all observations. Therefore we remove all rows with NaN values.
- The variable `became_member_on` is converted to `membership_time` which denotes the number of days the customer has been a Starbucks member.
- As `gender` is a categorical variable, it is one-hot coded into three dummy variables `gender_female`, `gender_male` and `gender_other`.

Lastly we check for extreme outliers. According to the definition^[Stats: Measures of Position] that outliers are more than 3.0 times the interquartile range below the first quartile or above the third quartile, no extreme outliers are found.

We want to find how to best predict whether an offer would be responded to based on demographic, profiling and portfolio data. Therefore we combine all three datasets in the end. To make a more educated, we only consider offers that have been **viewed**. In this case, we ensure that the customer is conscious of the offer details and actively make a decision as to whether to respond to the offer or not.

Our predictors are:

- Continuous variable
    - difficulty
    - age
    - income
    - duration
    - reward
    - membership_time
- Dummy variables
    - offer_bogo
    - offer_discount
    - offer_informational
    - channel_email
    - channel_mobile

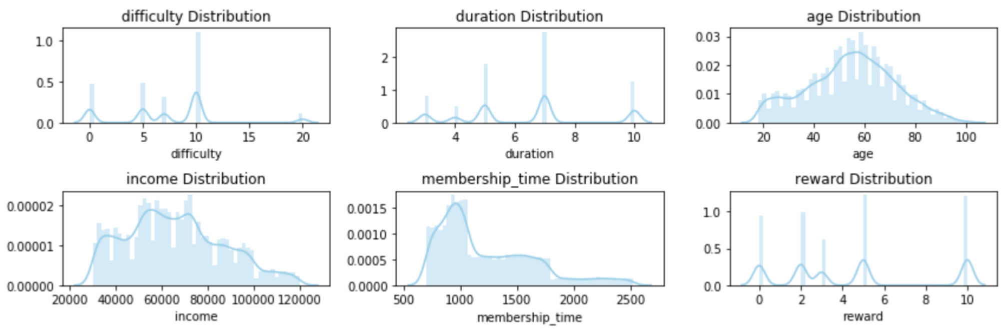
    - channel_social
    - channel_web
    - gender_female
    - gender_male
    - gender_other

Our target variable is:

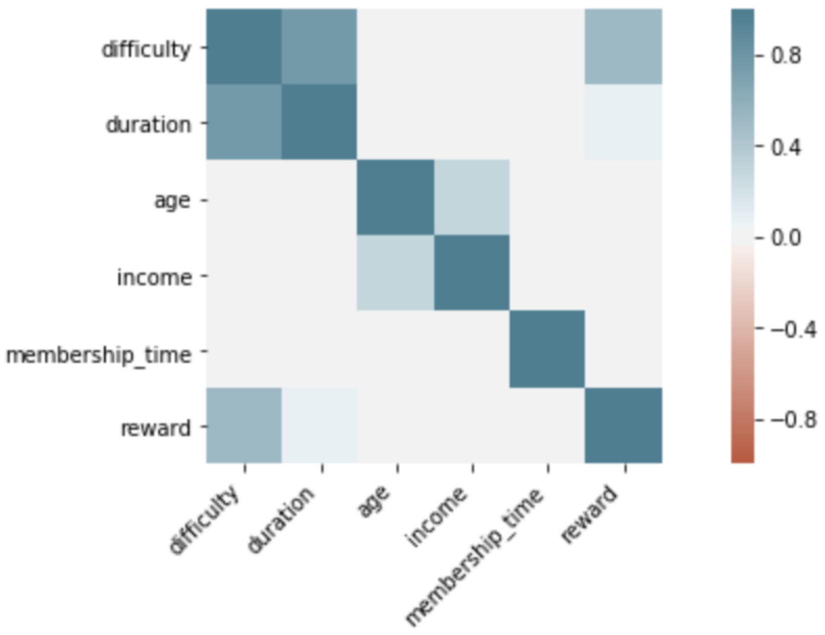
- outcome

Exploratory Visualization

The dataset is visualized through histograms.



The features `age`, `income` and `membership_time` have relatively smooth and Gaussian like distributions. Other features like `difficulty`, `duration` and `reward` cannot be approximated as Gaussian, which we wil take account when considering algorithms.



Looking at the correlation pairplot, we can see that two feature pairs seem to be significantly correlated between each other:

1. `difficulty` and `duration`. 2. `difficulty` and `reward`.

Algorithms and Techniques

I start with the Naive Bayes algorithm which is also our benchmark model. Then I move on to Decision Trees, Support Vector Machines, Random Forests, Adaboost and XGBoost. The final classifier is a XGBoost classifier,

The measurements of performance used are accuracy score and Area Under the Receiver Operating Characteristic Curve (ROC AUC).

Benchmark

I used Naive Bayes Classifier from *sklearn* as our benchmark model. A naive bayes classifier *naively* assumes that features in the dataset are mutually independent, and can be trained easily and fast.

As there isn't any related literature in the problem, a simple classifier like naive bayes classifier makes a great candidate for benchmarking other more sophisticated model.

The Naive Bayes classifier achieved an accuracy of 74.28%.

III. Methodology

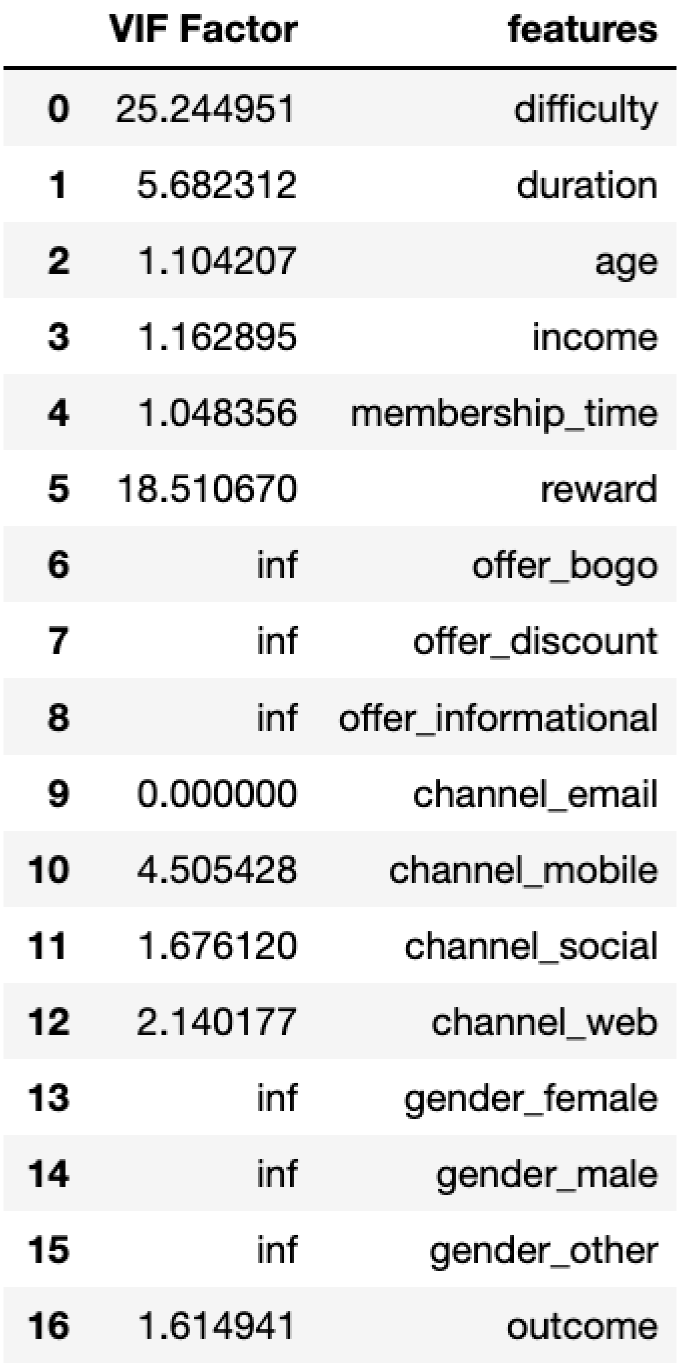
Data Preprocessing

The steps of data preprocessing done in the "Starbucks_Project" notebook are as follows:

Feature Selection

After cleaning and combining the three datasets, I seek to identify outliers, which can be defined as +/- 3*IQR deviations from the the Q1/Q3. In code I define a function `remove_outlier` to remove outliers one feature at a time.

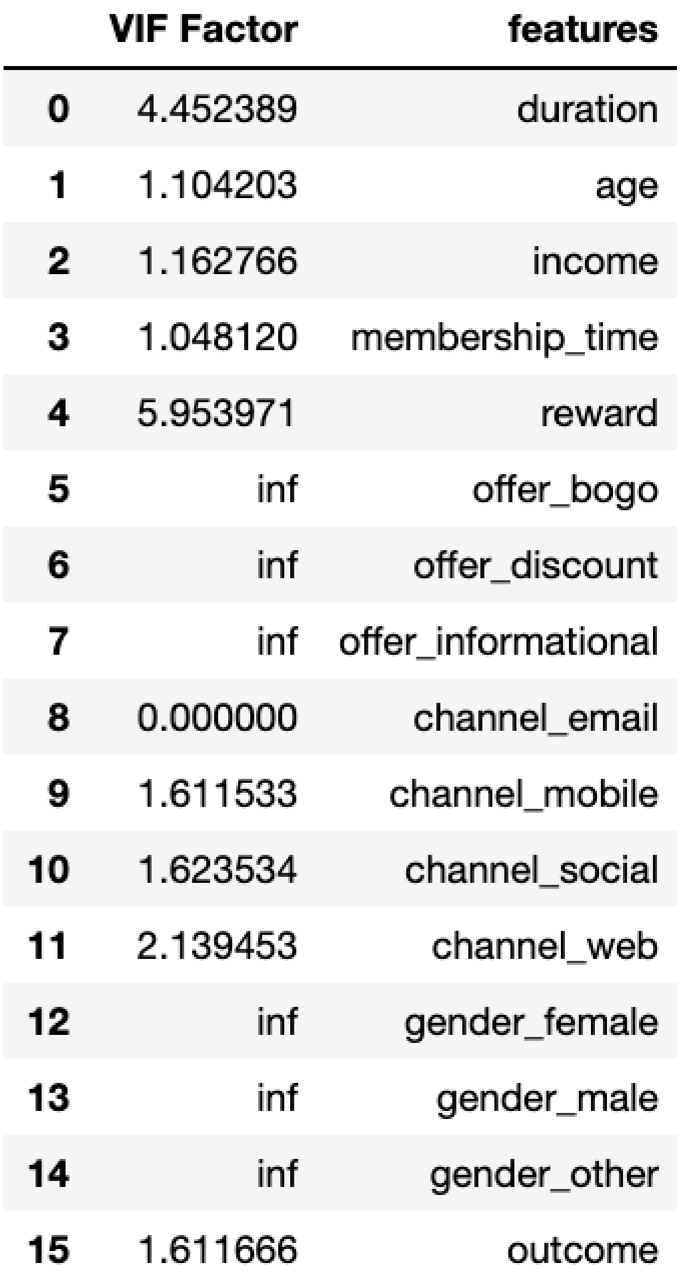
To select the most relevant features, I rank the features based on the variance inflation factor(VIF). VIF effectively measures the multicollinearity between variables. A rule of thumb is: a VIF > 10 indicates high multicollinearity^[See



	VIF Factor	features
0	25.244951	difficulty
1	5.682312	duration
2	1.104207	age
3	1.162895	income
4	1.048356	membership_time
5	18.510670	reward
6	inf	offer_bogo
7	inf	offer_discount
8	inf	offer_informational
9	0.000000	channel_email
10	4.505428	channel_mobile
11	1.676120	channel_social
12	2.140177	channel_web
13	inf	gender_female
14	inf	gender_male
15	inf	gender_other
16	1.614941	outcome

Reference [12]].

We first remove the one with the highest VIF-- `difficulty` -- and calculate the VIFs again.



	VIF Factor	features
0	4.452389	duration
1	1.104203	age
2	1.162766	income
3	1.048120	membership_time
4	5.953971	reward
5	inf	offer_bogo
6	inf	offer_discount
7	inf	offer_informational
8	0.000000	channel_email
9	1.611533	channel_mobile
10	1.623534	channel_social
11	2.139453	channel_web
12	inf	gender_female
13	inf	gender_male
14	inf	gender_other
15	1.611666	outcome

After removing the `difficulty`, VIF for `reward` is brought down significantly. All VIF factor values are under 10, and most other continuous variables have a VIF closed to the desired value 1. Therefore we can stop here.

Feature Transformation

Performing feature scaling can improve numerical stability and resolution and reduce training time^[See reference [2]]. Also, some machine learning algorithms are sensitive to feature scaling^[See reference [4]]. Since our data does not follow a Gaussian distribution, we use normalization here rather than standardization.

Using the `MinMaxScaler` function from `sklearn.preprocessing` library, we can apply min-max normalization on the features.

Split into Training/Validation Sets

We use the `train_test_split` function from the `sklearn.model_selection` library.

Implementation

I implemented seven different algorithms in total, including Naive Bayes, Decision Tree model, Support Vector Machine, Random Forest, Adaboost, Neural Network and XGBoost, which is our final choice.

Refinement

An initial solution is obtained by fitting a "naive" XGBOOST model without any parameter tuning. Its accuracy score stands at 80.44%, and from there begins the refinement process. The refinement is gradually achieved through hyparameter tuning.

In this project, I search for the best hyperparameters following the guide in *Complete Guide to Parameter Tuning in XGBoost with codes in Python*^[See Reference 7].

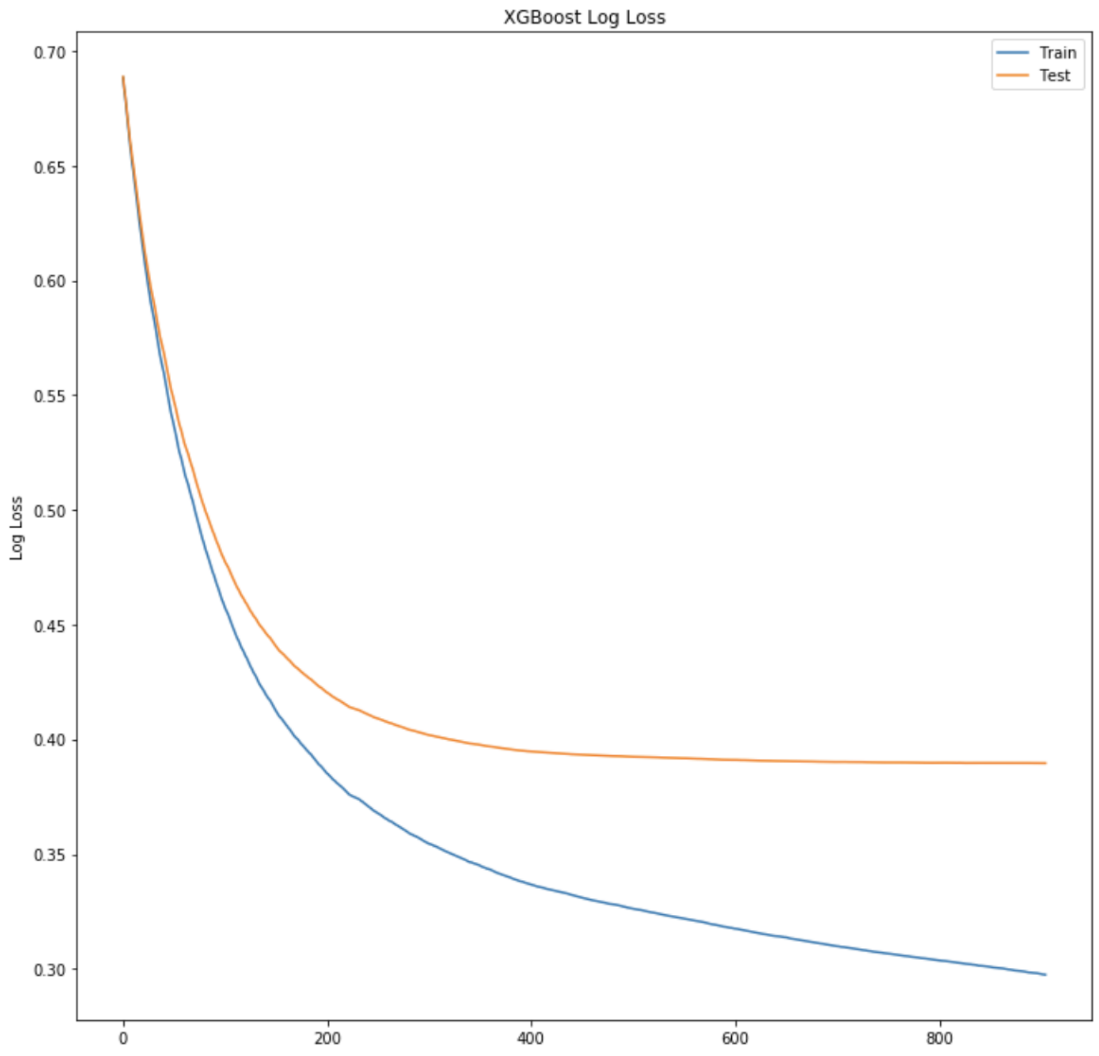
First I start training with minimum parameter settings and add range of parameters for further optimization in the following order: 1.`max_depth` and `min_child_weight`:We tune them together in order to find a good trade-off between model bias and variance. 2.`gamma` 3.`subsample` and `colsample_bytree` 4.`reg_alpha`

The number of epochs over which no improvement is observed is specified in the `early_stopping_rounds` parameter, which enforces Validation error needs to decrease at least every to continue training. To acquire a faster and more fitted response, early stopping techniques to stop model training.

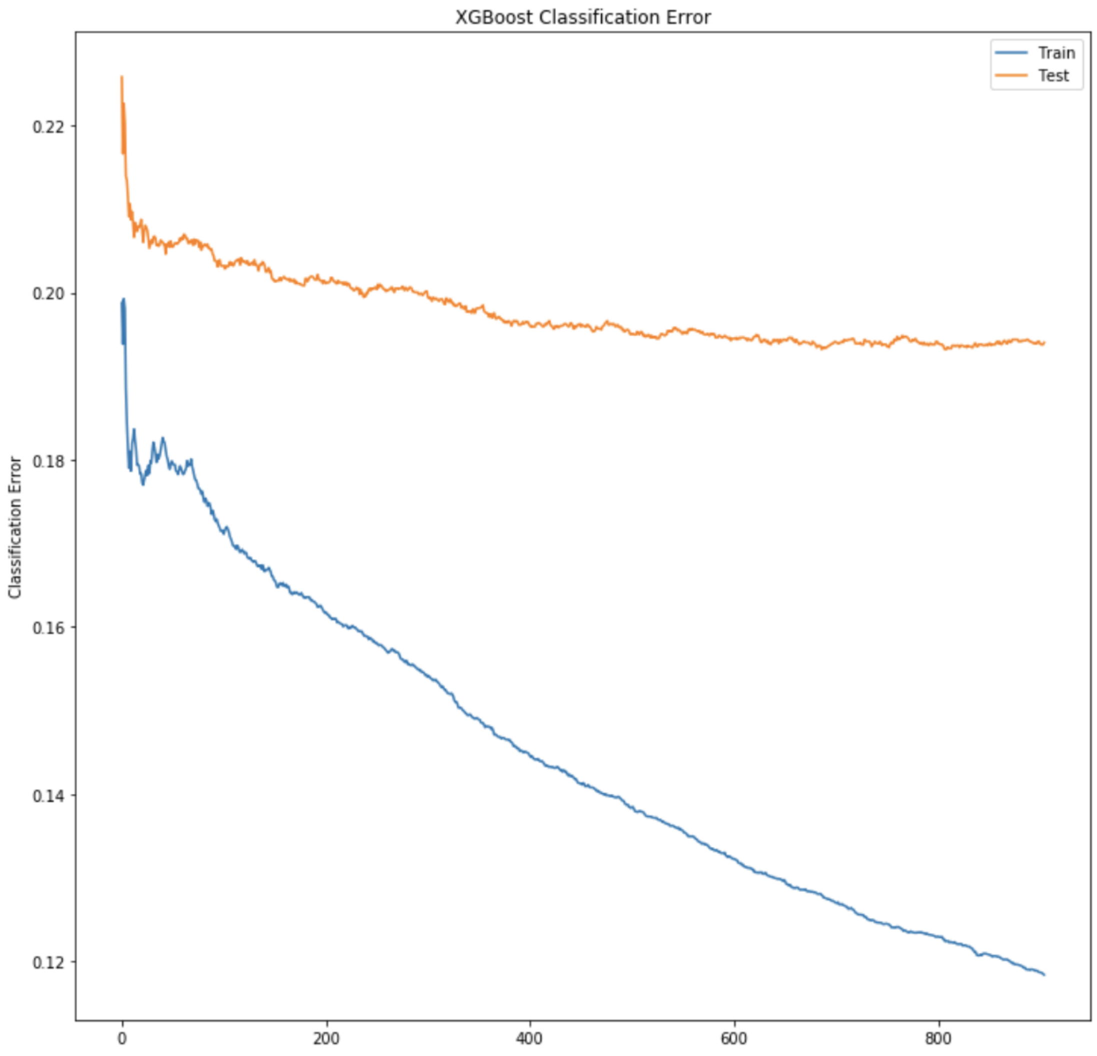
IV. Results

Model Evaluation and Validation with Free-Form Visualization

The following graph shows the logarithmic loss of the XGBoost model for each epoch on the training and test datasets.



The following plot shows the classification error of the XGBoost model for each epoch on the training and test datasets.



We can see that the log loss for unseen test data is as low as 0.4, and classification error for unseen test data is as low as 0.2.

With such low log loss and classification error in test data, we can conclude that the final model is robust and generalizes well to unseen data. Small perturbations in training data or the input space do not greatly affect the results. Therefore results from our model can be trusted to be reasonable and aligned with solution expectations, with appropriate final parameters.

Justification

The final model has: Accuracy : 0.8799 AUC Score (Train): 0.953107

The benchmark has: accuracy: 0.7433513767945399

The final results are stronger than the benchmark reported and hence is significant enough for the problem.

V. Conclusion

Reflection

The entire process familiarizes me with the machine learning workflow: 1.Data Collection 2.Data pre-processing 3.Model exploration 4.Model training and testing 5.Evaluation

The interesting aspect of the project lie in model exploration. It is fun trying various algorithms and compare them against each other.

The difficult aspects of the project lie in tuning the `max_depth` and `min_child_weight` parameters for XGBoost. It turns out that when I try to fine-tune the hyperparameters `max_depth` and `min_child_weight`, the process took too long and my computer almost got stuck. Fortunately it eventually went through.

Improvement

Under gradient boosting, I cannot think of any other ways for optimal improvement. Other boosting algorithms, such as LightBGM, use the same boosting paradigm, and therefore would not make a big difference in terms of performance

Through research, I got to know other methods outside of gradient boosting, such as regularized greedy forests and factorization machines. Both could be applied to binary classification problems, and have already garnered momentum in the machine learning community. I would consider using those algorithms if I knew how to implement it.

References [1] Quick and Easy Data Analysis for Your Machine Learning Problem [2]Understanding Data Normalization in Machine Learning [3]Tour of Data Preparation Techniques for Machine Learning [4]Feature Scaling for Machine Learning [5]XGBoost hyperparameter tuning in Python using grid search [6]Selecting Optimal Parameters for XGBoost Model Training [7]Complete Guide to Parameter Tuning in XGBoost with codes in Python [8]Stats: Measures of Position [9]MLmuse: Correlation and Collinearity — How they can make or break a model [10]The 5 Classification Evaluation Metrics Every Data Scientist Must Know [11]Avoid Overfitting By Early Stopping With XGBoost In Python [12]What in the World Is a VIF?