Starbucks Capstone Project Report

August 22, 2021

0.1 I. Definition

0.1.1 Project Overview

The purpose of this project is to analyze from simulated Starbucks' customers data provided by Starbucks and Udacity, in order to gain insight on the relationship of the customers' attributes and their response to promotional offers being given to them.

Once in a while, Starbucks sends promotional offer to its mobile customers and the data gained from it are being used to simulate the dataset this project is based on.

From a business perspective, it is important to understand whether an offer is effective and how to personalize offers based on customers' attributes. This personalization could improve the efficacy of the promotional offer itself and might even increase the revenue, if more people are being attracted to buy based on that personalized offer.

Some research has been conducted using machine learning model to classify things based on marketing data. It is a good practice to learn from them before solving problems in the marketing area and using the marketing data. The followings are some of them:

- https://www.researchgate.net/publication/282657577_Marketing_Research_Data_Classification_by_Means_of_Machine_Learning_Methods
- https://www.researchgate.net/publication/260707025_Using_Neural_Networks_for_Marketing_Research_Data_Classification

Also, this project is a great fit for students of Data Science or Machine Learning to tinker on, since it would widen their experience on a different kind of dataset and also for them to engineer features that matter and algorithm that would perform best.

0.1.2 Problem Statement

Would a customer respond to a particular offer?

- The problem of this project would be a classification problem: there needs to be a classification of whether a promotional offer is going to make a customer reponds or not.
- An approach to this problem would be to see if there could be a pattern emerged from customer's attributes and the promotional offer's data (duration, rewards, etc.) to determine whether a customer would respond to a promotional offer: customer's attributes and promotional offer's data to be the inputs and a binary classification of responding or not would be the output.
- Thus, a model needs to be built based on those inputs and it is expected to output a binary classification: whether a customer would respond (with the value of `1`) or not (with the value of `0`).

0.1.3 Metrics

The metric to be used for the evaluation of this project would be the accuracy level, since it is more important to maximize the true positives and true negatives (whether an offer would get a response), rather than to minimize the false positives or false negatives (customers get an offer s/he would not respond to).

Additionally, a preliminary data exploration suggests a slight imbalance, but still not large enough for the accuracy metrics to render to be a bad metric. Even so, an F1 metric would also be analyzed further to complement the accuracy measure.

0.2 II. Analysis

0.2.1 Data Exploration

1. Portfolio Data Exploration The portfolio dataset only has ten data points or offers, thus it is easy to explore the data without using any python coding. From the overview of the data above, it can be seen that `channels` contains a list of channels where the promotional offers are sent with and that the `offer_type` contains categorical value of what kind of offer it is. Both of those columns could be expanded by using the one-hot encoding method in the data preprocessing step.

Also, there is no abnormalities in the values of the data. One thing to notice would be that the `informational` offer does not give any kind of rewards to the customers: it might contain only informational news, e.g. highlighting the product features, as the type suggests.

2. Profile Data Exploration There are 17,000 data points or customers in the profile dataset. Out of that number, there are 2,175 missing values or around 12.79% from the `gender` and `income` columns.

For the missing values in the income column, I would impute it with the median income, as it would predict the central tendency of the income column. The trade-off is that there would be less variation from the income feature: the income feature would have less predictive power.

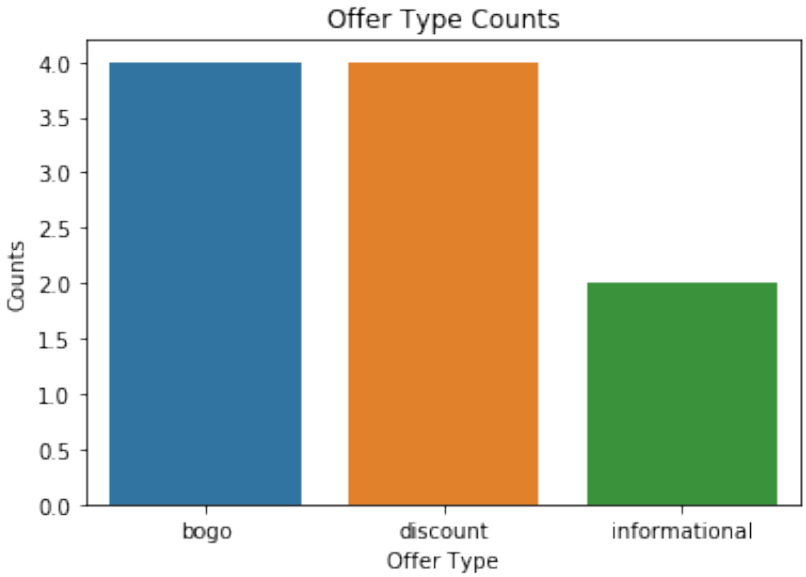
However, since it only comprises of 12.79% of the total data, it might prove useful to still include these data points and impute the missing value, since removing them might decrease the variation of the other features, such as the age feature.

For the missing values in the gender column, they would be taken into account by the use of one-hot encoding for the gender column: inferred by having 0 value in the M, F, and O columns (all the values in the gender column). I deliberately not imputing anything, since it might be useful to assume that an absence of the value here might have a predictive quality.

Also, for the `became_member_on` column, since the values are now on a format of integer, they would be first processed by casting it into a datetime format, then they would be formatted further as timestamp, and then they would be scaled or normalized.

Further, there are age values of `118` quite often showing in the data point which has missing value in the gender and income columns. This might be a placeholder value for data point with missing data. Depending on the distribution, which is going to be explored in the below step, I might replace these values with the median age instead.

3. Transcript Data Exploration There seems to be no missing values in this dataset, which is great. However, there are some tricky structure of the values, which corresponds exactly on what the event type is, e.g. there are two ways to get the offer ID: `offer_id` and `offer id`, depending



Offer Types

on the value of the event. This would be relevant in how to preprocess the data to get the data points which are to be the input of the models to be trained below.

In general, I would like to produce a dataset whose data points consist of costumer's attributes, the offer's attributes, and the class of whether the customer responds. A response would then be defined as whether an offer being received by the customer (inferred from the `offer_id` key in the value column and the `offer received` value in the event column) ended up being used by the customer (inferred from the `offer id` key in the value column and the `offer completed` value in the event column).

Thus, I might need to produce new columns for the `offer_id` inferred from the `value` column in this dataset and it might be the case that the key of `reward` and `amount` would not be included, since we only care about whether the customers respond, not how much revenue from transaction has an offer generated.

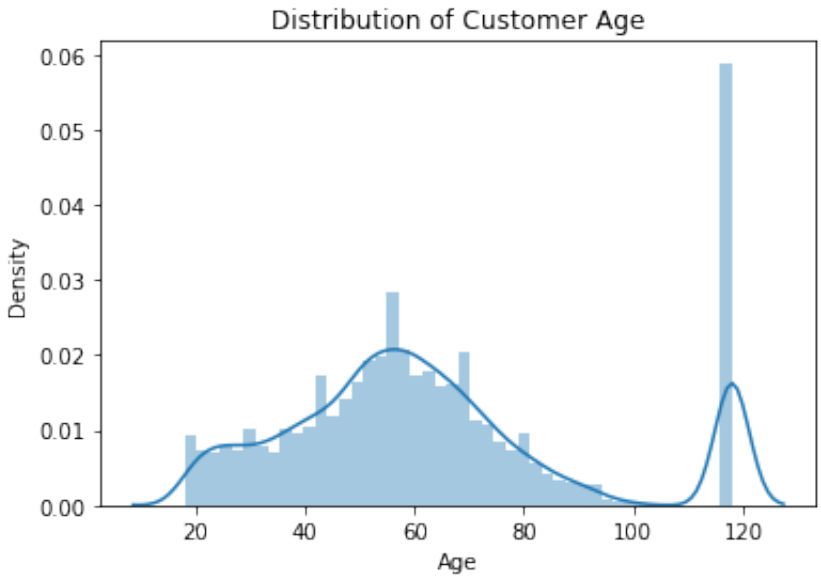
0.2.2 Exploratory Visualization

1. Portfolio Data Visualization There are four bogos, four discounts, and two informationals type of offers in the portfolio dataset. The imbalance here (of the informational type being only half as many as the others) does not really relevant, since the relevant one would be the imbalance of the data for each label class (the respond and the non-repond class).

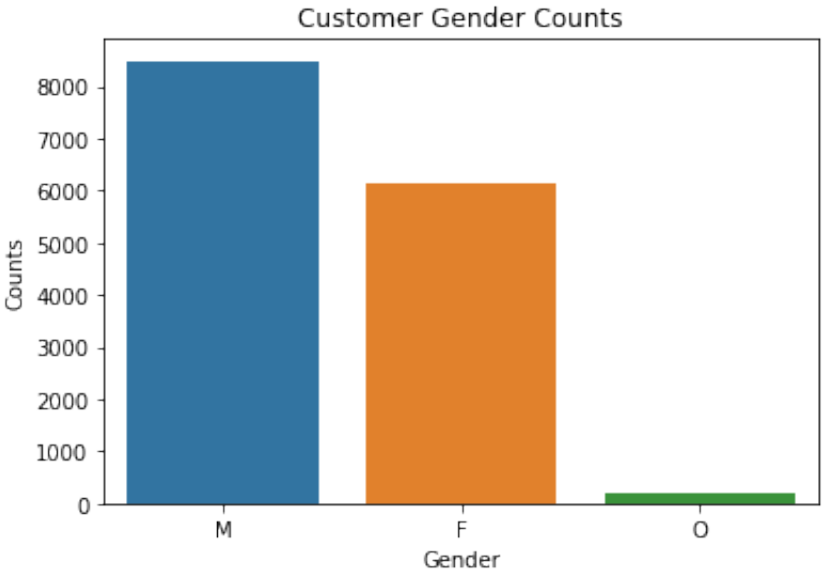
2. Profile Data Visualization It is highly likely that the age of `118` found above is the result of a placeholder value assigned whenever there are no data for the customer (gender and income). Thus, I will replace the value with the median age instead.

There is imbalance by gender, but there should be no relevant consequences for this, as the only relevant imbalance would be the one based on the class label.

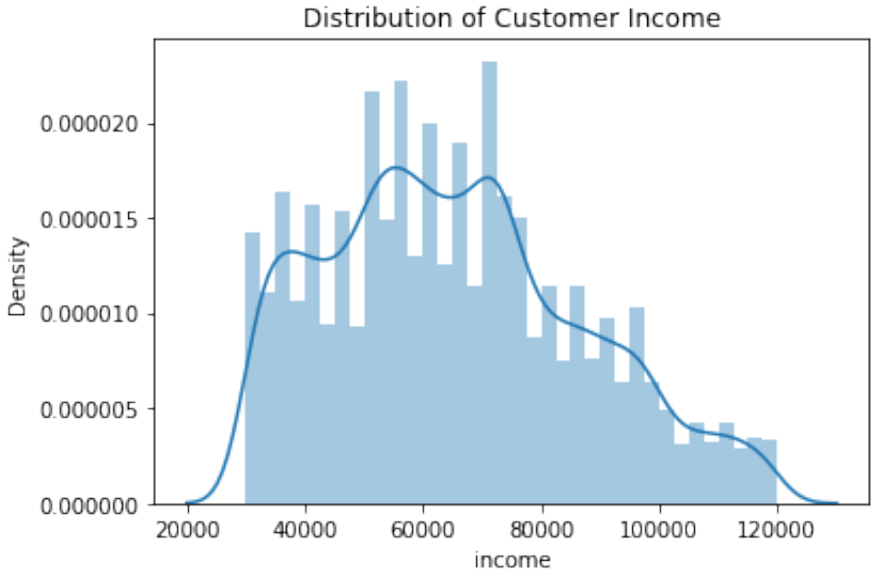
The income data is quite good: it could resemble a rightly-skewed distribution. Thus standardizing the values of this column could be done without any other preprocessing step.



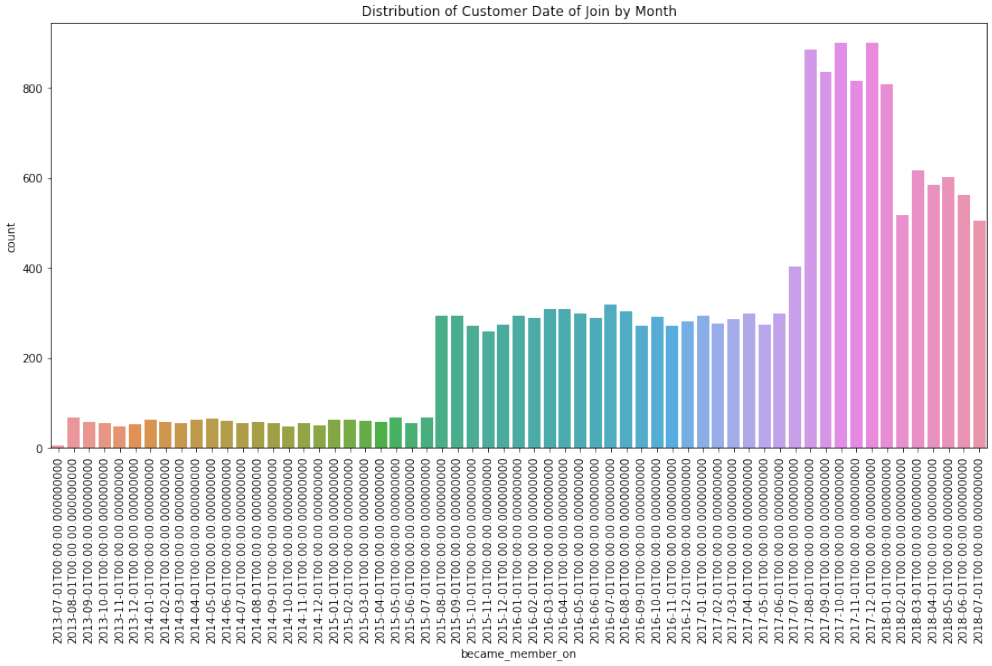
Customer Age



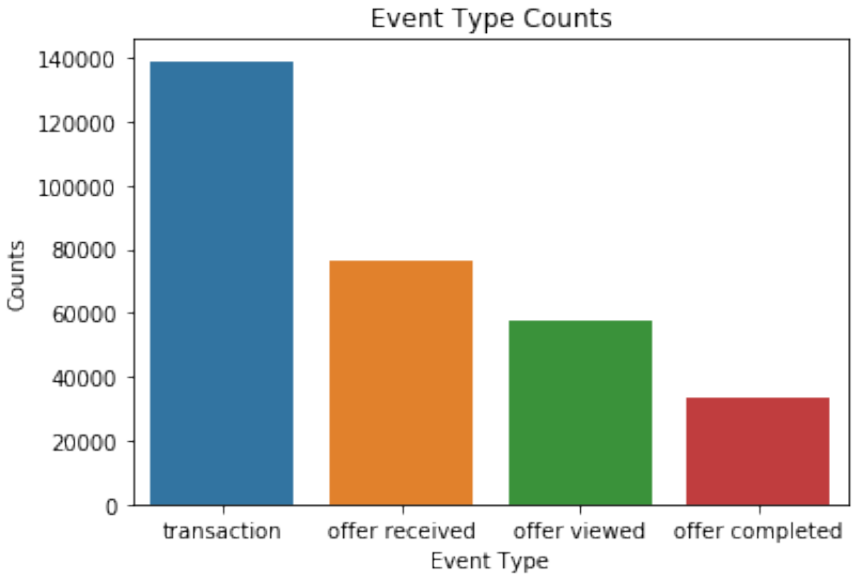
Customer Gender



Customer Income



Customer Date of Join



Event Type

The distribution of the customer date of join is rather imbalance: the early dates have few customer join counts, while the later dates have many. On whether to use day or month casting, it might be useful to just stick with the day format, as it would result in a more granular value when doing scaling on the preprocessing step.

3. Transcript Data Visualization The number of offers being used is about a half of the number of offers given. However, it cannot be assumed that the offers used are the result the offer being given to customer: the customer might not even be aware of the offer s/he received, right until after the transaction. This would be relevant in the way I am going to define what counts as a customer responding to an offer: s/he needs to be aware of it (offer viewed) before the transaction occurs.

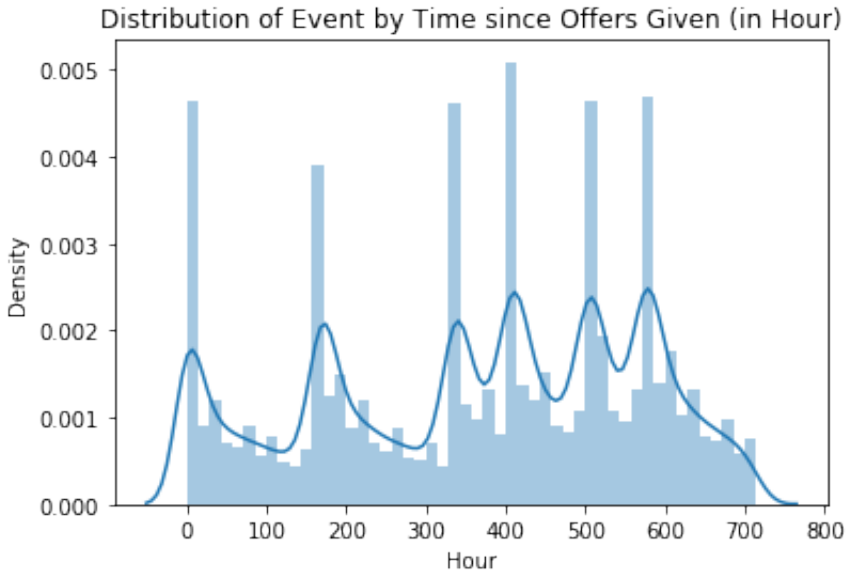
There seems to be a pattern of events based on the hour after the offers given. The huge spike in the beginning might be dominated by the events of offers being sent to the users. The other spikes would be unknownable by direct observation, but it suggests that this data might be a good feature in determining response for time-series forecasting.

Thus, it would not be a relevant feature then for this project's problem, since we would like to know whether a customer would respond: the problem is agnostic to *when* does the customer responds or whether s/he responds *given a time* after an offer is received.

0.2.3 Algorithms and Techniques

1. Preprocess the data: the rationale are provided from the data exploration above.

- Dropping duplicate rows from all datasets
- Impute missing values of customers income with the median of income data.
- Replace customers age of `118` with the median of age data.
- One-hot encode these columns: customer gender, offer channel, and offer type.
- Cast `became_member_on` column values into a timestamp of day format.



Time of Transaction

- Replace the `value` column of the transcript DF with the `offer_id` column with values inferred.
- Group the joined dataframe on customer ID then offer ID.
- Join the grouped dataframe with portfolio and profiles dataframes to add customers and offers attributes.
- Add labels to the data points by inferring response only to those customers who have received, viewed, and complete the offer.
- Min-max scale the values of the numerical columns (excluding the one-hot encoded).
- Split the data into train and test dataset.

2. Obtain a benchmark result

- Train a basic Naive Bayes algorithm to obtain a benchmark result.

3. Train other algorithm

- Train the following algorithms with grid-search and cross-validation:
  - Logistic Regression
  - Grid search parameters:
    - penalty: ['l1', 'l2']
    - C: [1, 10, 100, 1000]
    - max_iter: [25, 50, 100]}
  - Cross validation fold: 5
  - Decision Tree
  - Grid search parameters:
    - criterion: ['gini', 'entropy']
    - max_depth: [None, 2, 5]
    - min_samples_split: [2, 5, 10]}

- Cross validation fold: 5
- Random Forest
- Grid search parameters:
  - n_estimators: [25, 50, 100]
  - criterion: ['gini', 'entropy']
  - max_depth: [None, 2, 5]
  - min_samples_split: [2, 5, 10]}
- Cross validation fold: 5
- Support Vector Machine
- Grid search parameters:
  - kernel: ['linear', 'rbf']
  - C: [1, 10, 100]
  - max_iter: [-1, 15, 30]}
- Cross validation fold: 5

4. Determine the model with the best result

0.2.4 Benchmark

A Naive Bayes algorithm would be used to be the benchmark algorithm in predicting whether a customer would respond to a particular offer. This is due to that Naive Bayes is a basic algorithm in solving a binary classification problem, without any presumption on what algorithm would best predict the classification, such as that of the problem statement above.

0.3 III. Methodology

0.3.1 Data Preprocessing

Dropping duplicates There is no need to drop duplicates on the portfolio dataset, since it can be seen from the ten data points, that there is no duplicate rows in the dataframe.

As for the profile dataset, there also no duplicates found.

However, for the transcript dataset, there are 397 duplicates which are dropped from the dataset. The final count of the data points is then to be 306,137 data points.

Impute missing values The missing valus imputed are only for the customers' income and this is done by imputing the median income.

Replace invalid age values Since it is known from the exploratory step that the age value of 118 is a placeholder for a missing customer data point, they have to be replaced by a value, in which I chose the median of the customers age.

One-hot encoding categorical values One-hot encoding is conducted for the following features: * Customer gender * Offer channels * Offer type

Cast `became_member_on` values The `became_member_on` feature is being parsed as a timestamp based on day.

Replace the `value` column with `offer_id` values As for the `value` feature in the transcript dataset, it is parsed to get the offer ID values, which are then placed back to the column, which is renamed to be `offer_id`.

Grouped transcript based on customer ID and offer ID and infer response label The transcript dataset is transformed into a dataset whose data points are customer ID, offer ID, and response label. It is done by grouping the dataset by the customer ID and offer ID and then infer whether an offer is used based on whether a customer received, viewed, and completed an offer at least once.

Join dataframes The three datasets are joined based on the customer ID and offer ID, so that each data point contains a customer ID, an offer ID, a response label, customers' attributes, and offer's atributes.

Scale the values The values of the dataset are scaled to be on the range of 0-1.

Split data into train and test datasets The joined dataset is then split by 3:1 for the purpose of having a dataset for training and a dataset for testing the trained models.

0.3.2 Implementation

The benchmark model with Naive Bayes algorithm is trained and resulting in the following result:

```
Accuracy: 0.6026418910377955

             precision    recall  f1-score   support

          0       1.00      0.35      0.52      9671
          1       0.49      1.00      0.66      6151

avg / total       0.80      0.60      0.57     15822
```

0.3.3 Refinement

Several other models are trained based on different algorithm and parameters to refine the machine learning model which would be the solution to the classification problem of this project.

Train & evaluate Logistic Regression algorithm The result of training the Logistic Regression model with grid-search and cross-validation is the folowing:

```
# Tuning hyper-parameters for accuracy

Best parameters set found on development set:

{'C': 1, 'max_iter': 25, 'penalty': 'l2'}

Grid scores on development set:
```

```
0.753 (+/-0.004) for {'C': 1, 'max_iter': 25, 'penalty': 'l1'}
0.753 (+/-0.004) for {'C': 1, 'max_iter': 25, 'penalty': 'l2'}
0.753 (+/-0.004) for {'C': 1, 'max_iter': 50, 'penalty': 'l1'}
0.753 (+/-0.004) for {'C': 1, 'max_iter': 50, 'penalty': 'l2'}
0.753 (+/-0.004) for {'C': 1, 'max_iter': 100, 'penalty': 'l1'}
0.753 (+/-0.004) for {'C': 1, 'max_iter': 100, 'penalty': 'l2'}
0.753 (+/-0.004) for {'C': 10, 'max_iter': 25, 'penalty': 'l1'}
0.753 (+/-0.004) for {'C': 10, 'max_iter': 25, 'penalty': 'l2'}
0.753 (+/-0.004) for {'C': 10, 'max_iter': 50, 'penalty': 'l1'}
0.753 (+/-0.004) for {'C': 10, 'max_iter': 50, 'penalty': 'l2'}
0.753 (+/-0.004) for {'C': 10, 'max_iter': 100, 'penalty': 'l1'}
0.753 (+/-0.004) for {'C': 10, 'max_iter': 100, 'penalty': 'l2'}
0.753 (+/-0.004) for {'C': 100, 'max_iter': 25, 'penalty': 'l1'}
0.753 (+/-0.004) for {'C': 100, 'max_iter': 25, 'penalty': 'l2'}
0.753 (+/-0.004) for {'C': 100, 'max_iter': 50, 'penalty': 'l1'}
0.753 (+/-0.004) for {'C': 100, 'max_iter': 50, 'penalty': 'l2'}
0.753 (+/-0.004) for {'C': 100, 'max_iter': 100, 'penalty': 'l1'}
0.753 (+/-0.004) for {'C': 100, 'max_iter': 100, 'penalty': 'l2'}
0.753 (+/-0.004) for {'C': 1000, 'max_iter': 25, 'penalty': 'l1'}
0.753 (+/-0.004) for {'C': 1000, 'max_iter': 25, 'penalty': 'l2'}
0.753 (+/-0.004) for {'C': 1000, 'max_iter': 50, 'penalty': 'l1'}
0.753 (+/-0.004) for {'C': 1000, 'max_iter': 50, 'penalty': 'l2'}
0.753 (+/-0.004) for {'C': 1000, 'max_iter': 100, 'penalty': 'l1'}
0.753 (+/-0.004) for {'C': 1000, 'max_iter': 100, 'penalty': 'l2'}

Detailed classification report:

The model is trained on the full training set.
The scores are computed on the full test set.

             precision    recall  f1-score   support

          0       0.80      0.80      0.80      9671
          1       0.69      0.69      0.69      6151

avg / total       0.76      0.76      0.76     15822


# Tuning hyper-parameters for f1

Best parameters set found on development set:

{'C': 1, 'max_iter': 25, 'penalty': 'l2'}

Grid scores on development set:

0.682 (+/-0.005) for {'C': 1, 'max_iter': 25, 'penalty': 'l1'}
```

```
0.682 (+/-0.006) for {'C': 1, 'max_iter': 25, 'penalty': 'l2'}
0.682 (+/-0.006) for {'C': 1, 'max_iter': 50, 'penalty': 'l1'}
0.682 (+/-0.006) for {'C': 1, 'max_iter': 50, 'penalty': 'l2'}
0.682 (+/-0.006) for {'C': 1, 'max_iter': 100, 'penalty': 'l1'}
0.682 (+/-0.006) for {'C': 1, 'max_iter': 100, 'penalty': 'l2'}
0.682 (+/-0.006) for {'C': 10, 'max_iter': 25, 'penalty': 'l1'}
0.682 (+/-0.005) for {'C': 10, 'max_iter': 25, 'penalty': 'l2'}
0.682 (+/-0.005) for {'C': 10, 'max_iter': 50, 'penalty': 'l1'}
0.682 (+/-0.005) for {'C': 10, 'max_iter': 50, 'penalty': 'l2'}
0.682 (+/-0.005) for {'C': 10, 'max_iter': 100, 'penalty': 'l1'}
0.682 (+/-0.005) for {'C': 10, 'max_iter': 100, 'penalty': 'l2'}
0.682 (+/-0.006) for {'C': 100, 'max_iter': 25, 'penalty': 'l1'}
0.682 (+/-0.006) for {'C': 100, 'max_iter': 25, 'penalty': 'l2'}
0.682 (+/-0.006) for {'C': 100, 'max_iter': 50, 'penalty': 'l1'}
0.682 (+/-0.006) for {'C': 100, 'max_iter': 50, 'penalty': 'l2'}
0.682 (+/-0.006) for {'C': 100, 'max_iter': 100, 'penalty': 'l1'}
0.682 (+/-0.006) for {'C': 100, 'max_iter': 100, 'penalty': 'l2'}
0.682 (+/-0.005) for {'C': 1000, 'max_iter': 25, 'penalty': 'l1'}
0.682 (+/-0.006) for {'C': 1000, 'max_iter': 25, 'penalty': 'l2'}
0.682 (+/-0.006) for {'C': 1000, 'max_iter': 50, 'penalty': 'l1'}
0.682 (+/-0.006) for {'C': 1000, 'max_iter': 50, 'penalty': 'l2'}
0.682 (+/-0.006) for {'C': 1000, 'max_iter': 100, 'penalty': 'l1'}
0.682 (+/-0.006) for {'C': 1000, 'max_iter': 100, 'penalty': 'l2'}

Detailed classification report:

The model is trained on the full training set.
The scores are computed on the full test set.

             precision    recall  f1-score   support

          0       0.80      0.80      0.80      9671
          1       0.69      0.69      0.69      6151

avg / total       0.76      0.76      0.76     15822
```

Train & evaluate Decision Tree algorithm The result of training the Decision Tree model with grid-search and cross-validation is the folowing:

```
# Tuning hyper-parameters for accuracy

Best parameters set found on training set:

{'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 2}

Grid scores on training set:

0.727 (+/-0.009) for {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 2}
```

```
0.730 (+/-0.009) for {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 5}
0.733 (+/-0.006) for {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 10}
0.657 (+/-0.011) for {'criterion': 'gini', 'max_depth': 2, 'min_samples_split': 2}
0.657 (+/-0.011) for {'criterion': 'gini', 'max_depth': 2, 'min_samples_split': 5}
0.657 (+/-0.011) for {'criterion': 'gini', 'max_depth': 2, 'min_samples_split': 10}
0.750 (+/-0.008) for {'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 2}
0.750 (+/-0.008) for {'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 5}
0.750 (+/-0.008) for {'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 10}
0.727 (+/-0.006) for {'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 2}
0.726 (+/-0.004) for {'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 5}
0.730 (+/-0.004) for {'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 10}
0.657 (+/-0.011) for {'criterion': 'entropy', 'max_depth': 2, 'min_samples_split': 2}
0.657 (+/-0.011) for {'criterion': 'entropy', 'max_depth': 2, 'min_samples_split': 5}
0.657 (+/-0.011) for {'criterion': 'entropy', 'max_depth': 2, 'min_samples_split': 10}
0.750 (+/-0.008) for {'criterion': 'entropy', 'max_depth': 5, 'min_samples_split': 2}
0.750 (+/-0.008) for {'criterion': 'entropy', 'max_depth': 5, 'min_samples_split': 5}
0.750 (+/-0.008) for {'criterion': 'entropy', 'max_depth': 5, 'min_samples_split': 10}

Detailed classification report:

The model is trained on the full training set.
The scores are computed on the full test set.

             precision    recall  f1-score   support

          0       0.80      0.79      0.79      9671
          1       0.67      0.70      0.69      6151

avg / total       0.75      0.75      0.75     15822


# Tuning hyper-parameters for f1

Best parameters set found on training set:

{'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 2}

Grid scores on training set:

0.645 (+/-0.009) for {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 2}
0.638 (+/-0.009) for {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 5}
0.646 (+/-0.005) for {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 10}
0.683 (+/-0.007) for {'criterion': 'gini', 'max_depth': 2, 'min_samples_split': 2}
0.683 (+/-0.007) for {'criterion': 'gini', 'max_depth': 2, 'min_samples_split': 5}
0.683 (+/-0.007) for {'criterion': 'gini', 'max_depth': 2, 'min_samples_split': 10}
0.685 (+/-0.018) for {'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 2}
0.685 (+/-0.018) for {'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 5}
0.685 (+/-0.018) for {'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 10}
```

```
0.646 (+/-0.006) for {'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 2}
0.637 (+/-0.006) for {'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 5}
0.644 (+/-0.006) for {'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 10}
0.683 (+/-0.007) for {'criterion': 'entropy', 'max_depth': 2, 'min_samples_split': 2}
0.683 (+/-0.007) for {'criterion': 'entropy', 'max_depth': 2, 'min_samples_split': 5}
0.683 (+/-0.007) for {'criterion': 'entropy', 'max_depth': 2, 'min_samples_split': 10}
0.685 (+/-0.018) for {'criterion': 'entropy', 'max_depth': 5, 'min_samples_split': 2}
0.685 (+/-0.018) for {'criterion': 'entropy', 'max_depth': 5, 'min_samples_split': 5}
0.685 (+/-0.018) for {'criterion': 'entropy', 'max_depth': 5, 'min_samples_split': 10}

Detailed classification report:

The model is trained on the full training set.
The scores are computed on the full test set.

             precision    recall  f1-score   support

          0       0.80      0.79      0.79      9671
          1       0.67      0.70      0.69      6151

avg / total       0.75      0.75      0.75     15822
```

Train & evaluate Random Forest algorithm The result of training the Random Forest model with grid-search and cross-validation is the folowing:

```
# Tuning hyper-parameters for accuracy

Best parameters set found on training set:

{'criterion': 'gini', 'max_depth': None, 'min_samples_split': 10, 'n_estimators': 100}

Grid scores on training set:

0.758 (+/-0.003) for {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 2, 'n_estimat
0.759 (+/-0.004) for {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 2, 'n_estimat
0.761 (+/-0.004) for {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 2, 'n_estimat
0.763 (+/-0.006) for {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 5, 'n_estimat
0.764 (+/-0.007) for {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 5, 'n_estimat
0.765 (+/-0.005) for {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 5, 'n_estimat
0.769 (+/-0.008) for {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 10, 'n_estima
0.772 (+/-0.004) for {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 10, 'n_estima
0.772 (+/-0.005) for {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 10, 'n_estima
0.746 (+/-0.013) for {'criterion': 'gini', 'max_depth': 2, 'min_samples_split': 2, 'n_estimators
0.743 (+/-0.011) for {'criterion': 'gini', 'max_depth': 2, 'min_samples_split': 2, 'n_estimators
0.750 (+/-0.007) for {'criterion': 'gini', 'max_depth': 2, 'min_samples_split': 2, 'n_estimators
0.746 (+/-0.016) for {'criterion': 'gini', 'max_depth': 2, 'min_samples_split': 5, 'n_estimators
0.751 (+/-0.004) for {'criterion': 'gini', 'max_depth': 2, 'min_samples_split': 5, 'n_estimators
0.746 (+/-0.005) for {'criterion': 'gini', 'max_depth': 2, 'min_samples_split': 5, 'n_estimators
```

```
0.736 (+/-0.014) for {'criterion': 'gini', 'max_depth': 2, 'min_samples_split': 10, 'n_estimator
0.746 (+/-0.010) for {'criterion': 'gini', 'max_depth': 2, 'min_samples_split': 10, 'n_estimator
0.749 (+/-0.006) for {'criterion': 'gini', 'max_depth': 2, 'min_samples_split': 10, 'n_estimator
0.761 (+/-0.006) for {'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 2, 'n_estimators
0.764 (+/-0.006) for {'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 2, 'n_estimators
0.765 (+/-0.003) for {'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 2, 'n_estimators
0.762 (+/-0.006) for {'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 5, 'n_estimators
0.763 (+/-0.007) for {'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 5, 'n_estimators
0.764 (+/-0.004) for {'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 5, 'n_estimators
0.764 (+/-0.005) for {'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 10, 'n_estimator
0.764 (+/-0.004) for {'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 10, 'n_estimator
0.764 (+/-0.007) for {'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 10, 'n_estimator
0.756 (+/-0.008) for {'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 2, 'n_esti
0.760 (+/-0.003) for {'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 2, 'n_esti
0.760 (+/-0.008) for {'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 2, 'n_esti
0.761 (+/-0.003) for {'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 5, 'n_esti
0.764 (+/-0.006) for {'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 5, 'n_esti
0.765 (+/-0.004) for {'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 5, 'n_esti
0.768 (+/-0.007) for {'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 10, 'n_est
0.769 (+/-0.008) for {'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 10, 'n_est
0.771 (+/-0.002) for {'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 10, 'n_est
0.741 (+/-0.015) for {'criterion': 'entropy', 'max_depth': 2, 'min_samples_split': 2, 'n_estimat
0.748 (+/-0.008) for {'criterion': 'entropy', 'max_depth': 2, 'min_samples_split': 2, 'n_estimat
0.747 (+/-0.007) for {'criterion': 'entropy', 'max_depth': 2, 'min_samples_split': 2, 'n_estimat
0.744 (+/-0.012) for {'criterion': 'entropy', 'max_depth': 2, 'min_samples_split': 5, 'n_estimat
0.747 (+/-0.015) for {'criterion': 'entropy', 'max_depth': 2, 'min_samples_split': 5, 'n_estimat
0.749 (+/-0.011) for {'criterion': 'entropy', 'max_depth': 2, 'min_samples_split': 5, 'n_estimat
0.747 (+/-0.012) for {'criterion': 'entropy', 'max_depth': 2, 'min_samples_split': 10, 'n_estima
0.749 (+/-0.005) for {'criterion': 'entropy', 'max_depth': 2, 'min_samples_split': 10, 'n_estima
0.751 (+/-0.008) for {'criterion': 'entropy', 'max_depth': 2, 'min_samples_split': 10, 'n_estima
0.761 (+/-0.006) for {'criterion': 'entropy', 'max_depth': 5, 'min_samples_split': 2, 'n_estimat
0.764 (+/-0.004) for {'criterion': 'entropy', 'max_depth': 5, 'min_samples_split': 2, 'n_estimat
0.764 (+/-0.005) for {'criterion': 'entropy', 'max_depth': 5, 'min_samples_split': 2, 'n_estimat
0.763 (+/-0.008) for {'criterion': 'entropy', 'max_depth': 5, 'min_samples_split': 5, 'n_estimat
0.765 (+/-0.007) for {'criterion': 'entropy', 'max_depth': 5, 'min_samples_split': 5, 'n_estimat
0.763 (+/-0.004) for {'criterion': 'entropy', 'max_depth': 5, 'min_samples_split': 5, 'n_estimat
0.762 (+/-0.009) for {'criterion': 'entropy', 'max_depth': 5, 'min_samples_split': 10, 'n_estima
0.763 (+/-0.004) for {'criterion': 'entropy', 'max_depth': 5, 'min_samples_split': 10, 'n_estima
0.765 (+/-0.004) for {'criterion': 'entropy', 'max_depth': 5, 'min_samples_split': 10, 'n_estima

Detailed classification report:

The model is trained on the full training set.
The scores are computed on the full test set.

              precision    recall  f1-score   support

           0       0.81      0.82      0.81      9671
```

```
          1       0.71      0.69      0.70      6151

avg / total       0.77      0.77      0.77     15822


# Tuning hyper-parameters for f1

Best parameters set found on training set:

{'criterion': 'entropy', 'max_depth': 5, 'min_samples_split': 5, 'n_estimators': 100}

Grid scores on training set:

0.684 (+/-0.004) for {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 2, 'n_estimat
0.683 (+/-0.003) for {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 2, 'n_estimat
0.685 (+/-0.003) for {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 2, 'n_estimat
0.691 (+/-0.004) for {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 5, 'n_estimat
0.694 (+/-0.007) for {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 5, 'n_estimat
0.692 (+/-0.006) for {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 5, 'n_estimat
0.696 (+/-0.006) for {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 10, 'n_estima
0.697 (+/-0.008) for {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 10, 'n_estima
0.699 (+/-0.007) for {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 10, 'n_estima
0.615 (+/-0.091) for {'criterion': 'gini', 'max_depth': 2, 'min_samples_split': 2, 'n_estimators
0.618 (+/-0.037) for {'criterion': 'gini', 'max_depth': 2, 'min_samples_split': 2, 'n_estimators
0.635 (+/-0.018) for {'criterion': 'gini', 'max_depth': 2, 'min_samples_split': 2, 'n_estimators
0.625 (+/-0.023) for {'criterion': 'gini', 'max_depth': 2, 'min_samples_split': 5, 'n_estimators
0.624 (+/-0.088) for {'criterion': 'gini', 'max_depth': 2, 'min_samples_split': 5, 'n_estimators
0.626 (+/-0.031) for {'criterion': 'gini', 'max_depth': 2, 'min_samples_split': 5, 'n_estimators
0.605 (+/-0.030) for {'criterion': 'gini', 'max_depth': 2, 'min_samples_split': 10, 'n_estimator
0.618 (+/-0.075) for {'criterion': 'gini', 'max_depth': 2, 'min_samples_split': 10, 'n_estimator
0.612 (+/-0.059) for {'criterion': 'gini', 'max_depth': 2, 'min_samples_split': 10, 'n_estimator
0.695 (+/-0.022) for {'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 2, 'n_estimators
0.696 (+/-0.017) for {'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 2, 'n_estimators
0.697 (+/-0.006) for {'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 2, 'n_estimators
0.695 (+/-0.012) for {'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 5, 'n_estimators
0.700 (+/-0.013) for {'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 5, 'n_estimators
0.699 (+/-0.011) for {'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 5, 'n_estimators
0.694 (+/-0.019) for {'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 10, 'n_estimator
0.696 (+/-0.009) for {'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 10, 'n_estimator
0.697 (+/-0.010) for {'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 10, 'n_estimator
0.683 (+/-0.003) for {'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 2, 'n_esti
0.681 (+/-0.005) for {'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 2, 'n_esti
0.686 (+/-0.005) for {'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 2, 'n_esti
0.688 (+/-0.008) for {'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 5, 'n_esti
0.690 (+/-0.006) for {'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 5, 'n_esti
0.694 (+/-0.005) for {'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 5, 'n_esti
0.693 (+/-0.007) for {'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 10, 'n_est
0.697 (+/-0.006) for {'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 10, 'n_est
```

```
0.698 (+/-0.005) for {'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 10, 'n_est
0.612 (+/-0.075) for {'criterion': 'entropy', 'max_depth': 2, 'min_samples_split': 2, 'n_estimat
0.630 (+/-0.057) for {'criterion': 'entropy', 'max_depth': 2, 'min_samples_split': 2, 'n_estimat
0.624 (+/-0.022) for {'criterion': 'entropy', 'max_depth': 2, 'min_samples_split': 2, 'n_estimat
0.620 (+/-0.051) for {'criterion': 'entropy', 'max_depth': 2, 'min_samples_split': 5, 'n_estimat
0.603 (+/-0.032) for {'criterion': 'entropy', 'max_depth': 2, 'min_samples_split': 5, 'n_estimat
0.626 (+/-0.024) for {'criterion': 'entropy', 'max_depth': 2, 'min_samples_split': 5, 'n_estimat
0.623 (+/-0.035) for {'criterion': 'entropy', 'max_depth': 2, 'min_samples_split': 10, 'n_estima
0.630 (+/-0.049) for {'criterion': 'entropy', 'max_depth': 2, 'min_samples_split': 10, 'n_estima
0.623 (+/-0.037) for {'criterion': 'entropy', 'max_depth': 2, 'min_samples_split': 10, 'n_estima
0.696 (+/-0.011) for {'criterion': 'entropy', 'max_depth': 5, 'min_samples_split': 2, 'n_estimat
0.698 (+/-0.012) for {'criterion': 'entropy', 'max_depth': 5, 'min_samples_split': 2, 'n_estimat
0.698 (+/-0.006) for {'criterion': 'entropy', 'max_depth': 5, 'min_samples_split': 2, 'n_estimat
0.694 (+/-0.020) for {'criterion': 'entropy', 'max_depth': 5, 'min_samples_split': 5, 'n_estimat
0.702 (+/-0.007) for {'criterion': 'entropy', 'max_depth': 5, 'min_samples_split': 5, 'n_estimat
0.703 (+/-0.008) for {'criterion': 'entropy', 'max_depth': 5, 'min_samples_split': 5, 'n_estimat
0.696 (+/-0.011) for {'criterion': 'entropy', 'max_depth': 5, 'min_samples_split': 10, 'n_estima
0.696 (+/-0.010) for {'criterion': 'entropy', 'max_depth': 5, 'min_samples_split': 10, 'n_estima
0.697 (+/-0.015) for {'criterion': 'entropy', 'max_depth': 5, 'min_samples_split': 10, 'n_estima

Detailed classification report:

The model is trained on the full training set.
The scores are computed on the full test set.

             precision    recall  f1-score   support

          0       0.83      0.78      0.80      9671
          1       0.68      0.74      0.71      6151

avg / total       0.77      0.76      0.77     15822
```

Train & evaluate Support Vector Machine algorithm The result of training the Support Vector Machine model with grid-search and cross-validation is the folowing:

```
Best parameters set found on training set:

{'C': 100, 'kernel': 'rbf', 'max_iter': -1}

Grid scores on training set:

0.754 (+/-0.004) for {'C': 1, 'kernel': 'linear', 'max_iter': -1}
0.542 (+/-0.010) for {'C': 1, 'kernel': 'linear', 'max_iter': 15}
0.546 (+/-0.010) for {'C': 1, 'kernel': 'linear', 'max_iter': 30}
0.760 (+/-0.008) for {'C': 1, 'kernel': 'rbf', 'max_iter': -1}
0.546 (+/-0.030) for {'C': 1, 'kernel': 'rbf', 'max_iter': 15}
0.560 (+/-0.045) for {'C': 1, 'kernel': 'rbf', 'max_iter': 30}
0.754 (+/-0.005) for {'C': 10, 'kernel': 'linear', 'max_iter': -1}
```

```
0.586 (+/-0.005) for {'C': 10, 'kernel': 'linear', 'max_iter': 15}
0.585 (+/-0.006) for {'C': 10, 'kernel': 'linear', 'max_iter': 30}
0.769 (+/-0.006) for {'C': 10, 'kernel': 'rbf', 'max_iter': -1}
0.583 (+/-0.011) for {'C': 10, 'kernel': 'rbf', 'max_iter': 15}
0.586 (+/-0.008) for {'C': 10, 'kernel': 'rbf', 'max_iter': 30}
0.754 (+/-0.004) for {'C': 100, 'kernel': 'linear', 'max_iter': -1}
0.488 (+/-0.155) for {'C': 100, 'kernel': 'linear', 'max_iter': 15}
0.614 (+/-0.057) for {'C': 100, 'kernel': 'linear', 'max_iter': 30}
0.769 (+/-0.007) for {'C': 100, 'kernel': 'rbf', 'max_iter': -1}
0.607 (+/-0.061) for {'C': 100, 'kernel': 'rbf', 'max_iter': 15}
0.615 (+/-0.029) for {'C': 100, 'kernel': 'rbf', 'max_iter': 30}

Detailed classification report:

The model is trained on the full training set.
The scores are computed on the full test set.

             precision    recall  f1-score   support

          0       0.82      0.81      0.81      9671
          1       0.70      0.72      0.71      6151

avg / total       0.77      0.77      0.77     15822


# Tuning hyper-parameters for f1
Best parameters set found on training set:

{'C': 10, 'kernel': 'rbf', 'max_iter': -1}

Grid scores on training set:

0.684 (+/-0.007) for {'C': 1, 'kernel': 'linear', 'max_iter': -1}
0.345 (+/-0.029) for {'C': 1, 'kernel': 'linear', 'max_iter': 15}
0.359 (+/-0.053) for {'C': 1, 'kernel': 'linear', 'max_iter': 30}
0.685 (+/-0.010) for {'C': 1, 'kernel': 'rbf', 'max_iter': -1}
0.349 (+/-0.033) for {'C': 1, 'kernel': 'rbf', 'max_iter': 15}
0.354 (+/-0.028) for {'C': 1, 'kernel': 'rbf', 'max_iter': 30}
0.684 (+/-0.008) for {'C': 10, 'kernel': 'linear', 'max_iter': -1}
0.650 (+/-0.003) for {'C': 10, 'kernel': 'linear', 'max_iter': 15}
0.650 (+/-0.003) for {'C': 10, 'kernel': 'linear', 'max_iter': 30}
0.704 (+/-0.002) for {'C': 10, 'kernel': 'rbf', 'max_iter': -1}
0.585 (+/-0.263) for {'C': 10, 'kernel': 'rbf', 'max_iter': 15}
0.586 (+/-0.259) for {'C': 10, 'kernel': 'rbf', 'max_iter': 30}
0.684 (+/-0.007) for {'C': 100, 'kernel': 'linear', 'max_iter': -1}
0.486 (+/-0.285) for {'C': 100, 'kernel': 'linear', 'max_iter': 15}
0.376 (+/-0.219) for {'C': 100, 'kernel': 'linear', 'max_iter': 30}
0.704 (+/-0.003) for {'C': 100, 'kernel': 'rbf', 'max_iter': -1}
```

```
0.645 (+/-0.025) for {'C': 100, 'kernel': 'rbf', 'max_iter': 15}
0.652 (+/-0.014) for {'C': 100, 'kernel': 'rbf', 'max_iter': 30}

Detailed classification report:

The model is trained on the full training set.
The scores are computed on the full test set.

             precision    recall  f1-score   support

          0       0.82      0.80      0.81      9671
          1       0.70      0.72      0.71      6151

avg / total       0.77      0.77      0.77     15822
```

0.4 IV. Results

0.4.1 Model Evaluation and Validation

Result Summary

- Naive Bayes (Benchmark)
  - Accuracy: 0.6026418910377955
  - F1-score: 0.57
- Logistic Regression
  - Best Parameters: {'C': 1, 'max_iter': 25, 'penalty': 'l2'}
  - Best Accuracy: 0.753
  - F1-score: 0.76
- Decision Tree
  - Best Parameters: {'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 2}
  - Best Accuracy: 0.750
  - F1-score: 0.75
- Random Forest
  - Best Parameters: {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 10, 'n_estimators': 100}
  - Best Accuracy: 0.772
  - F1-score: 0.77
- Support Vector Machine
  - Best Parameters: {'C': 100, 'kernel': 'rbf', 'max_iter': -1}
  - Best Accuracy: 0.769
  - F1_Score: 0.77

0.4.2 Justification

Based on the result summarized aboved, the best model found turns out to be the one using the algorithm of Random Forest with the parameters of `C=1, max_iter=25, penalty='l2'` with accuracy of 0.772 and an F1-score of 0.77.

This result is significantly better than the benchmark one: accuracy of 0.603 and F1-score of 0.57. It is also definitely better than randomly assigning an offer to a customer: a random offer would have a 0.5 chance of getting response from the customer (naive calculation of binary outcomes: response and non-response).

As to the problem of determining if an offer would get a customer response, aside from the accuracy of 0.772, the Random Forest model has 0.81 F1-score for the non-response label and 0.70 F1-score for the response label: this is better than the benchmark one of 0.52 F1-score for the non-response label and 0.66 F1-score for the response label, not to mention the naive random way of assigning offers.

0.5 V. Conclusion

0.5.1 Reflection

This project is a challenging one, since the dataset is quite large and it can answer multiple questions or problems imposing the business of Starbucks.

Not only do the data provide a sufficient number of data points to predict whether a customer would respond to a promotional offer, the fact that there exists features, such as `time`, might suggest that a prediction model based on time series could be built.

Not to mention that the `amount` and `reward` in the transcript dataset could also be used to do a regression model to predict the revenue of the business, given certain promotional offers.

As for the problem in this project, it is sufficient to provide a model that would predict a binary classification. And the result obtained by training multiple models shows that the best performing one has better chance to predict it rather than the benchmark or even by naively or randomly assigning an offer to a customer. Thus, the solution is successfully obtained and it is sufficient.

0.5.2 Improvement

A potential improvement to this project is by training other algorithm, such as XGBoost, and also using a neural network or a deep neural network as an algorithm of the solution model.

Another thing would be to try multiple other parameters to improve the accuracy and the F1-score.