Table of Contents

[Candidate Choice Algorithms 1](#_Toc83413025)

[Logistic Regression 2](#_Toc83413026)

[Random Forest 2](#_Toc83413027)

[Gradient Boosting 2](#_Toc83413028)

[Stochastic Gradient Descent 2](#_Toc83413029)

[Conclusion based on Training various models 3](#_Toc83413030)

[Hyperparameter tuning of chosen Algorithms 5](#_Toc83413031)

[Training the tuned models 6](#_Toc83413032)

# Candidate Choice Algorithms

The following algorithms were used on the final dataset. A total of 4 feature sets were created apart from a set containing all candidate columns.

1. Feature set obtained from the top 7 features of Random Forest Classifier.

***Feature\_set\_1***: ['transactions\_amount', 'count\_pay\_attempt', 'nunique\_beacon\_type', 'count\_user\_stay', 'count\_buy\_click', 'profile\_submit\_count', 'sum\_beacon\_value']

1. Feature set obtained from the top 7 features of Logistic Regression.

***Feature\_set\_2***: ['count\_pay\_attempt', 'nunique\_device', 'nunique\_report\_type', 'count\_buy\_click', 'nunique\_beacon\_type', 'count\_sessions', 'nunique\_language']

1. Feature sets obtained from Correlation Analysis.

***Feature\_set\_3***: ['sum\_beacon\_value', 'count\_pay\_attempt', 'count\_buy\_click', 'nunique\_dob', 'nunique\_language', 'nunique\_report\_type', 'nunique\_device', 'transactions\_amount']

***Feature set 4***: ['sum\_beacon\_value', 'count\_pay\_attempt', 'count\_buy\_click', 'nunique\_report\_type', 'nunique\_device', 'transactions\_amount']

The feature sets were scaled using **Max Absolute Scaler** which scaled each feature by its maximum absolute value and **Standard Scaler** which standardized data by removing the mean and scaling to unit variance.

## **Logistic Regression**

Reasons: -

* The problem is a binary classification problem.
* Logistic Regression helps solve classification and **probability problems** i.e. it not only classifies the dependent but also gives us an estimated probability value of the classification belonging to the positive/negative class.
* The algorithm also yields importance scores of all features which can help us make a more efficient choice by choosing the top few features.

## **Random Forest**

Reasons: -

* The primary reason and the one of the greatest qualities of Random Forest is that it is very easy to measure the relative **feature importance** of each feature on the prediction.
* Random Forest prevents overfitting because it creates decision tress on subset of data.
* It has several hyperparameters which one can tune to get a better predictive model.

## **Gradient Boosting**

Reasons: -

* It is a generalised algorithm that works well for any classification task. Its predictive scores are often better than other the scores of other algorithms.
* It has several hyperparameters which can be tuned to get a better predictive model.

## **Stochastic Gradient Descent**

Reasons: -

* Suggested by domain expert.

# Conclusion based on Training various models

The following is a list of ***Top 20*** models chosen out of various experiments done with our final dataset.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Model | N(features) | BAC\_test | RCC\_test | BAC\_train | RCC\_train | Fit\_time | Score\_time |
| GB feature\_set\_1 StdScale | 7 | 0.988 | 0.998 | 0.989 | 0.999 | 12.657 | 0.063 |
| GB feature\_set\_1 | 7 | 0.988 | 0.998 | 0.989 | 0.999 | 14.884 | 0.112 |
| GB feature\_set\_1 MaxAbs | 7 | 0.988 | 0.998 | 0.989 | 0.999 | 12.639 | 0.063 |
| RF all features | 13 | 0.986 | 0.994 | 0.996 | 0.999 | 9.485 | 0.372 |
| RF feature\_set\_1 | 7 | 0.986 | 0.994 | 0.996 | 0.999 | 10.112 | 0.5 |
| GB feature\_set\_4 StdScale | 6 | 0.986 | 0.999 | 0.986 | 1 | 8.578 | 0.064 |
| GB feature\_set\_4 MaxAbs | 6 | 0.986 | 0.999 | 0.986 | 1 | 8.59 | 0.064 |
| GB feature\_set\_3 StdScale | 8 | 0.986 | 0.999 | 0.986 | 0.999 | 9.42 | 0.063 |
| GB feature\_set\_3 MaxAbs | 8 | 0.986 | 0.999 | 0.986 | 0.999 | 9.458 | 0.064 |
| SGD feature\_set\_1 | 7 | 0.984 | 0.995 | 0.985 | 0.995 | 0.268 | 0.049 |
| RF feature\_set\_3 StdScale | 8 | 0.984 | 0.994 | 0.991 | 0.999 | 6.882 | 0.349 |
| RF feature\_set\_3 MaxAbs | 8 | 0.984 | 0.994 | 0.991 | 0.999 | 6.897 | 0.349 |
| RF feature\_set\_4 MaxAbs | 6 | 0.984 | 0.994 | 0.991 | 0.999 | 6.776 | 0.336 |
| RF feature\_set\_4 StdScale | 6 | 0.984 | 0.994 | 0.991 | 0.999 | 6.767 | 0.334 |
| SGD all features | 13 | 0.982 | 0.988 | 0.982 | 0.988 | 0.352 | 0.063 |
| LR all features | 13 | 0.963 | 0.943 | 0.962 | 0.942 | 27.399 | 0.038 |
| LR feature\_set\_1 | 7 | 0.962 | 0.942 | 0.962 | 0.941 | 4.919 | 0.066 |
| SGD feature\_set\_1 StdScale | 7 | 0.949 | 0.915 | 0.95 | 0.916 | 0.341 | 0.038 |
| SGD feature\_set\_3 StdScale | 8 | 0.949 | 0.914 | 0.949 | 0.915 | 0.412 | 0.037 |
| SGD feature\_set\_4 StdScale | 6 | 0.948 | 0.914 | 0.948 | 0.914 | 0.333 | 0.037 |
| LR feature\_set\_4 StdScale | 6 | 0.941 | 0.898 | 0.941 | 0.898 | 0.472 | 0.037 |
| LR feature\_set\_1 StdScale | 7 | 0.941 | 0.897 | 0.941 | 0.897 | 0.576 | 0.038 |
| LR feature\_set\_3 StdScale | 8 | 0.941 | 0.898 | 0.941 | 0.898 | 0.535 | 0.037 |
| LR feature\_set\_3 MaxAbs | 8 | 0.929 | 0.872 | 0.929 | 0.872 | 1.182 | 0.038 |
| LR feature\_set\_4 MaxAbs | 6 | 0.929 | 0.872 | 0.929 | 0.872 | 0.974 | 0.038 |
| LR feature\_set\_1 MaxAbs | 7 | 0.929 | 0.871 | 0.929 | 0.871 | 1.095 | 0.038 |

To make a selection of 5 algorithms from the above table, we consulted a domain expert who gave us certain threshold values for the training and testing scores.

**Thresholds**: -

* Training Balanced accuracy = <0.96
* Training Recall = <0.97
* Testing Balanced Accuracy = >=0.92
* Testing Recall = >=0.87

Based on the above thresholds, the following sets of models were obtained: -

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| model\_name | N(feature) | bac\_test | rec\_test | bac\_train | rec\_train | time\_fit | Time\_score |
| SGD feature\_set\_1 StdScale | 7 | 0.949 | 0.915 | 0.95 | 0.916 | 0.341 | 0.038 |
| SGD feature\_set\_3 StdScale | 8 | 0.949 | 0.914 | 0.949 | 0.915 | 0.412 | 0.037 |
| SGD feature\_set\_4 StdScale | 6 | 0.948 | 0.914 | 0.948 | 0.914 | 0.333 | 0.037 |
| LR feature\_set\_4 StdScale | 6 | 0.941 | 0.898 | 0.941 | 0.898 | 0.472 | 0.037 |
| LR feature\_set\_1 StdScale | 7 | 0.941 | 0.897 | 0.941 | 0.897 | 0.576 | 0.038 |
| LR feature\_set\_3 StdScale | 8 | 0.941 | 0.898 | 0.941 | 0.898 | 0.535 | 0.037 |
| LR feature\_set\_3 MaxAbs | 8 | 0.929 | 0.872 | 0.929 | 0.872 | 1.182 | 0.038 |
| LR feature\_set\_4 MaxAbs | 6 | 0.929 | 0.872 | 0.929 | 0.872 | 0.974 | 0.038 |
| LR feature\_set\_1 MaxAbs | 7 | 0.929 | 0.871 | 0.929 | 0.871 | 1.095 | 0.038 |

Among the models, we can choose the Top-5 models to tune further with our testing dataset. They are: -

1. **SGD feature\_set\_1 Standard Scaled**
2. **SGD feture\_set\_3 Standard Scaled**
3. **SGD feature\_set\_4 Standard Scaled**
4. **Logistic Regression feature\_set\_4 Standard Scaled**
5. **Logistic Regression feature\_set\_1 Standard Scaled.**

# Hyperparameter tuning of chosen Algorithms

The five models chosen after training were tested for various combinations of hyperparameters using Grid Search Cross Validation. We used the following parameter grids: -

* param\_grid\_lr1 = {'C': [0.001, 0.01, 0.1, 1], 'class\_weight': [None, 'balanced'], 'solver': ['newton-cg', 'lbfgs']} for Logistic Regression with penalty ‘l2’.
* param\_grid\_lr2 = {'C': [0.01, 0.1, 1], 'class\_weight': [None, 'balanced'],'penalty': ['l1', 'l2']} for Logistic Regression with solver ‘saga’.
* param\_grid\_sdc = {'penalty': ['l2', 'l1', 'elasticnet'], 'alpha': [0.00001, 0.0001, 0.01, 0.1], 'class\_weight': [None, 'balanced']} for Stochastic Gradient Descent Classifier with loss ‘log’.

After, running Grid Search, these are the tuned hyperparameters we obtained.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # Best hyperparameters for LOGISTIC REGRESSION models: - | | | | | |  |  |  |
| model best hyperparameters | | | | | | |  |  |
| LR1 feature\_set\_1 {'C': 0.001, 'class\_weight': 'balanced', 'solver': 'newton-cg'} | | | | | | | | |
| LR1 feature\_set\_4 {'C': 0.001, 'class\_weight': 'balanced', 'solver': 'newton-cg'} | | | | | | | | |
| LR1 feature\_set\_1 StdScaled {'C': 1, 'class\_weight': 'balanced', 'solver': 'newton-cg'} | | | | | | | | |
| LR1 feature\_set\_4 StdScaled {'C': 1, 'class\_weight': 'balanced', 'solver': 'newton-cg'} | | | | | | | | |
| LR2 feature\_set\_1 StdScaled {'C': 1, 'class\_weight': 'balanced', 'penalty': 'l1'} | | | | | | | | |
| LR2 feature\_set\_4 StdScaled {'C': 1, 'class\_weight': 'balanced', 'penalty': 'l1'} | | | | | | | | |
|  |  |  |  |  |  |  |  |  |
| # Best hyperparameters for STOCHASTIC GRADIENT DESCENT: - | | | | | | |  |  |
| model best hyperparameters | | | | | | |  |  |
| SGD feature\_set\_1 StdScaled {'alpha': 1e-05, 'class\_weight': 'balanced', 'penalty': 'l1'} | | | | | | | | |
| SGD feature\_set\_3 StdScaled {'alpha': 1e-05, 'class\_weight': 'balanced', 'penalty': 'l1'} | | | | | | | | |
| SGD feature\_set\_4 StdScaled {'alpha': 1e-05, 'class\_weight': 'balanced', 'penalty': 'l1'} | | | | | | | | |

# Cross Validating the tuned models

The tuning parameters we obtained were cross validated again to obtained the following: -

  **model feature\_count BAC\_train RCC\_train BAC\_train RCC\_train Fit\_t Score\_t**

Among the models above, we chose ***Stochastic Gradient Descent with feature set 1 that were Standard Scaled.*** The primary reason for not choosing the Logistic Regression model is that it doesn’t allow us to perform a partial fit on the new data.